UNIVERSITE CHEIKH ANTA DIOP DE DAKAR

N° d'ordre : **THS 2018-0001**

FACULTE DES SCIENCES ET TECHNIQUES

ECOLE DOCTORALE DE MATHEMATIQUES ET INFORMATIQUE (EDMI)

THESE DE DOCTORAT UNIQUE

Pour obtenir le grade de Docteur de l'Université Cheikh Anta Diop de Dakar

Spécialité : Informatique

Titre :

Gestion des communications mobiles dans les réseaux P2PSIP hiérarchiques basée sur IPv6

Directeur de thèse **: Ibrahima NIANG**

Présentée et soutenue publiquement par

Abel DIATTA

le 26 janvier 2018

Devant le jury composé de :

Président :	Hamidou DATHE	Professeur	Université Cheikh Anta Diop	Sénégal
Rapporteurs :	Cheikh SARR	Professeur	Université de Thiès	Sénégal
	Yahya SLIMANI	Professeur	Université de la Manouba	Tunisie
Examinateurs :	Olivier FLAUZAC	Professeur	Université de Reims Champagne-Ardenne	France
	Karim KONATE	Professeur	Université Cheikh Anta Diop	Sénégal
Directeur de thèse :	Ibrahima NIANG	Maître de Conférences	Université Cheikh Anta Diop	Sénégal

A mes parents défunts.
A ma famille.

Remerciements

Au termes d'un long parcours souvent parsemé d'incertitudes et de doutes, je rend grâce au Dieu Tout Puissant pour m'avoir permis de surmonter ce qui, au début, paraissait insurmontable.

Je souhaite réserver mes premiers remerciements à mon Directeur de thèse *Ibrahima NIANG*. Nul remerciement n'est de trop pour vous exprimer ma gratitude et ma reconnaissance pour m'avoir donné l'opportunité de faire une thèse. Par votre claire-voyance, vous avez toujours su m'orienter avec rigueur tout en me donnant la liberté nécessaire pour effectuer le travail. En plus de la rigueur scientifique que vous avez toujours inculquée en moi, vous saviez également trouver les bons mots pour me redonner confiance, chaque fois que soufflait le vent du découragement et du désespoir. Votre grande générosité et vos qualités humaines exceptionnelles ont fait que avez été pour moi, plus qu'un Directeur de thèse.

A tous les membres du jury, j'adresse mes très sincères remerciements. Malgré des calendriers chargés, vous avez su consacrer un temps pour évaluer le travail et accepté d'être membres du jury. Permettez moi d'abord d'exprimer mes vifs remerciements et ma forte reconnaissance à mes deux rapporteurs, les Professeurs *Cheikh SARR* et *Yahya SLIMANI* d'avoir bien épluché le rapport de thèse. Vos pertinentes remarques et suggestions ont permis une amélioration significative de la qualité du document. Mes remerciements vont ensuite à l'endroit des examinateurs, les Professeurs *Olivier FLAUZAC* et *Karim KONATE* d'avoir accepté de faire parti du jury pour évaluer le travail. Je remercie le Professeur *Hamidou DATH* d'avoir accepté de présider ce jury.

A toi *Mandicou BA*, je te remercie vivement pour le rôle déterminant que tu as joué dans ce travail. Ton expérience et tes pertinentes suggestions nous ont toujours permis d'avancer plus vite dans le travail. En plus de tes qualités scientifiques, tu as des qualités humaines qu'il me semble important de magnifier.

A toi, *Bassirou GUEYE*, j'adresse très sincèrement mes remerciements. Nos discussions ont souvent été houleuses, très houleuses même, mais toujours très constructives. Je te remercie de manière particulière pour tout le travail que nous avons accompli ensembles, notamment dans la dernière partie.

Je remercie tout le personnel du *DRTP* de l'UCAD pour l'accompagnement logis-

tique, les encouragements, l'accueil chaleureux, auxquels j'ai toujours eu droit. Je me permets de citer de manière singulière et distinguée *Papa Dame BA* à qui je renouvelle ma gratitude. Sa gentillesse, sa disponibilité, sa générosité, ses qualités relationnelles resteront à jamais gravées en moi.
Je remercie *Malick GAYE* et *François Xavier DIOP* pour le compagnonnage et tout ce qu'ils ont été pour moi. Je remercie les enseignants de la section informatique de la FST pour les encouragements renouvelés.

Je remercie *Marie Salomon SAMBOU*, *Alassane DIEDHIOU*, *Clément MANGA*, *Mélyan MENDY*, *Philippe K. BASSENE*, *Edouard DIOUF*, *Jules Fada MANGA*, tous enseignants à l'Université Assane Seck de Ziguinchor, pour leurs encouragements et leurs conseils. J'attribue une mention spéciale à *Clément MANGA* pour ses multiples soutiens mais aussi pour son bureau qu'il a mis à ma disposition.

A vous chers enseignants permanents et vacataires du Département Informatique de l'Université Assane Seck de Ziguinchor, recevez ici mes très sincères et vifs remerciements pour le compagnonnage, l'intégration, la collaboration, la considération, les encouragements, les conseils, etc.

A vous mes *AMIS*, vous me comprendrez de ne pas pouvoir citer des noms, au risque d'oublier certains qui ne seraient pourtant pas des moindres. Vous avez été à plusieurs reprises, à mes côtés, à votre manière. Sachez qu'à chaque fois que vous m'avez demandé l'état d'avancement de mes travaux, cela a été pour moi une source d'encouragements et de motivation à me surpasser. J'associe à ces remerciements, *Habib Ampa Florentin DIENG*, *Joseph DIEDHIOU*, *Théophile DIEDHIOU*, et vous mes cousins et cousines que je suis fière d'appeler *frères et soeurs*.

Je remercie très vivement feu *Damase DIATTA*, paix à son âme, Madame *Awa NDIAYE* son épouse, Monsieur et Madame *Francis DIATTA*, et toute leurs familles respectives pour m'avoir offert un bon cadre familial me permettant de mener mes études secondaires puis universitaires en toute quiétude. J'adresse également un grand merci à *Tonton Jean Marie NGANDOUL* qui est plus qu'un oncle pour moi.

A toi, *Flore Anockindo BASSE*, je te remercie très chaleureusement pour ton soutien moral, ta patience, tes encouragements et pour tout ce que tu as été et es encore pour moi. Saches que tu occupes une place importante dans ma vie.

Enfin, je rend **TOUS LES HONNEURS** à mon exceptionnelle famille. Je ne trouve pas de mots pour vous témoigner de ma reconnaissance, de ma gratitude, de ma satisfaction en vers vous. Je remercie simplement Dieu de m'avoir donné une famille si merveilleuse. Je suis fière de vous et suis convaincu que sans vous je ne suis rien.

Résumé

La gestion des communications mobiles a toujours soulevé de nombreuses problématiques de recherche. Avec l'arrivée des réseaux pair-à-pair, reconnus très dynamiques, les défis pour cette gestion des communications mobiles ne peuvent que se multiplier.

Dans cette thèse, nous proposons une nouvelle architecture pair-à-pair hiérarchique utilisant le protocole SIP dans les communications, basée sur les performances du protocole IPv6 pour la gestion de la mobilité. Notre architecture nommée $HPAMM^{IPv6}$, est organisée en trois niveaux hiérarchiques : les super noeuds complexes ou $LMAs$, les super noeuds ou $MAGs$ et les noeuds mobiles simples ou $NMSs$. L'architecture $HPAMM^{IPv6}$ permet de gérer d'une part, de la localisation et d'autre part, les *handovers* physique et logique. Pour ce faire, nous avons proposé un serveur de gestion du *handover* appelé $HoMS$, distribué à tous les noeuds de plus haut niveau, à savoir les $LMAs$. Grâce à sa table CLN, le $HoMS$ indexe tous les noeuds en communication et/ou en localisation, et permet ainsi de déclencher le processus de gestion du *handover* lorsqu'un noeud se déplace ou change de point d'attachement.

Grâce à l'architecture hiérarchique proposée et du protocole de routage déployé (consistant à envoyer les requêtes vers le plus haut niveau), la localisation des noeuds se fait dans un délai relativement court avec des coûts assez faibles en nombre de messages. Avec la solution de tolérance aux pannes intégrée, $HPAMM^{IPv6}$ permet d'obtenir de faibles délais de *handover* et de localisation même dans un contexte de forte mobilité. Les résultats obtenus des multiples expérimentations et des études analytiques montrent la robustesse et le passage à l'échelle de $HPAMM^{IPv6}$.

Mots-clés : Réseaux pair-à-pair, Mobilité $IPv6$, Tolérance aux pannes, gestion de *handover*, gestion de mobilité, protocole SIP, architecture hiérarchique.

Abstract

Mobile communication management have always raised many research issues. With peer-to-peer networks, recognized as very dynamic, challenges for managing mobile communications, become more important.

In this thesis, we propose a new hierarchical peer-to-peer architecture using SIP protocol in communications, based on the performance of IPv6 protocol for mobility management. Our architecture, named $HPAMM^{IPv6}$, is organized in three hierarchical levels : complex super-nodes or $LMAs$, super-nodes or $MAGs$ and simple mobile nodes or $NMSs$. $HPAMM^{IPv6}$ allows to manage, on the one hand, location and on the other hand, physical and logical handovers. To do this, we proposed a handover management server called $HoMS$, distributed to all nodes of higher level, namely $LMAs$. With its CLN table, the $HoMS$ server indexes all nodes in communication and/or location, and thus makes it possible to trigger handover management process when a node moves or changes its attachment point.

According to the proposed hierarchical architecture and the deployed routing protocol (consisting of sending requests to the highest level), nodes location is done in a relatively short time and relatively low costs in number of messages. With the integrated fault-tolerance solution, $HPAMM^{IPv6}$ provides low handover and location times even in a context of high mobility. Results from multiple experiments and analytical studies show the robustness and scalability of $HPAMM^{IPv6}$.

Keywords : Peer-to-Peer Networks, IPv6 Mobility, Fault Tolerance, Handover Management, Mobility Management, SIP Protocol, Hierarchical Architecture.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

ACK	Acknowledgment
CADC	Changing Address During Communication
CADL	Changing Address During Location
CLN	Communicating and Locating Node
CoA	Care-of-Address
FA	Foreign Agent
FMIPv6	Fast handover for Mobile IPv6
FPMIPv6	Fast Proxy Mobile IPv6
H-P2PSIP	Hierarchical Peer to Peer SIP
HA	Home Agent
HBMMP	Host-Based Mobility Management Protocols
HMIPv6	Hierarchical MIPv6
HoMS	Handover Management Server
LMA	Local Mobility Anchor
MAG	Mobile Access Gateway
MAP	Mobility Anchor Point
MIPv4	Mobile IPv4
MIPv6	Mobile IPv6
NAT	Network Address Translation
NBMMP	Network-Based Mobility Management Protocols
NM	Noeud Mobile
P2PSIP	Peer to Peer Session Initiation Protocol
P2P	Peer to Peer
PMIPv6	Proxy Mobile IPv6
SIP	Session Initiation Protocol
TTL	Time to Live
UAC	user agent client

Introduction générale

Avec le développement de l'Internet, l'Homme a toujours senti le besoin d'échanger des données. Pendant très longtemps, la distribution de données textes, d'applications ou de musique sur internet se faisait par téléchargements. Dans la plupart des cas, ces téléchargements prennent beaucoup de temps car les serveurs sont trop sollicités et en cas d'interruption, les utilisateurs risquent de perdre les payements effectués. Pour pallier ces manquements, il fallait répartir les tâches du serveur à plusieurs entités qui, au vu des utilisateurs, fonctionnent de manière simultanée : c'est le début des systèmes distribués. D'après [Tanen2007], un système distribué est une collection d'ordinateurs indépendants qui apparaissent aux utilisateurs du système comme un seul ordinateur. Ainsi, avec les systèmes distribués, une même tâche peut être répartie entre plusieurs processus ou processeurs qui s'exécutent de manière coordonnée. L'interruption d'un processus empêche la réalisation complète de l'ensemble [Leslie1978]. Pour pallier ces dysfonctionnements, les réseaux pair-à-pair encore appelés réseaux *peer to peer (P2P)* voient le jour.

Un réseau pair-à-pair est un réseau virtuel de noeuds et de liaisons logiques qui est construit sur un réseau physique existant dans le but de mettre en oeuvre un service réseau qui n'est pas disponible dans le réseau existant [Zheng2010]. Les réseaux *P2P* sont des systèmes distribués, dans lesquels il n'y a pas d'organisation hiérarchique ni de contrôles centralisés, à l'opposé des systèmes Client/Serveur où tout est centralisé au sein d'une même entité (Serveur). Dans un système *P2P*, tous les noeuds sont considérés égaux en fonctionnalité. Ils jouent à la fois le rôle de clients et de serveurs. Ils peuvent demander des services mais aussi les fournir à d'autres noeuds (pairs). Ainsi, lorsqu'un noeud tombe en panne, ses homologues assurant le même rôle que lui permettent au système de continuer à fonctionner [Fessa2006].

D'autre part, avec les coûts élévés des communications téléphoniques, les utilisateurs se tournent de plus en plus vers la téléphonie IP. Comme les réseaux *P2P* offrent une gestion distribuée et une tolérance aux pannes des noeuds, de nouveaux services de communication sont bâtis sur ces systèmes. C'est le cas des réseaux *peer to peer SIP (P2PSIP)* où le protocole *SIP* est distribué dans les noeuds du réseau pair-à-pair.

Cependant, dans un réseau *P2PSIP*, les pairs sont tous égaux et forment une organisation plate (non hiérarchique). Or, dans le fonctionnement réel de l'internet, les noeuds n'ont pas toujours les mêmes performances. Certains ont des vitesses de traitement et/ou des capacités de stockage plus importantes que d'autres. En mettant ensemble tous les noeuds, sans aucune catégorisation, les plus faibles peuvent détériorer les performances du réseau et augmenter les délais de latences. Or, l'intro-

duction de la téléphonie dans ces systèmes en a fait des systèmes de communication en temps réel dans lesquels le gain de temps est important. Il faut donc mettre en place une structuration permettant d'éviter ces latences éventuelles et ainsi rendre plus performantes les communications. C'est dans ce cadre que les systèmes hiérarchiques ont été mis en place donnant naissance aux réseaux *P2PSIP* hiérarchiques (*hierarchical peer to peer SIP, H-P2PSIP* en anglais) . Ils sont essentiellement organisés en deux types de noeuds : les super noeuds et les noeuds ordinaires [Zoels2006], [Yelmo2009], [Lekuo2007].

Avec la prolifération des objets de communication mobile tels que les ordinateurs portables, les smartphones, les tablettes, etc., et l'utilisation soutenue des systèmes pair-à-pair dans la communication (Skype, WhatsApp, Libon, etc.) beaucoup de problèmes surgissent. En effet, la possibilité pour les usagers de communiquer tout en se déplaçant entraîne très souvent des pertes de connexion, des pertes de paquets, des interruptions de la communication, etc. Ce qui bien sûr influe négativement sur la qualité de service. En plus, d'autres défis sont à relever notamment l'insuffisance d'adresses, le risque de saturation de la bande passante, etc. Il faut donc impérativement gérer la mobilité des usagers tout en leur permettant de communiquer sans interruption, ni dégradation du service et d'assurer la cohérence du réseau pair-à-pair ; tout cela dans des délais raisonnables.

Depuis plusieurs années les solutions de *mobile IP* ont été mises en place pour gérer la mobilité dans un réseau IP. Elles sont essentiellement de deux natures : *mobile IPv4 (MIPv4)* et *mobile IPv6 (MIPv6)* [Irfan2009, Sor2012, Aish2013, Yizhen2014]. Toutefois, les solutions actuelles s'orientent de plus en plus vers le *MIPv6* en raison de ses nombreux avantages liés à l'optimisation du temps de *handover* [Plan2010],Kha2014, MPhil2014, Leb2016, Kim2016]

Notons que, la plupart des travaux existants proposent soit des solutions de gestion de la mobilité dans IPv6 [Ana2015, Abd2016, Kha2014, Kim2016, Kris2016, Leb2016, Mus2012, Lei2016, Zoh2014], soit des solutions de gestion de la mobilité dans un système pair-à-pair [Eun2014, Cost2010, Mat2008, Waq2013, Far2006, Yel2009]. Ces travaux se sont surtout focalisés sur la mobilité physique lors des communications.

Or, dans un réseau pair-à-pair, la mobilité d'un noeud est soit logique soit physique [John2004]. Pour la mobilité physique, il s'agit d'un déplacement effectif d'un noeud d'une zone vers une autre. Quant à la mobilité logique, elle consiste en ce qu'un noeud, même étant sur place, change de point d'attachement à cause d'un changement de la topologie du réseau. D'autre part, il est important aussi de gérer la mobilité lors des phases de localisation des noeuds.

Nos travaux dans cette thèse s'intéressent à la gestion de la mobilité basée sur IPv6 pour des communications temps-réel dans un réseau *P2PSIP*. Il s'agit d'exploiter les performances offertes par les solutions de mobilité dans IPv6, la gestion optimale de la tolérance aux pannes des réseaux pair-à-pair et les bénéfices d'une hiérarchisation des réseaux *P2PSIP*, afin de proposer une architecture de gestion de mobilité optimale en termes de tolérance aux pannes mais aussi de coûts et de

délais lors des phases de localisation, de *handover*, etc. A ce titre, deux questions importantes devraient être analysées :(i) l'impact, sur la qualité de la communication, de la gestion de la mobilité dans les réseaux pair-à-pair (ii) la pertinence d'utiliser la mobilité IPv6 pour des applications de communication temps-réel basées sur un réseau pair-à-pair *SIP*.

Pour la première problématique, nous avons déjà mentionné que les réseaux pair-à-pair sont fortement dynamiques. Dans ce cas, les noeuds intègrent et quittent le système à tout instant. Cette situation entraîne, d'une part, la déconnexion d'autres noeuds initialement connectés aux noeuds partants, et d'autre part, un changement fréquent de point d'attachement. Ceci augmente fortement les coûts de maintenance des tables de routage (nouvelles adresses IP à considérer), accentue les risques d'interruption des services à cause des *handovers*[1], accroît les coûts de localisation, etc. Une gestion efficace de la mobilité dans les réseaux pair-à-pair doit minimiser tous ces coûts et éviter les interruptions des services comme la localisation et la communication, même lors des phases de *handover*.

Pour la deuxième problématique, notons que les réseaux pair-à-pair *SIP* héritent des problèmes de performance soulevés dans les réseaux *P2P* mobiles. De plus, les réseaux *P2PSIP*, étant des systèmes de communication temps-réel, sont plus exigeants en termes de délai de *handover* et de délai de localisation. Il faut donc mettre en place une solution qui amoindrit les coûts des opérations de maintenance et de mise à jour ayant un impact négatif sur ces délais. Dans ce cas, la possibilité offerte à un noeud, dans la mobilité IPv6, de se déplacer tout en conservant son adresse [Sil2006] est donc un atout important pour minimiser les nombreux coûts engendrés par la maintenance des tables de routages. L'exploitation de cette opportunité et de celle obtenue du protocole *REsource LOcation And Discovery (RELOAD)* [RFC6940] proposé par le *P2PSIP Working Group* pour la localisation et la découverte de ressources, permet d'atteindre ces objectifs d'optimisation de la gestion de la mobilité.

Contributions

Très peu de travaux ont combiné la gestion de la mobilité dans IPv6 avec un environnement pair-à-pair [Ana2015, Luo2011]. En plus, ils présentent des limites car ils ne gèrent que le *handover* physique d'un noeud durant la phase de communication. Ainsi, ces travaux ne gèrent pas le *handover* logique et n'intègrent pas la gestion de la mobilité lors des phases de localisation des noeuds. Alors que dans un système pair-à-pair la gestion du *handover* logique d'un noeud est très importante vue la nature dynamique de ces réseaux. D'autre part, la phase de localisation est primordiale car elle est au début de tout processus de communication.

Partant de ce constat, nous nous proposons de mettre en place une architecture *peer to peer SIP* hiérarchique dans laquelle, nous gérons la mobilité des noeuds en nous basant sur les solutions utilisées dans IPv6. Notre solution d'une part, prend en compte la gestion des *handovers* physique et logique lors des phases de localisation et de communication, et d'autre part, permet de réduire non seulement les

1. handover : phase qui sépare l'instant de déconnexion du noeud de son ancien point d'attachement et l'instant de sa connexion à un autre point d'attachement

délais de *handover* mais aussi les coûts de mise à jour lors de la mobilité des noeuds. Pour ce faire, dans [Diat2015], nous proposons des études comparatives des solutions d'architectures *P2PSIP* hiérarchiques. L'objectif étant de déterminer la meilleure architecture hiérarchique présentant des coûts moindres en termes de messages lors des phases de localisation. En plus, dans nos travaux sur la gestion de la mobilité dans IPv6 [Diat2016, Abe2016,Diat2017], nous proposons une modélisation analytique des coûts de mise à jour et des délais de *handover* en nous basant sur différentes solutions de mobilité IPv6 (*FMIPv6*[2] et *FPMIPv6*[3]) pour les architectures *H-P2PSIP*. Les résultats ont montré que l'architecture *H-P2PSIP* basée sur *FPMIPv6* présente des performances meilleures en termes de coûts et de délais de *handover*.

Partant de ces résultats, nous proposons une architecture [Abe2017] *P2PSIP* hiérarchique optimisée pour la gestion de la mobilité dans un réseau pair-à-pair *SIP* basé sur IPv6. Cette architecture, nommée $HPAMM^{IPv6}$, s'appuie sur trois niveaux, comme défini dans [Dian2010] et fonctionne avec le protocole *FPMIPv6*. L'organisation des niveaux est faite selon l'évaluation des noeuds sur leurs propres performances en termes de capacités. Cette architecture *peer-to-peer SIP* hiérarchique, basée sur IPv6 permet de gérer la mobilité et de réduire les délais de *handover*. Pour bien assurer la gestion de la mobilité durant les phases de localisation et/ou de communication, nous avons mis en place un serveur que nous appelons **Handover Management Server (HoMS)** ou encore serveur de gestion du *handover*. Nous l'avons distribué à tous les noeuds du premier niveau de l'architecture (noeuds dotés de plus grandes capacités). Chaque serveur *HoMS* est doté d'une table appelée **Communicating and Locating Node (CLN) Table** ou table des noeuds en communication ou en localisation. Cette table contient des informations sur les noeuds en communication ou en instance de communication.

L'architecture *P2PSIP* hiérarchique proposée est validée par des études de performances analytiques et expérimentales qui ont démontré le bon comportement de $HPAMM^{IPv6}$. De plus, un dimensionnement de l'architecture est fait afin de déterminer le nombre de noeuds de niveau inférieur rattachés à un noeud de niveau supérieur.

Plan du document

Notre document s'articule autour de six chapitres.

Chapitre 1 : Dans ce chapitre, nous proposons un aperçu sur les systèmes distribués et présentons de manière détaillée le fonctionnement des réseaux pair-à-pair, des réseaux pair-à-pair SIP et de leur organisation hiérarchique. Ces notions permettront de mieux comprendre les concepts développés dans le chapitre suivant.

Chapitre 2 : Dans ce chapitre, nous présentons les solutions de gestion de la mobilité dans IPv6 ainsi que les exigences de cette mobilité dans un réseau *P2PSIP*.

Chapitre 3 : Ce chapitre est consacré aux études comparatives, d'une part, entre les modes de gestion de la mobilité (en termes de délai et de coûts de mise à jour lors des phases de *handover*), et d'autre part, entre les différents types d'architectures pair-à-pair *SIP* hiérarchiques (en termes de coûts de localisation). Des tableaux de synthèse sont fournis pour mettre en exergue les différents résultats issus des

2. FMIPv6 (Fast handover for Mobile IPv6)
3. FPMIPv6 (Fast handover for Proxy Mobile IPv6)

comparaisons.

Chapitre 4 : Dans ce chapitre, une nouvelle architecture de gestion de la mobilité dans un réseau *P2PSIP* hiérarchique (*H-P2PSIP*) basé sur IPv6 est proposée. La solution, nommée $HPAMM^{IPv6}$, mise en place, en plus de gérer la mobilité lors de la localisation et de la communication, prend en compte les *handovers* logique et physique. Elle est basée sur une organisation des noeuds en trois niveaux selon leurs performances en termes de capacités.

Chapitre 5 : Dans ce chapitre, nous proposons des stratégies pour un bon dimensionnement du réseau et une solution de gestion de la tolérance aux pannes liée au départ de noeuds. Ce chapitre démontre le passage à l'échelle de l'architecture proposée.

Chapitre 6 : Ce chapitre est consacré aux études de performances analytiques et expérimentales de l'architecture proposée, $HPAMM^{IPv6}$. Ainsi, d'une part, notre architecture est comparée théoriquement avec la solution proposée dans SARP. D'autre part, nous présentons les résultats issus des simulations de l'architecture proposée.

Chapitre 1

Réseaux pair-à-pair et protocole SIP

1 Introduction

Dans les premiers temps de l'internet, le partage de données se faisait suivant le mode Client/Serveur. Les ressources sont stockées au sein d'une entité centrale (serveur). Lorsqu'un client souhaite trouver une ressource, il envoie une requête vers le serveur qui, après un temps de recherche dans sa base de données lui envoie en retour la ressource demandée. Les internautes devenant de plus en plus nombreux, les serveurs commençaient à atteindre leurs limites notamment en termes de capacités de stockage, de vitesse de traitement, de temps d'accès à cause des accès concurrents, etc. Il faut ajouter à cela le risque de panne du serveur qui provoquerait l'arrêt de tout le processus et la non disponibilité des ressources. Pour éviter de tels scénarios, l'idée de répartir les tâches du serveur à plusieurs entités à été développée et mise en place. C'est le concept de systèmes distribués qui sont le soubassement des réseaux pair-à-pair.

Ainsi, dans ce chapitre, il s'agit pour nous de donner un aperçu sur les systèmes distribués, et puis de détailler le fonctionnement des réseaux pair-à-pair, du protocole *SIP* et des réseaux pair-à-pair *SIP* sur lesquels se fonde notre travail.

2 Les systèmes distribués

2.1 Généralités sur les système distribués

Un système distribué est un ensemble d'entités (ordinateurs, mémoires, processus, ...) indépendantes, communiquant entre elles et vues par les utilisateurs comme une seule entité [Tanen2007]. Le rôle essentiel d'un système distribué est d'éviter de tout centraliser au sein d'une entité appelée noeud et donc de répartir les tâches initialement réservées aux serveurs aux autres constituants du système. Ceci a comme avantages immédiats le dés-engorgement des serveurs centraux et ainsi permet d'améliorer le temps de réponse. Ces tâches sont entre autres le stockage et le traitement. En distribuant les ressources et les tâches à plusieurs noeuds, le système devient plus performant. Toutefois, dans les systèmes distribués, les tâches bien qu'étant réparties sont coordonnées. L'échec de l'exécution d'une d'entre elles bloque l'achèvement des autres dépendantes de cette dernière. Pour éviter cette situation, les noeuds doivent être autonomes et pouvoir fonctionner indépendamment

des autres. En d'autres termes, un système distribué doit mettre en place des stratégies pour permettre la continuité du service même lorsqu'un de ses composants ne fonctionne plus. C'est la notion de tolérance aux pannes.

2.2 Tolérance aux pannes

Dans les systèmes distribués, plusieurs entités fonctionnent de manière coordonnée pour accomplir une tâche donnée. La panne d'un composant perturbe la bonne exécution de la tâche. Une entité est en panne lorsqu'elle se trouve dans un état qui l'empêche de remplir sa mission.
Les composants n'ayant pas les mêmes performances techniques, encore moins la même durée de vie, il est important, voire indispensable de mettre en place des solutions pour éviter que tout un système soit bloqué, à cause d'une défaillance d'un de ses éléments constitutifs. Ces solutions constituent les algorithmes de tolérance aux pannes. D'après J. C. Laprie [Lap1988], la tolérance aux pannes est l'ensemble des techniques mises en oeuvre dans un système pour lui permettre de continuer à fonctionner même en présence d'une panne d'un de ses composants.

3 Les réseaux P2P

L'acronyme *peer to peer (P2P)* vient du mot pair-à-pair qui traduit une relation d'égal à égal entre les noeuds (ordinateurs, processeurs, téléphones, etc.) du réseau. En d'autres termes, dans les réseaux pair-à-pair, il n'y a pas de rôles réservés à une catégorie de noeuds. Tous les noeuds peuvent exécuter les mêmes fonctions. Le principe du pair-à-pair est de mettre en relation des programmes, des ordinateurs pour partager des ressources directement sans intermédiaires [Fessa2006], ceci grâce à la construction d'un réseau virtuel au dessus du réseau physique. Ainsi, lorsque les liens virtuels [1] sont établis entre la source de la requête (demandeur de service) et le destinataire, les communications se passent directement entre les concernés.
Les réseaux pair-à-pair étant des systèmes distribués, ils héritent de la robustesse [2] de ces derniers et de la haute disponibilité des ressources. Ils sont extensibles, c'est-à-dire qu'un noeud a juste besoin de se connecter à un autre pour, à son tour, faire partie du réseau. Ce sont donc des systèmes ouverts. A côté de ces nombreux avantages du pair-à-pair, il est cependant important de noter qu'ils sont souvent utilisés par des utilisateurs mal intentionnés pour véhiculer des virus, des logiciels espions, des vidéos non recommandées, etc.

Le tout premier réseau pair-à-pair est Napster. Il a été créé en 1999 avec une structuration centralisée. Un serveur central est utilisé pour la recherche des ressources. Pour localiser une ressource, Napster fait une table de correspondance entre d'une part, le nom du fichier et celui du client et, d'autre part, celui du client et l'adresse IP de l'utilisateur. Ce qui d'ailleurs pose le problème de l'anonymat dans Napster.

De par leur mode de fonctionnement, les réseaux pair-à-pair sont classés en réseaux non structurés et en réseaux structurés.

1. lien virtuel : est un lien qui met en relation directe deux noeuds qui peuvent être géographiquement très éloignés

2. robustesse : la panne d'un noeud n'empêche pas au système de continuer à fonctionner

3.1 Les réseaux P2P non structurés

Un réseau *P2P* est non structuré si les liens entre les noeuds sont établis de façon arbitraire [Man2014]. Dans ce cas, pour faire partie du réseau, un nouveau pair doit connaître au moins un membre du réseau auquel se connecter ou à défaut envoyer sur le réseau un message de diffusion (*broadcast*) pour trouver les pairs déjà connectés. Dans ce type de cas, les requêtes et réponses se font par inondation. Chaque pair envoie la requête à ses voisins qui l'évaluent puis y répondent si toutefois ils disposent de la ressource demandée. Sinon ils l'acheminent à leurs voisins, ainsi de suite suivant un nombre maximal de sauts ou un temps de survie appelé *time to live (TTL)* . Parmi les réseaux pair-à-pair non structurés, nous pouvons citer quelques exemples comme Gnutella, BitTorrent, etc.

Gnutella : proposé en 2000 par Tom Pepper et Justin Frankel [Mar2003], Gnutella est un système décentralisé qui assure l'anonymat de ses utilisateurs. Depuis sa version 0.6, il a apporté une grande amélioration dans la gestion de la bande passante notamment en introduisant les *ultrapeers* (ou *superpeers*) avec les protocoles *GUESS* et *GWebCache* [Bud2003]. Les serveurs sont répartis en deux catégories : les *ultrapeers* qui sont des serveurs stables ayant une bonne connectivité et les clients (ou noeuds feuilles). Les noeuds feuilles sont rattachés à trois (3) *ultrapeers*. Chaque *ultrapeer* peut prendre jusqu'à 30 autres *ultrapeers* et 30 à 45 clients. Les *ultrapeers* indexent le contenu des clients auxquels ils sont rattachés. Ainsi, lorsqu'un client interroge un *ultrapeer* la requête peut atteindre 30 autres clients. Si la requête est sans succès, l'*ultrapeer* la retransmet aux 30 autres *ultrapeers* éventuels auxquels il est connecté, et la recherche atteint le contenu de 900 clients.

BitTorrent : c'est un réseau *peer to peer* créé dans le but d'avoir une bande passante plus grande pour les échanges et partages de fichiers. Il est basé sur le principe suivant : tout utilisateur qui télécharge une partie d'un fichier devient automatiquement serveur et peut partager la partie déjà téléchargée avec d'autres. Avec BitTorrent, l'utilisateur a la possibilité d'effectuer un téléchargement à partir de plusieurs sources à la fois[Fessa2006]. Il est également possible d'interrompre le téléchargement pour le reprendre plus tard. Son fonctionnement est régi par deux éléments essentiels : le fichier .torrent et le *tracker*[3]. Le premier est indispensable pour effectuer les téléchargements. Il contient en effet toutes les informations nécessaires au téléchargement (nom du fichier recherché, type de fichier, etc.). Pour en disposer, un client doit le rechercher sur internet ou l'obtenir d'un autre client.
Quant au *tracker*, il permet d'identifier et de connecter les clients entre eux. Lorsqu'un client est démarré, il dialogue avec le *tracker* pour le renseigner sur l'état d'avancement du téléchargement. Le *tracker* en réponse lui envoie une liste d'autres clients ou sources (par HTTP ou HTTPS) qui sont sur le même fichier. A intervalle de temps régulier, le client met à jour le *tracker* en indiquant ce qu'il a déjà téléchargé, et en retour le *tracker* lui redonne une nouvelle liste de clients auprès desquels il peut télécharger les parties manquantes.

De façon générale, les réseaux pair-à-pair non structurés sont caractérisés par la non garantie du succès de la recherche mais aussi par une surcharge de la bande

3. tracker : est un serveur qui aide à la communication entre pairs dans le logiciel BitTorrent

passante à cause des *time-to-live*.

3.2 Les réseaux P2P structurés

Un problème fondamental des réseaux *peer to peer* est comment localiser efficacement un noeud stockant des données recherchées. Les réseaux *P2P* structurés ont la particularité d'organiser le réseau en une topologie routable, c'est-à-dire que chaque pair dispose d'un identifiant permettant de le localiser en suivant un chemin déterministe. Ils sont régis par l'utilisation d'une table de hachage distribuée. La connexion d'un noeud à ces types de réseaux est souvent plus longue [Fessa2006] car nécessite l'établissement de listes de contacts appelées tables de routages. Grâce à ces tables de routage et à l'existence d'un identifiant pour chaque pair, le succès de la recherche est garanti. En plus, les réseaux structurés ont l'avantage d'être moins pollueurs que les réseaux non structurés en raison du routage déterministe, contrairement au routage par inondation dans les non structurés. Il existe beaucoup de réseaux structurés. Ils se distinguent par la simplicité qu'un noeud a pour intégrer ou quitter le réseau mais aussi pour assurer un routage efficace.

Tapestry : inspiré du maillage de Plaxton [4], Tapestry [Ben2004] utilise un mécanisme de routage assez semblable à celui de Pastry [Ant2001]. Par contre, l'identifiant unique des noeuds est codé sur 160 bits contrairement à Pastry où il est sur 128 bits. Chaque noeud maintient une table de routage à plusieurs niveaux. Un niveau contient les adresses des noeuds qui possèdent le même préfixe que le noeud courant. Chaque noeud maintient un ensemble de pointeurs sur les noeuds voisins. Pour augmenter la robustesse et l'extensibilité du réseau, Tapestry utilise plusieurs procédés. Il s'agit entre autres de l'augmentation de la réplication afin d'accroître la décentralisation, de l'envoi de messages périodiques pour garder les caches à jour, de l'utilisation du *surrogate routing* [5], etc.
Tapestry a comme avantages le fait que son réseau recouvrant est proche de la topologie du réseau physique. Toutefois il a l'inconvénient de devoir gérer par noeud une quantité importante d'informations nécessitant une mise à jour.

CAN : Content Adressable Network (CAN) est un réseau qui repose sur un espace cartésien de dimension d [Rat2001]. Chaque noeud occupe une portion de cet espace et possède un nombre de voisins en $O(d)$. Plus la dimension est grande, plus le routage est rapide. En revanche, le coût de maintien des tables de routage est élevé. Chaque noeud est responsable d'un espace appelé zone. Le noeud responsable stocke toutes les paires (clé, identifiant) se trouvant dans sa zone selon la fonction de hachage utilisée. Chaque clé est représentée sous forme de points ayant d coordonnées $c = (c1, c2, ..cd)$. Le routage d'une clé vers un noeud se fait en envoyant le message vers le voisin qui a les coordonnées les plus proches de celles du noeud recherché.
Les avantages de CAN sont multiples. En effet, il offre la possibilité de choisir une route de remplacement en cas de panne de noeuds, la possibilité d'utiliser plusieurs

4. le maillage de Plaxton consiste à répartir de manière uniforme les noeuds sur le réseau. Il utilise pour cela un algorithme de hachage tel que SHA-1 pour les identifier

5. surrogate routing : consiste à doubler chaque entrée de la table de routage et à utiliser une route de remplacement en cas de défaillance de la route principale.

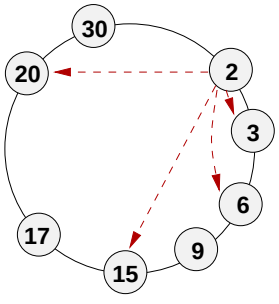
fonctions de hachage, ce qui permet de retrouver un même objet sur plusieurs noeuds, etc. Il permet également de mettre en cache les objets les plus fréquemment sollicités afin d'accroître le temps de réponse.
Toutefois, CAN surcharge beaucoup le réseau car avec son système de mise en cache, les fichiers réputés font l'objet de plusieurs sollicitations.

Chord : c'est un réseau structuré avec une topologie en anneau. Il utilise la fonction cryptographique $SHA-1$ [Stoic2001] pour la détermination des identifiants des noeuds avec un nombre maximal de bits d'adressage de $m = 160$ bits. Chaque pair dispose d'un identifiant obtenu par hachage sur son adresse IP. Les identifiants des ressources stockées sont générés par hachage sur le nom de la ressource. Les informations sont représentées sous forme de couple $(id, valeur)$ avec id représentant l'identifiant du pair sur lequel se trouve la ressource $valeur$.
Avec un nombre de bits d'adressage de $m = 160$, il est possible d'avoir jusqu'à 2^{160} pairs dans l'anneau Chord. Or le plus souvent, il n'y en a pas autant dans le réseau. En d'autres termes, la plupart des identifiants obtenus ne correspondent pas à des noeuds réellement existants. C'est pourquoi Chord définit une fonction successeur() qui permet de déterminer les successeurs d'un noeud sur l'anneau. Pour un noeud dont l'identifiant est ID, l'ensemble $\{ID + 2^0, ID + 2^1, ID + 2^2, ..., ID + 2^{m-1}\}$ représente respectivement les identifiants de son 1^{ier}, 2^{ieme}, 3^{ieme}, etc., m^{ieme} successeur sur l'anneau orienté dans le sens des aiguilles d'une montre. Chaque pair de l'anneau maintient une table de successeurs de longueur $log_2(n)$ avec $n = 2^m$ étant le nombre de noeuds dans le réseau [Furn2013].
L'identifiant du k^{ieme} successeur d'un pair dont l'identifiant est $\mathcal{P}$ est déterminé par :

$$ID(k) = (\mathcal{P} + 2^{k-1})\ modulo\ 2^m, \quad avec\ (0 < k \leq m).$$

Lorsque $[(\mathcal{P}+2^{k-1})\ modulo\ 2^m]$ ne correspond pas à l'identifiant d'un noeud existant, le successeur est alors le premier noeud de l'anneau dont l'identifiant est $\geq [(\mathcal{P} + 2^{k-1})\ modulo\ 2^m]$.

La figure 1.1 montre la table de successeurs du noeud 2.



Noeud d'identifiant P = 2	
Valeur de k	***Successeur du noeud 2***
0	***3***
1	***6***
2	***6***
3	***15***
4	***20***
...	...

FIGURE 1.1 – Table des successeurs d'un noeud dans Chord

Après avoir parcouru certaines tables de hachage distribuées qui sont utilisées dans les réseaux structurés, il est important de noter que pour bien assurer la localisation d'une ressource, tout protocole pair-à-pair doit mettre en oeuvre une bonne stratégie pour router (acheminer) les requêtes et les réponses entre les noeuds sources et les noeuds destinataires.

3.3 Méthodes de routage dans les réseaux P2P structurés

Une méthode de routage est jugée efficace si elle permet de localiser une ressource existante avec peu de messages mais aussi dans un délai raisonnable. Les méthodes de routage dans les réseaux pair-à-pair peuvent être regroupées essentiellement en trois familles [Bau2012] :

1. la méthode itérative
2. la méthode récursive
3. la méthode semi-récursive

méthode itérative : le routage avec la méthode itérative se fait de voisin en voisin, depuis le noeud source jusqu'au noeud destinataire. Chaque noeud recevant le message de la source, s'il n'est pas le noeud recherché, envoie au noeud source l'adresse de son voisin. La source contacte alors le voisin de son voisin qui fera pareil, ainsi de suite jusqu'à atteindre le destinataire. La figure 1.2 montre les échanges de messages dans le cas où le noeud A est à la recherche du noeud D.

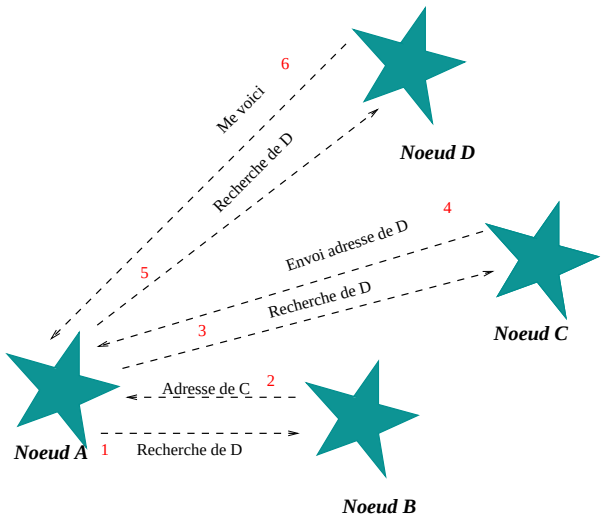


FIGURE 1.2 – Recherche d'un noeud par la méthode itérative

méthode récursive : avec la méthode récursive, la requête de la source est achiminée de voisin en voisin jusqu'au destinataire. La réponse emprunte le même chemin retour jusqu'à la source. Un tel scénario est illustré à la figure 1.3.

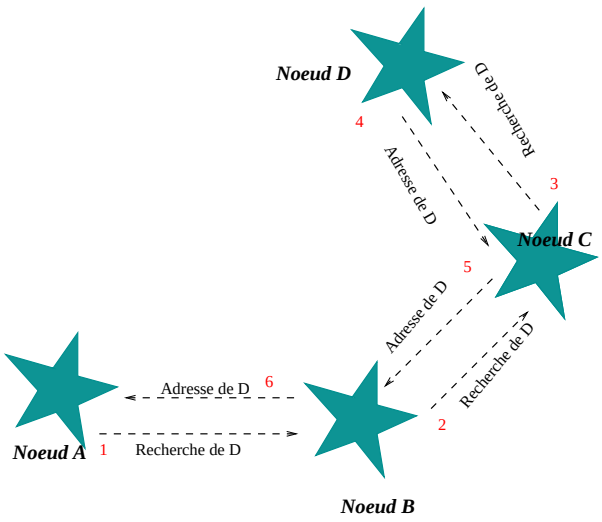


FIGURE 1.3 – Recherche d'un noeud par la méthode récursive

méthode semi-récursive : dans ce cas, le message du noeud source est acheminé de voisin en voisin comme avec la méthode récursive. Toutefois, la réponse est envoyée directement au noeud source. Elle n'emprunte pas le chemin inverse de la requête. Un exemple d'une recherche par la méthode semi-récursive est donnée à la figure 1.4

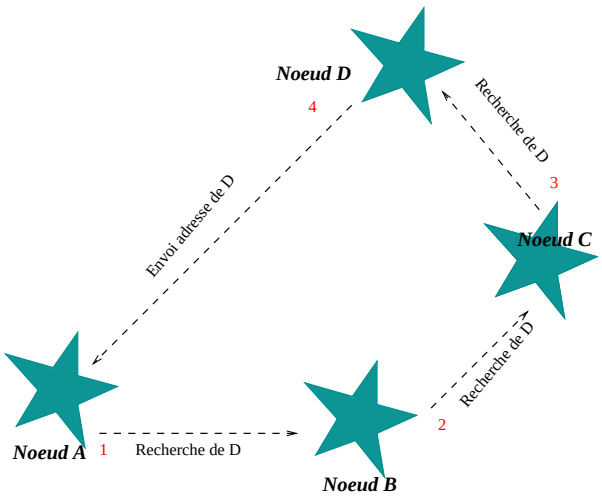


FIGURE 1.4 – Recherche d'un noeud par la méthode semi-récursive

La méthode semi-récursive ne rencontre pas de problèmes même lorsqu'un des deux noeuds communiquants (la source ou le destinataire) se trouve derrière un *NAT* contrairement aux méthodes itérative et récursive [Bry2008]. De plus, elle génère moins de messages que les deux autres. D'après D. Bryan et *al.* [Bry2008], dans un réseau de n noeuds, la méthode semi-récursive génère par noeud un nombre de messages égal à $(n-1)$ pour la localisation. Quant aux méthodes itérative et récursive, elles génèrent chacune par noeud $2 \times (n-1)$ messages.

Les protocoles pair-à-pair étaient à l'origine mis en place pour partager des données (textes, musique, etc.). Cependant, vers la fin de l'année 2003, Skype a révolu-

tionné le monde du pair-à-pair en y permettant les communications téléphoniques [Fessa2006]. Pour ce faire, il s'est appuyé sur le protocole d'initiation de session (*SIP*) pour mieux gérer l'aspect temps réel désormais introduit par la téléphonie.

4 Le protocole SIP

Le protocole d'initiation de session (*Session Initiation Protocol - SIP*) est un protocole de signalisation appartenant à la couche application du modèle OSI. Son rôle est d'ouvrir, de modifier et de libérer des sessions entre un ou plusieurs utilisateurs qui peuvent communiquer en mode point-à-point [6].
A l'origine, le protocole SIP a été spécifié par le groupe de travail *MMUSIC* (*Multiparty Multimedia SessIon Control*) de l'IETF (*Internet Engineering Task Force*) dans la RFC 2543 en mars 1999. Il est bâti sur le modèle Client/Serveur. Pour assurer sa mission de signalisation, il s'appuie sur d'autres protocoles comme :

- *TCP* ou *UDP* : pour le transport des messages de signalisation.
- *SDP* : pour la description des sessions.
- *RTP* ou *RTCP* : pour la propagation des flux d'information sur le réseau.
- *RSVP* : pour la réservation de ressources réseaux sur IP avec une bonne qualité de service.
- *SAP* : pour préciser si les sessions mutimedia ouvertes le sont en *multicast*.

La pile des couches d'utilisation de ces protocoles est illustrée à la figure 1.5

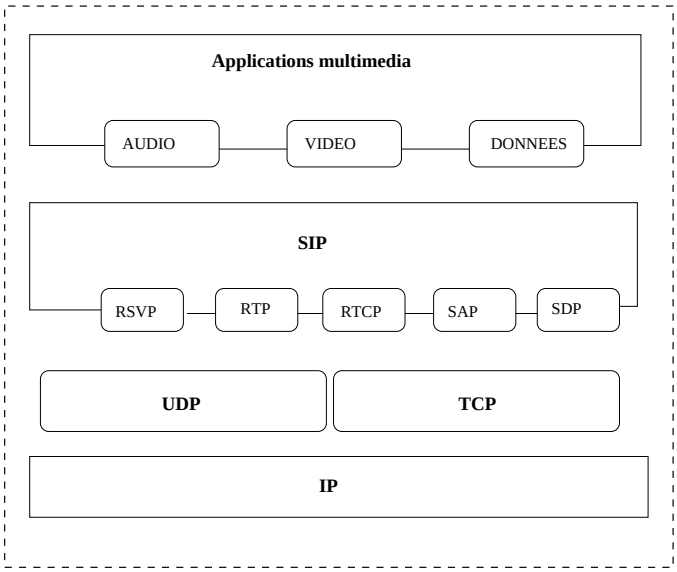


FIGURE 1.5 – Pile des couches du protocole SIP

Le protocole SIP se compose essentiellement de deux éléments : les applications clientes et les applications serveurs [Sim2005].

6. point-à-point : communication directe entre deux utilisateurs sans passer par une passerelle

4.1 Les applications clientes

Une application cliente encore appelée *user agent* est une application au sein d'un composant *SIP* final. Cela peut être, par exemple, un terminal de téléphonie ou de visioconférence sur IP, un serveur audio ou vidéo ou encore une passerelle vers un autre protocole. Les applications clientes sont constituées d'une partie cliente appelée *user agent client* (qui envoie les requêtes *SIP*) et d'une partie serveur nommée *user agent server* (qui reçoit les requêtes et y répond).

4.2 Les applications serveurs

Il y a quatre applications serveurs dans le protocole *SIP*.

Serveur d'enregistrement ou *registrar server* : c'est un serveur qui gère les requêtes *REGISTER* envoyées par les applications clientes pour signaler leur emplacement courant. Ces requêtes contiennent les adresses IP et SIP de l'utilisateur. Le serveur d'enregistrement détient une base de données dans laquelle sont stockés les couples (adresse IP, adresse *SIP*) des utilisateurs enregistrés. Une adresse *SIP* est similaire à une adresse email : *"sip :utilisateur@domaine"*. La figure 1.6 montre l'exemple d'enregistrement d'un utilisateur Fatou dont l'adresse IP est 192.168.10.24, et d'adresse email "fatou@ucad.edu.sn". A cette adresse email est associée une adresse sip "sip : fatou@ucad.edu.sn".

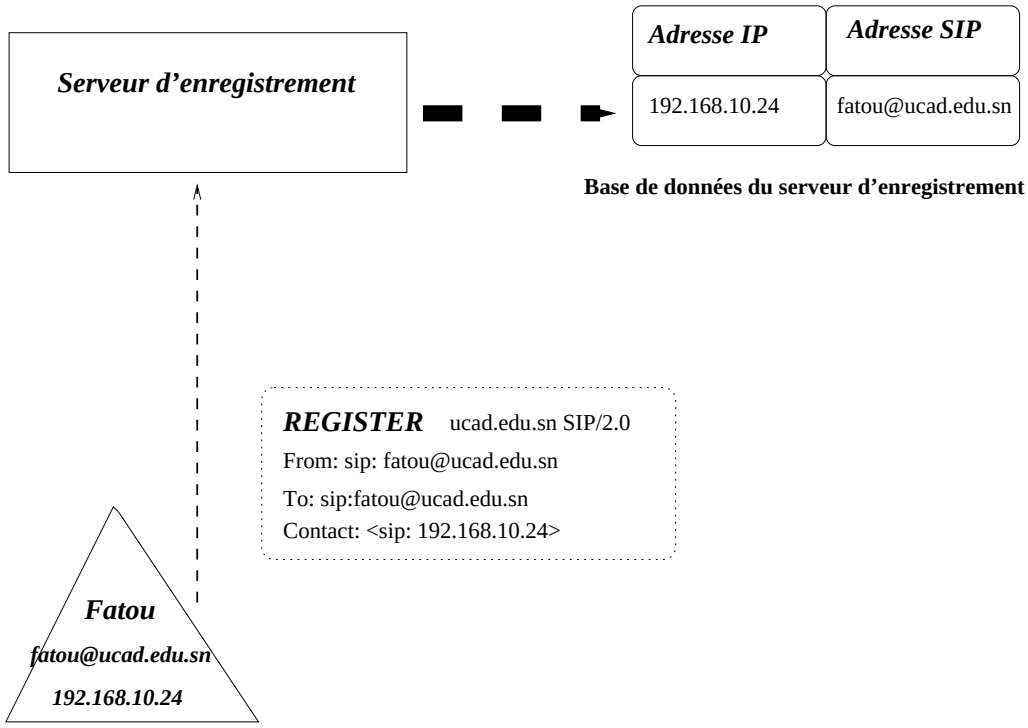


FIGURE 1.6 – Enregistrement d'un utilisateur au serveur SIP

Serveur relais mandataire (*proxy server*) : il représente un équipement agissant à la fois comme un *user agent client* et un *user agent server*. Il est chargé du

routage d'une session *SIP*. Lorsqu'il réceptionne une requête, il la transmet à un autre relais mandataire, sur la route ou directement à l'agent équipement concerné, si ce dernier est en liaison directe avec lui. Un agent équipement est un terminal *SIP* (téléphone, ordinateur, *softphone*, etc.) qui initie et reçoit des requêtes. La figure 1.7 illustre les étapes qui se font depuis l'invitation à une session jusqu'à l'établissement de la communication avec le protocole *SIP*.

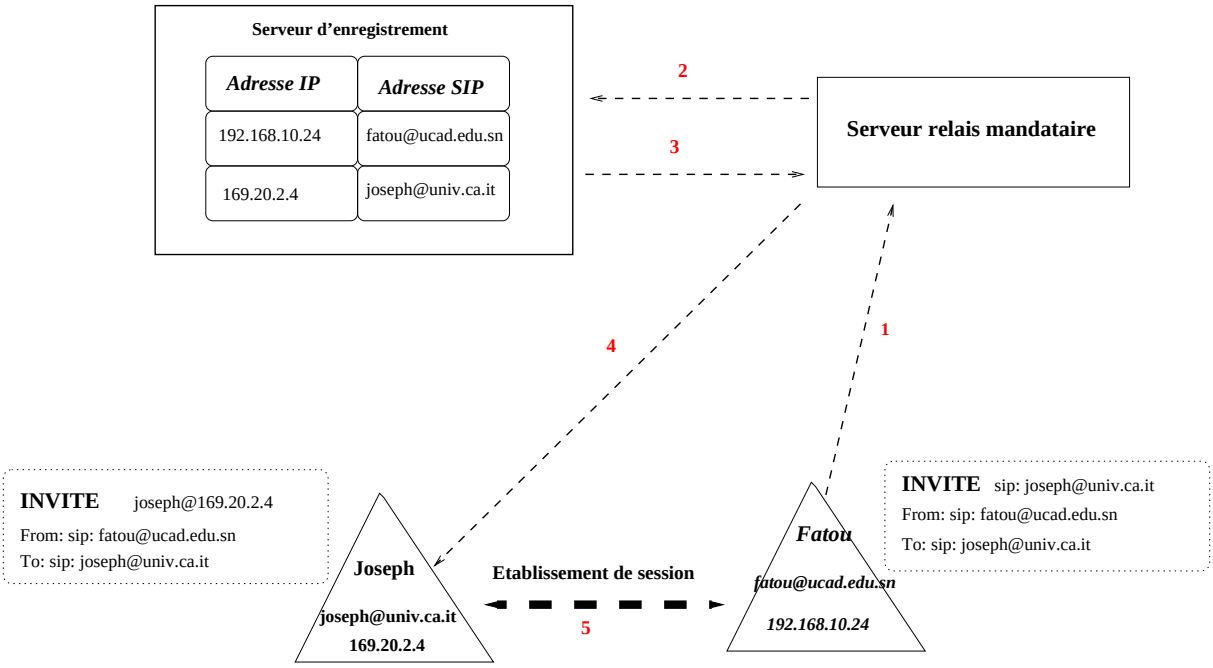


FIGURE 1.7 – Invitation à une session de communication via SIP

L'utilisateur Fatou connaît l'adresse *SIP* de Joseph (ou son pseudonyme) et souhaite entrer en communication avec lui. Elle lance une invitation avec l'adresse *SIP* de Joseph (1) vers le serveur relais mandataire qui envoie le message vers le serveur d'enregistrement (2) à la recherche de l'adresse IP correspondant à l'adresse *SIP* de Joseph. Le serveur d'enregistrement renvoie l'adresse IP de Joseph au serveur relais mandataire (3) qui peut désormais envoyer la requête vers le destinataire (4) et la session est établie (5).

Serveur de localisation (*location server*) : il aide à la localisation d'un *user agent server* en fournissant la position actuelle de ce dernier. Il renvoie à ses contacteurs (serveur d'enregistrement ou relais mandataire) une ou des adresses alternatives où le destinataire recherché pourrait être contacté.

Serveur de redirection (*redirect server*) : il s'appuie sur le serveur de localisation pour rediriger les appels vers de nouvelles adresses où l'utilisateur appelé se trouverait.

L'établissement d'une communication *SIP* entre deux utilisateurs s'accompagne d'échanges de plusieurs types de messages depuis l'invitation de l'appelant jusqu'à l'acceptation de l'appelé.

4.3 Types de messages SIP

Plusieurs types de messages sont échangés lors de l'établissement d'une session *SIP*. Il s'agit entre autres des messages :

- *INVITE* : est le message lancé par l'appelant pour demander l'ouverture d'une session ;
- *100 TRYING* : est un message envoyé par un pair intermédiaire pour signifier à l'appelant qu'il a acheminé le message ;
- *180 RINGING* : est un message envoyé par le destinataire de l'appel pour montrer qu'il a reçu l'invitation ;
- *200 OK* : est un message envoyé par le destinataire de l'appel pour préciser qu'il accepte l'invitation ;
- *ACK* : il est envoyé par l'appelant, après avoir reçu le message *200 OK*, pour confirmer la demande de communication ;
- *CANCEL* : est un message envoyé par l'appelant pour annuler un appel en cours ;
- *BYE* : est un message pouvant être envoyé par n'importe qui des deux communicants pour signifier qu'il souhaite mettre fin à la communication.

Ces messages sont illustrés à la figure 1.8

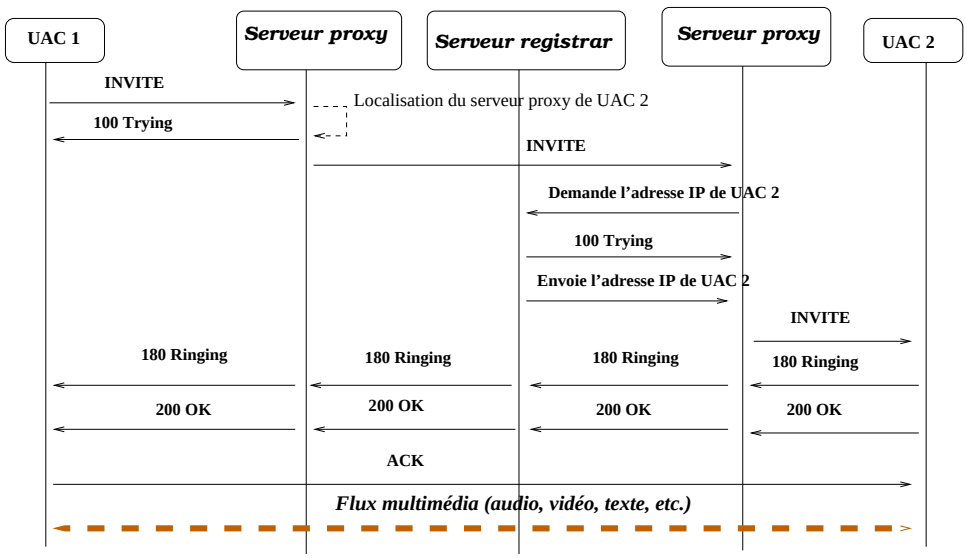


FIGURE 1.8 – Etablissement de session avec le protocole SIP

Sur la figure 1.8, le *user agent client 1 (UAC 1)* souhaite entrer en communication avec *UAC 2*. Il envoie un message *INVITE* à son serveur relais mandataire (serveur *proxy*) avec l'adresse *SIP* du *UAC 2*. Le serveur *proxy* de *UAC 1* trouve celui de *UAC 2* grâce au domaine dans l'adresse *SIP* de *UAC 2*. IL lui transfère le message *INVITE*. Le serveur *proxy* de *UAC 2* demande à son serveur d'enregistrement s'il détient dans sa base de données une adresse IP correspondant à l'adresse *SIP* de *UAC 2*. Le serveur d'enregistrement lui envoie l'adresse IP et le serveur *proxy* achemine le message *INVITE*. Dès lors, *UAC 2* peut répondre à *UAC 1*, d'abord par un message *100 Trying*, puis *180 Ringing* et ensuite *200 OK*. *UAC 1* envoie un

message *ACK* de confirmation et la session est établie entre *UAC 1* et *UAC 2*.

Le protocole SIP fonctionne avec une architecture centralisée car tous ces serveurs sont souvent localisés au sein d'une même entité, ce qui constitue un réel problème pour les éventuels pannes de serveur. Pour éviter un tel scénario, un groupe de l'IETF s'est donné comme charge de fournir un modèle standard de l'architecture *SIP* décentralisée. Il s'appuie sur la technologie pair-à-pair en distribuant le rôle des serveurs *SIP* aux pairs : c'est la naissance des réseaux *P2PSIP*.

5 Les réseaux P2PSIP

P2PSIP est le nom du Groupe de Travail (*Working Group*) sur l'*Internet Engineering for Task Force (IETF/WG)* en charge de fournir un modèle standard de l'architecture *SIP* décentralisée. Il s'appuie sur la technologie *peer to peer* dans le but d'offrir un service *SIP* robuste et distribué aux utilisateurs. Contrairement au *SIP* qui travaille avec le modèle client serveur classique, le *P2PSIP* délègue l'ensemble des fonctionnalités du protocole *SIP* aux pairs (téléphones, *softphones*, etc. membre du réseau *P2P*). Le protocole *P2PSIP* produit par l'*IETF/WG* est le *RESource Location And Discovery (RELOAD)*. C'est un protocole de signalisation *peer to peer* pour être utilisé dans l'internet. Il fournit un service de réseau de recouvrement générique et auto-organisé, permettant aux nœuds de router les messages vers d'autres nœuds, de stocker et récupérer des données [RFC6940].

Dans le *P2PSIP*, les noeuds participant au réseau peuvent être classifiés en deux types : les *peers* (pairs) et les clients. Les clients *P2PSIP* peuvent uniquement récupérer et annoncer des informations sur le réseau *P2P* créé par les *peers P2PSIP*. Les *peers*, en plus de ces fonctionnalités, travaillent ensemble pour maintenir la topologie du réseau, router les messages et stocker des données pour le réseau.

5.1 Les concepts du P2PSIP

Il y a des terminologies utilisées dans un réseau *P2PSIP* et qui définissent son fonctionnement [RFC7890].

P2PSIP : C'est un ensemble de protocoles qui étendent le protocole *SIP* en utilisant les techniques du *P2P* pour traduire et trouver les destinations des requêtes *SIP*.
P2PSIP overlay : C'est un réseau *P2P* dans lequel les noeuds participants remplissent les fonctionnalités du protocole *SIP* à savoir l'enregistrement, le routage, la localisation, etc. Dans un réseau *P2PSIP* il n'y a pas de serveurs centraux comme les serveurs *Proxy*, *Registrar* et de Localisation dans le *SIP*. Tout est décentralisé au niveau des pairs du réseau.
P2PSIP peer : C'est un noeud participant au *P2PSIP overlay*. Il assure entre autres services, le routage et le stockage d'informations. Un P2PSIP peer peut se retrouver derrière un *NAT* (*Network Adress Translation*) et fonctionner correctement. Il peut quitter ou rejoindre à tout moment le *P2PSIP overlay*.
P2PSIP client : C'est un noeud participant au réseau *P2PSIP* sans fournir des fonctions de stockage, ni de routage, ni de récupération d'informations sur le

réseau. Pour interagir avec le résau, il passe par un *P2PSIP peer*.

P2PSIP peer Enrollment : L'*Enrollment* est le processus qu'un *P2PSIP peer* suit pour obtenir un identificateur et des pouvoirs sur le résau *P2PSIP overlay*. Il est exécuté lorqu'un *P2PSIP peer* perd son identificateur ou ses pouvoirs.

Peer protocol : C'est le protocole utilisé entre les *P2PSIP peers* pour partager des informations et organiser le réseau *P2PSIP Overlay*. Il est implémenté par le protocole *RELOAD* [RFC6940].

Client protocol : C'est le protocole utilisé entre les *P2PSIP clients* et les *P2PSIP peers*. Il est implémenté par le protocole *RELOAD*.

Peer protocol connection / P2PSIP client protocol connection : Ce sont les couches de transport à travers lesquelles les messages du protocole *RELOAD* transitent. Il s'agit entre autres de, *Transport Layer Security (TLS)*, *Datagram Transport Layer Security (DTLS)*, *Transmission Control Protocol (TCP)*, *User Datagram Protocol (UDP)*.

Neighbors : L'ensemble des *P2PSIP peers* qu'un *P2PSIP peer* ou un *P2PSIP client* connaît directement et qu'il peut atteindre sans autres recherches.

Joining peer : Un noeud qui cherche à devenir un *P2PSIP peer*, c'est-à-dire, qui cherche à intégrer le *P2PSIP overlay*.

Bootstrap peer : Un *P2PSIP peer* du réseau, qui est le premier point de contact d'un *Joining peer*.

Bootstrap server : Un noeud du réseau, utilisé par les *Joining peers* pour localiser un *Bootstrap peer*. Un *Bootstrap server* peut agir comme un serveur relais mandataire pour les messages entre le *Joining peer* et le *Bootstrap peer*.

Tous ces éléments sont utilisés dans un réseau *P2PSIP overlay* respectant une architecture donnée.

5.2 Architectures P2PSIP

Il y a essentiellememt trois modèles d'architectures *P2PSIP* selon le type de noeuds participant à la distribution (routage, localisation, etc.) [Waro2012].

Modèle centralisé : dans ces types de réseaux il y a des serveurs centraux qui facilitent l'interaction entre les pairs du réseau. Ces serveurs identifient les noeuds du réseau et sont chargés de la recherche des pairs en cas de besoin. Ce type de modèle ne nécessite pas de modifications au niveau des clients.

Modèle plat ou purement décentralisé : dans ce modèle, tous les noeuds sont égaux en terme d'exécution des tâches (ils sont clients et serveurs). Ils sont au même niveau et participent tous à la distribution.

Modèle hiérarchique ou partiellement décentralisé : ce modèle se base en réalité sur le fait que tous les noeuds n'ont pas tous les mêmes performances (bande passante, CPU, connectivitée,...). Les noeuds avec de bonnes performances représentent les *P2PSIP peers* et les autres sont les *P2PSIP clients*. La distribution est assurée par les *P2PSIP peers*. Ce modèle est celui choisi par le *P2PSIP Working Group*.

Tous ces modèles sont illustrés à la figure 1.9.

Il existe, pour ces architectures, deux modèles standardisés par le *P2PSIP Working Group* : il s'agit de *P2P over SIP* et de *SIP-using-P2P*.

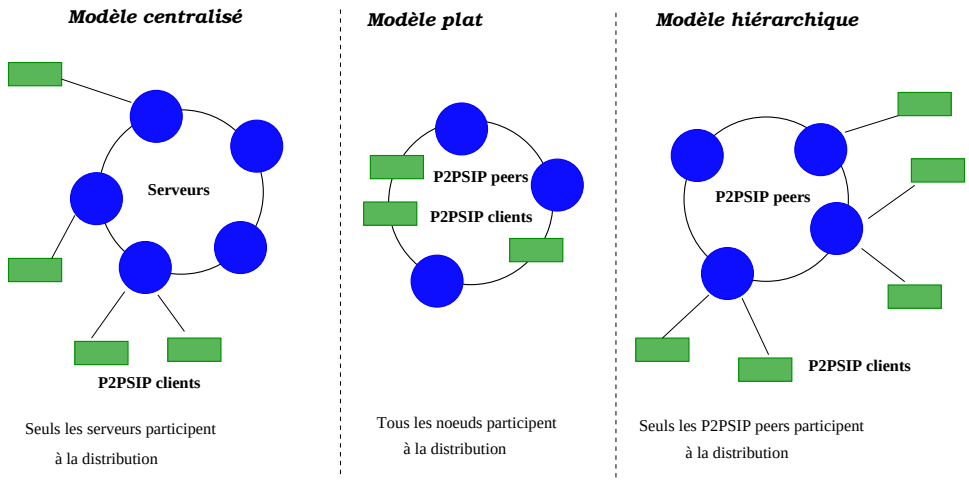


FIGURE 1.9 – Modèles de distribution dans le P2PSIP

5.2.1 P2P over SIP

Dans ce modèle, en plus de l'enregistrement des utilisateurs, de la recherche de ressources, de l'établissement des sessions, les messages *SIP* sont également utilsés pour la maintenance du réseau *P2P*. Ce modèle n'est pas interopérable avec n'importe qu'elle application *P2P*.

Lorsqu'un noeud rejoint le réseau, il combine simultanément l'envoi d'un message *REGISTER* pour s'enregistrer auprès d'un *peer*, et l'envoi d'un message à un *peer* connu (ou en *multicast*) pour chercher un identificateur d'insertion au sein du *P2PSIP overlay* [Sing2005].

Lorsqu'un *P2PSIP peer* quitte le réseau *P2PSIP overlay*, il utilise un message *REGISTER* pour transférer, les informations de tous les *P2PSIP clients* dont il est responsable, à un autre *P2PSIP peer*. Quand un *P2PSIP client* quitte le *P2PSIP overlay*, il informe son *P2PSIP peer* responsable par un message *unREGISTER*. C'est au *P2PSIP peer* responsable d'informer les autres.

Lors d'un départ brutal d'un *P2PSIP client*, son *P2PSIP peer* responsable constate qu'il ne reçoit plus de messages de mise à jour de la part du *P2PSIP client*. Il le supprime et informe les autres *P2PSIP peers*. De la même manière, si un *P2PSIP peer* quitte le réseau, ses voisins (*Neighbors*) remarquent qu'ils ne reçoivent plus de messages de mise à jour. Dans ce cas, ils considèrent que ce dernier a quitté le réseau et tentent de récupérer ses *P2PSIP clients*.

Les protocoles comme *Dsip*, *SoSIMPLE*, *MISE-P2PSIP (Middleware-Independent and SEcure P2PSIP)*, etc. implémentent le *p2p-over sip* [Elys2012].

5.2.2 SIP-using-P2P

L'architecture *SIP-using-P2P* remplace le service de localisation de *SIP* par un protocole *P2P* [Lekuo2007] (autrement dit, il n'y a pas de serveur de localisation classique). En effet la localisation des ressources se fait en utilisant les techniques purement *P2P*.

De même, dans *SIP-using-P2P*, ce sont les messages *P2P* qui sont utilisés pour

maintenir le réseau surtout en cas d'ajout ou de retrait de noeuds.
Le protocole *RELOAD* proposé par le *P2PSIP Working Group* est implémenté avec le *SIP-using-P2P*. L'envoi des messages *SIP* par exemple se fait en utilisant l'identificateur du noeud et non l'adresse IP, car avec *RELOAD*, la phase d'enregistrement d'un *P2PSIP peer* se fait par le *mapping* entre l'adresse d'enregistrement (*Address of Record - AoR*) et l'identificateur du noeud. En d'autres termes, pour s'enregistrer dans un réseau *P2PSIP* utilisant *sip-using-p2p*, il suffit de suivre la procédure d'ajout de noeud du réseau *P2P* considéré (Chord, Pastry, etc.).

Les architectures comme *JXTA*, *P2PNS (Secure Distributed Name Service for P2PSIP)*, implémentent le *SIP-using-P2P*. L'architecture *SIP-using P2P* présente deux couches :

- Une couche *SIP* pour l'enregistrement des utilisateurs, la recherche de ressources et l'établissement de session
- Une couche *P2P* pour le maintien d'un réseau *P2P*.

Les deux couches peuvent être implémentées sur un même noeud ou sur des noeuds différents.

Sur le tableau 1.1 sont résumées les procédures de fonctionnement du *P2P over SIP* et du *SIP-using-P2P* vis-à-vis des services *SIP* que sont l'enregistrement, la localisation, etc.

	SIP-using-P2P	**P2P over SIP**
Localisation	messages P2P	messages SIP (les peers jouent le rôle de serveurs de localisation)
Enregistrement	messages P2P pour les *P2PSIP peers* messages SIP *REGISTER* pour les *P2PSIP clients*	messages SIP et P2P à la fois (*REGISTER*, *multicast*)
Routage	messages P2P	messages SIP (les *P2PSIP peers* jouent le rôle de serveurs *Proxy* et *Registrar*)

TABLE 1.1 – SIP-using-P2P contre P2P over SIP

Remarque 5.1 *P2PSIP est un réseau peer to peer pour les communications en temps réel [RFC7890]. Certains P2PSIP peers ou P2PSIP clients sont couplés à des entités SIP. Par exemple, il est possible de coupler un P2PSIP peer avec un user agent server, et un P2PSIP client avec un user agent client, ainsi de suite.*

6 Réseaux P2PSIP hiérarchiques

Les réseaux pair-à-pair *SIP* hiérarchiques sont des réseaux *P2PSIP* dans lesquels les noeuds de grandes capacités nommés super noeuds (SN) sont dans un même niveau, et les autres de faibles capacités appelés noeuds ordinaires sont dans d'autres niveaux plus bas. Les super noeuds jouent entre autres, le rôle des serveurs *SIP* et

assurent les tâches les plus complexes. Quant aux noeuds ordinaires (NO), ils représentent les applications clientes. Il existe différents types d'architectures *P2PSIP* hiérarchiques :

- architectures à un domaine de recouvrement : dans ce cas les super noeuds sont sur un même domaine. Ces architectures se présentent sous deux configurations. Dans le premier cas, les noeuds ordinaires sont rattachés aux super noeuds [Zoels2006, Bra2012] formant ainsi une architecture à un seul domaine avec une seule couche. Dans le second cas, les noeuds ordinaires forment des couches hiérarchiques rattachées au domaine [Lekuo2007]. Ceci donne l'architecture à un seul domaine avec plusieurs couches.
- architectures à plusieurs domaines de recouvrement : dans ce cas, les super noeuds sont sur un même domaine et les noeuds ordinaires sont organisés en sous-domaines rattachés aux super noeuds [Shi2007, Isla2011, Yelmo2009]. Il est possible d'utiliser une table de hachage distribuée par domaine comme c'est le cas dans [Liu2012, Stoi2001].

D'après Xianghan Zheng et al. [Zheng2010], un réseau de recouvrement est un réseau virtuel de noeuds et de liens logiques construit sur un réseau physique existant et permettant de mettre en oeuvre un service réseau non disponible sur le réseau physique.

6.1 Architectures P2PSIP hiérarchiques à un domaine de recouvrement

Les architectures *peer to peer SIP* hiérarchiques à un seul domaine sont de deux types : un seul domaine avec une couche et un seul domaine avec plusieurs couches [LeKuo2007, Zoels2006].

Architecture hiérarchique à seul domaine avec une seule couche : il s'agit de l'architecture par défaut pour les réseaux pair-à-pair *SIP* hiérarchiques. Les super noeuds sont au même niveau et chaque noeud ordinaire se rattache à un super noeud. Les noeuds ordinaires ne font pas de routage et passent par leur super noeud responsable en cas de tentative de localisation d'un autre noeud. Toutefois, après avoir localisé le destinataire d'une requête, les communications entre les deux se font directement sans passer par le super noeud. Chaque super noeud connaît les noeuds ordinaires qui lui sont rattachés. Cette architecture est illustrée à la Figure 1.10.

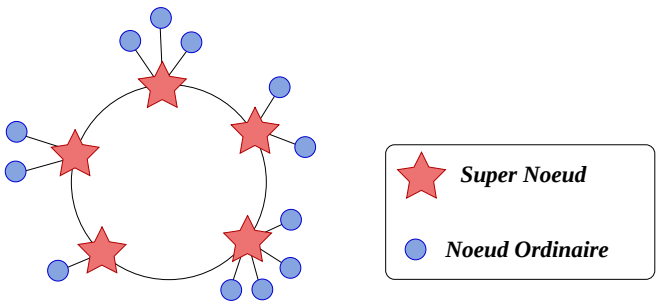


FIGURE 1.10 – Architecture hiérarchique à un domaine de recouvrement et une seule couche

Architecture hiérarchique à seul domaine avec plusieurs couches : ces types d'architectures sont constituées de plusieurs couches. La plus haute contient les super noeuds et les autres noeuds, en fonction de leurs capacités, sont répartis à travers les autres couches [LeKuo2007]. Les noeuds de plus faibles capacités occupent la couche la plus basse. Différentes tables de hachage distribuées (comme Chord, Pastry, CAN, etc.) peuvent être utilisées dans les différentes couches. Un exemple de ce type d'architecture est illustré par la Figure 1.11.

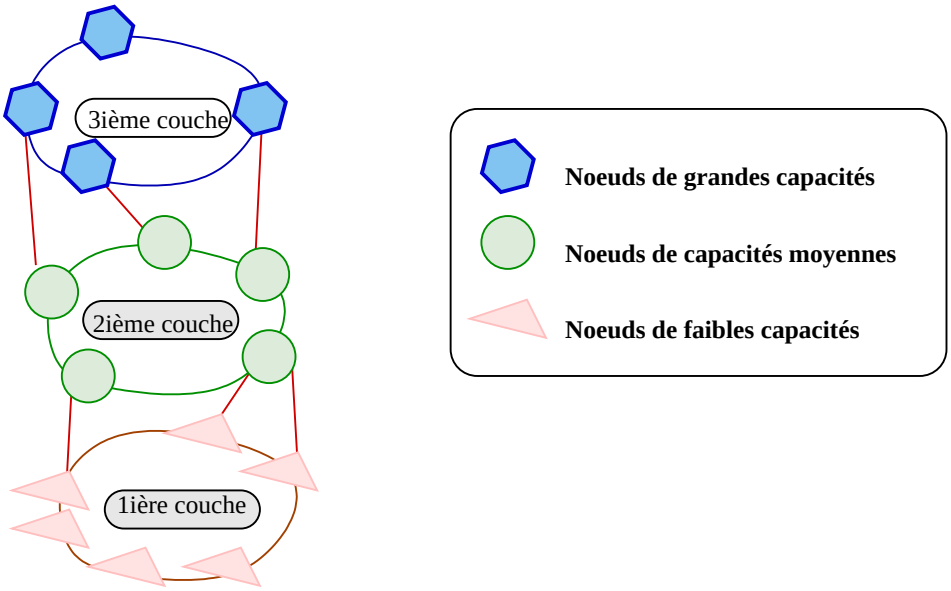


FIGURE 1.11 – Architecture hiérarchique à un domaine de recouvrement et plusieurs couches

Lorsqu'un noeud arrive dans le système, il est d'abord inséré au niveau le plus bas. Si ses capacités sont suffisamment grandes, alors il "monte" au niveau supérieur ainsi de suite jusqu'à atteindre son vrai niveau (selon ses capacités).
La recherche d'un noeud se fait dans un premier temps sur le niveau du noeud source[7]. Lorsque le noeud recherché ne s'y trouve pas, la requête est envoyée au niveau suivant (niveau inférieur ou supérieur selon la position du noeud source).

6.2 Architectures P2PSIP hiérarchiques à plusieurs domaines de recouvrement

Les architectures *P2PSIP* hiérarchiques à plusieurs domaines sont constituées de sous domaines rattachés à un domaine principal. Selon la table de hachage utilisée, elles sont classées en deux catégories. D'un côté se trouvent celles fonctionnant avec une seule table de hachage distribuée et de l'autre celles utilisant à la fois plusieurs tables de hachage distribuées.

Architectures P2PSIP hiérarchiques à plusieurs domaines avec une seule table de hachage distribuée : Dans ce cas, le domaine principal et les sous domaines fonctionnent avec la même table de hachage distribuée. La figure 1.12 montre

7. noeud source : noeud demandeur, qui cherche à localiser un autre

un exemple où Chord est utilisée comme table de hachage distribuée. Lorsqu'un

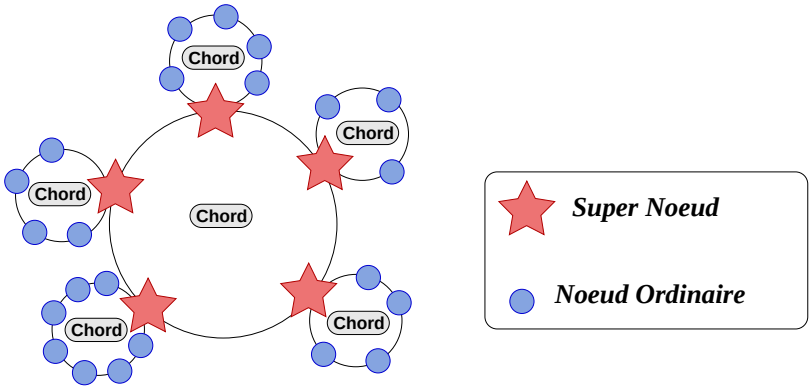


FIGURE 1.12 – Architecture hiérarchique à plusieurs domaines et une seule table de hachage distribuée

noeud tente de localiser un autre, la recherche est d'abord faite au sein de son sous domaine. C'est seulement en cas de non succès que la requête est envoyée au domaine principal via le super noeud.

Architectures P2PSIP hiérarchiques à plusieurs domaines avec plusieurs tables de hachage distribuées : Dans ces types d'architectures, la table de hachage distribuée qui est mise en oeuvre dans le domaine principal n'est pas nécessairement celle qu'utilisent les sous domaines. Un bel exemple d'une telle architecture se trouve dans [Isla2011]. Dans leur travaux Md. Saqiful et al. considèrent un domaine principal avec Bamboo comme table de hachage distribuée et des sous domaines fonctionnant respectivement avec Chord, Tapestry, CAN, etc.

7 Conclusion

Partant des systèmes centralisés Client/Serveur, nous avons montré dans ce chapitre la genèse des réseaux pair-à-pair qui, de par leur fonctionnement sont regroupés en réseaux structurés et non structurés. Les nombreux avantages des réseaux structurés (recherche garantie, moins pollueurs de la bande passante, etc.) par rapport aux non structurés ont guidé notre choix vers les premiers.

Nous avons présenté le protocole *SIP* grâce auquel la téléphonie a été introduite dans les réseaux *P2P* pour donner naissance au *P2PSIP*. Nous avons également montré le fonctionnement des modèles d'architecture du *P2PSIP*, à savoir le *SIP-using-P2P* et le *P2P over SIP*. Avec la mobilité des utilisateurs, de nouveaux défis sont apparus. En effet, les réseaux *P2PSIP* sont caractérisés par des noeuds qui arrivent dans le réseau et quittent à tout moment. Ce qui rend plus complexe le maintien des sessions.

C'est pourquoi, dans le chapitre suivant, nous aborderons les solutions de gestion de la mobilité dans les réseaux *peer to peer SIP* hiérarchiques afin de relever les problèmes y afférents et d'y apporter des solutions.

Chapitre 2

Solutions de gestion de la mobilité dans IPv6 et Réseaux pair-à-pair

1 Introduction

De nos jours, les communications mobiles occupent une bonne partie du trafic sur internet. Avec des utilisateurs de plus en plus mobiles, le maintien d'une session multimédia pose souvent de nombreuses difficultés. En effet, lors des déplacements, les utilisateurs peuvent passer d'une cellule à une autre ou même d'une technologie d'accès à une autre (c'est la mobilité physique). Ce qui provoque parfois des déconnexions, des pertes de paquets, des interruptions de la communication, etc. Les solutions de *mobile IP* ont été mises en place pour permettre à un noeud de se déplacer tout en restant connecté à internet. Elles sont essentiellement de deux natures : le *mobile IPv4 (MIPv4)* et le *mobile IPv6 (MIPv6)*. Les multiples avantages du *MIPv6* par rapport à *MIPv4*, qui sont entre autres le principe de l'optimisation de la route, la possibilité pour un noeud de se déplacer tout en conservant son adresse, etc. font que de plus en plus les solutions s'orientent vers IPv6.

Par ailleurs, l'utilisation d'applications pair-à-pair *SIP* sur des périphériques mobiles est devenue une réalité. Or, les systèmes *peer to peer SIP* sont caractérisés par des noeuds qui intègrent le réseau et le quittent à tout moment, entraînant ainsi un changement de la topologie. Les noeuds dont les voisins ont quitté le réseau doivent se connecter à d'autres points d'attachement (c'est la mobilité logique) pour rester dans le réseau. Puisque que les mouvements des utilisateurs influent sur les nombreux *churns*[1] dont souffrent les réseaux *peer to peer*, et par conséquent affectent négativement leurs performances [Yel2009], pour mettre en place une solution de gestion de la mobilité dans IPv6 avec un réseau *peer to peer SIP*, il est nécessaire de prendre en compte ces deux types de mobilité (mobilité physique et mobilité logique). Cependant, sachant que les réseaux *P2PSIP* hiérarchiques sont une optimisation des réseaux *P2PSIP* [Yel2009], leur combinaison avec la mobilité dans IPv6 permet d'avoir une solution efficace.

Dans ce chapitre, nous détaillons les solutions de gestion de la mobilité dans IPv6 à la section 2. La section 3 est consacrée aux réseaux pair-à-pair *SIP* hiérarchiques.

1. churn : mouvements occasionnés par les arrivées et les départs des noeuds dans le réseau

Dans la section 4, nous traitons la gestion de la mobilité dans un réseau pair-à-pair.

2 Solutions de gestion de la mobilité dans IPv6

Dans cette section, nous donnons des explications sur les communications mobiles de façon générale et ensuite nous détaillons les concepts de la mobilité dans IPv6.

2.1 Généralités

La mobilité est l'ensemble des techniques mises en place, dans un réseau composé de plusieurs entités ou plusieurs technologies d'accès, pour permettre d'établir ou de maintenir une communication lors du déplacement d'un mobile d'une zone à une autre ou d'une technologie à une autre [Thie2002]. Avec la multiplication du nombre d'abonnés des opérateurs de téléphonie, le volume des communications a nettement augmenté. Les cellules sont souvent saturées, ne disposant pas de canaux de communication libres. Des solutions à ce problème étaient soit de multiplier le nombre de canaux d'une cellule soit de permettre à des cellules voisines de s'emprunter mutuellement des fréquences selon la nécessité [Georg2011]. Ces solutions ont vite rencontré des limites. Pour la première, la multiplication du nombre de cellules consiste à diviser les cellules ce qui contribue à augmenter le nombre de *handover* et donc une gestion plus difficile. Quant à la deuxième, elle suppose l'élargissement de la bande des fréquences des cellules, ce qui pourrait (à force d'élargir) créer des interférences.

La téléphonie sur IP (*Telephony over IP - ToIP)* est une des solutions car permettant de désengorger dans une certaine mesure les réseaux cellulaires en offrant aux usagers la possibilité de s'appeler via le réseau internet. Les utilisateurs sont épargnés des nombreuses attentes souvent rencontrées dans les réseaux *GSM* par exemple dues à un manque de canaux libres. En raison de tous ces avantages et aussi des prix élevés souvent imposés par les opérateurs de téléphonie, les utilisateurs, dans un soucis de bénéficier d'un service de qualité à moindre coûts, s'orientent de plus en plus vers les communications IP.
Dès lors de nouveaux défis voient le jour. Puisque les utilisateurs sont souvent très mobiles, il faut leur donner la possibilité de communiquer via internet tout en leur accordant l'éventualité d'un déplacement durant la communication. C'est dans ce cadre que les solutions de mobilité sur IPv4 et sur IPv6 ont été mises en place. Avec la prolifération des périphériques de communication mobile et l'émergence des objets connectés les solutions basées sur IPv4 ont montré leurs limites en raison d'un manque d'adresses disponibles. De plus, dans IPv4, avec le routage triangulaire, la localisation d'un noeud prend souvent plus de temps à cause des éventuelles encapsulations lorsqu'il s'agit d'adresses non routables. A l'inverse, dans IPv6, une infinité d'adresses est disponible et le routage obéit au principe de l'optimisation de la route (*route optimization*). Ce principe consiste à éviter le routage triangulaire de IPv4 en envoyant directement les messages vers les destinataires. Il n'y a pas d'encapsulation nécessaire car toutes les adresses sont routables. Toutes ces multiples raisons expliquent l'engouement vers les solutions basées sur IPv6.

2.2 La mobilité IPv6

Le Protocole Internet version 6 (IPv6) offre plusieurs solutions de gestion de la mobilité. Selon leur mode de fonctionnement, elles peuvent être groupées en deux grandes familles [Kha2014] : les protocoles de gestion de la mobilité basés sur le mobile (*Host-Based Mobility Management Protocols - HBMMP*) et les protocoles de gestion de la mobilité basés sur le réseau (*Network-Based Mobility Management Protocols - NBMMP*) .

2.2.1 HBMMP : protocoles de gestion de mobilité basés sur le mobile

Avec ce protocole, la participation du noeud mobile est nécessaire durant le processus de *handover*. Son fonctionnement est régi par un certain nombre d'éléments : le *home agent (HA)*, le *foreign agent (FA)* , le *mobility anchor point (MAP)* et le noeud mobile (NM). Un *home agent* est un routeur d'un réseau dans lequel se trouve le noeud mobile et qui maintient des informations sur la localisation courante des périphériques mobiles [Sil2006]. Le *foreign agent* est un routeur dans le réseau visité par le noeud mobile. Quant au *MAP*, il est un composant placé entre le *HA* et le noeud mobile. Il stocke les informations du noeud mobile et agit en local comme un *HA*. Un noeud mobile est un noeud pouvant changer de point d'attachement d'un réseau à un autre, tout en restant joignable avec sa *home address* (c'est-à-dire l'adresse obtenue de son *HA*). *HBMMP* fonctionne suivant trois modes [Wei2015] : (*Mobile IPv6 - MIPv6*), Mobile IPv6 hiérarchique (*Hierarchical MIPv6 - HMIPv6*), rapide handover dans le mobile IPv6 (*Fast handover for Mobile IPv6 - FMIPv6*) .

Mobile IPv6 : dans ce cas le noeud mobile communique directement avec son *HA* sans intermédiaire [John2004]. La figure 2.1 montre le fonctionnement de *MIPv6* durant le déplacement d'un noeud mobile. Le noeud mobile maintient une session avec son noeud correspondant (NC)(0). Lorsque le noeud mobile se déplace dans un autre réseau, il envoie au routeur un message pour demander une nouvelle adresse (1). Le message est transféré au *FA* (2) qui se charge d'attribuer au noeud mobile une nouvelle adresse appelée *Care-of-Address (CoA)* (3) et (4). Le noeud mobile informe son *HA* afin que ce dernier mette à jour ses informations (5). Ensuite le *HA* répond par un accusé de réception (6). Dès lors, les paquets destinés au noeud mobile (7) sont interceptés par le *HA* grâce au protocole de résolution d'adresse *Address Resolution Protocol (ARP)*. Le *HA* les achemine ensuite au *FA* en utilisant la *care-of-address* (8) et la session se poursuit entre le noeud mobile et son correspondant (9).

Mobile IPv6 hiérarchique (*Hierarchical MIPv6 - HMIPv6)* : il permet de réduire les coûts de signalisation entre le noeud mobile et son *HA* [Sil2006]. Un *MAP* est placé entre les deux composants. Chaque noeud mobile obtient deux adresses : une adresse par rapport à sa position (région) géographique appelée *Regional Care-of-address* et une adresse locale à son réseau d'appartenance. Cette dernière, nommée *Local Care-of-address (LCoA)* représente ce qu'est la *CoA* dans le *MIPv6*. Un *MAP* peut couvrir plusieurs domaines comme le montre la figure 2.2 . Lors du déplacement du noeud mobile entre des domaines gérés par un même *MAP*, seule la *LCoA* change.

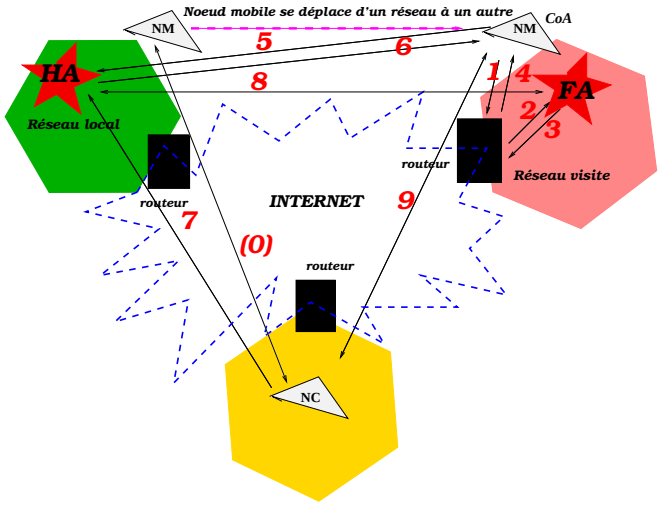


FIGURE 2.1 – Déplacement d'un NM de son réseau vers un autre avec le MIPv6

La *RCoA* ne change pas et les mises à jour à apporter n'atteignent pas le *HA*. Elles se limitent au niveau du *MAP*.

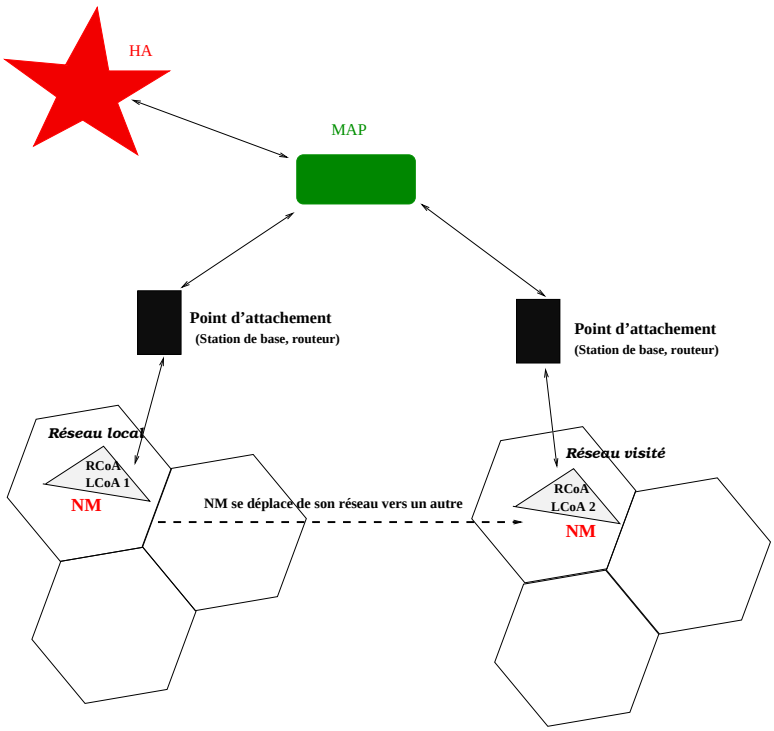


FIGURE 2.2 – Déplacement d'un NM de son réseau vers un autre avec le HMIPv6

Le noeud mobile (NM) possède une *RCoA* et une local *care-of-address LCoA* 1 dans son réseau local. Lorsqu'il entre dans le nouveau réseau (réseau visité), il obtient une nouvelle *local care-of-address LCoA* 2. Cependant, la *RCoA* est toujours la même.

Rapide handover dans MIPv6 (*Fast handover for MIPv6 - FMIPv6*) : *FMIPv6* a été mis en oeuvre pour réduire le délai de *handover* [Mor2003]. Le noeud

mobile obtient une adresse avant même de se déconnecter de son ancien *home agent*. *FMIPv6* fonctionne en mode prédictif ou réactif. Dans le mode prédictif, il y a une anticipation sur le *handover*. Le noeud mobile lance la recherche d'une nouvelle adresse (*CoA*) dès qu'il constate que le *handover* sera bientôt nécessaire. Pour ce faire, lorsque la force du signal atteint une certaine valeur seuil, il lance la recherche de la nouvelle adresse en envoyant un message au *Foreign Agent*. Dans le mode réactif par contre, il est nécessaire que le *handover* soit déclenché avant que le noeud mobile ne commence à chercher une autre adresse. Cependant, contrairement au mode prédictif pour lequel le message est envoyé au *Foreign Agent*, dans le mode réactif, le noeud mobile envoie le message à son *Home Agent* pour préciser qu'il est entré dans la phase de *handover*.

2.2.2 NBMMP : protocoles de gestion de mobilité basés sur le réseau

Dans le *NBMMP*, les éléments constitutifs sont les points locaux d'ancrage de la mobilité ou *Local Mobility Anchors (LMA)* , les passerelles d'accès mobile (*Mobile Access Gateways - MAG)* et les noeuds mobiles . Les *LMAs* assurent le rôle des *home agent* à l'endroit des noeuds mobiles [Soto2010]. Alors que les *MAGs* sont des détecteurs de mouvement. Ils effectuent la signalisation de mobilité au nom des nœuds mobiles qui leur sont attachés. Le *NBMMP* contient deux grands protocoles standardisés par le groupe de travail *Internet Engineering Task Force (IETF)*. Il s'agit du *Proxy Mobile IPv6 (PMIPv6)* et du rapide *proxy mobile IPv6* (*Fast Proxy Mobile IPv6 - FPMIPv6*) [Song2015].

Proxy Mobile IPv6 :

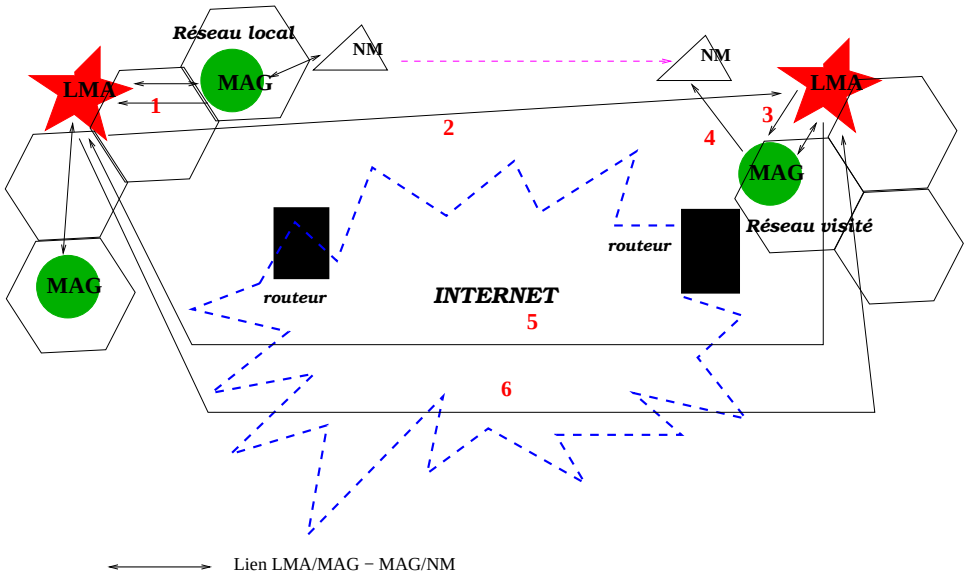


FIGURE 2.3 – Déplacement d'un NM de son réseau vers un autre avec le PMIPv6

Dans ce protocole, le noeud mobile n'intervient pas durant la phase de *handover*. Tout le processus est assuré par les *LMAs* et les *MAGs*. Il est possible de trouver plusieurs *LMAs* dans un même domaine basé sur le *Proxy Mobile IPv6*. Dans ce cas, chacun d'eux gère un groupe de *MAGs* et de noeuds mobiles [Leu2008]. Les noeuds

mobiles se rattachent aux *MAGs* qui à leur tour se rattachent aux *LMAs* (figure 2.3). Lorsqu'un noeud mobile se déplace, le mouvement est détecté par son *MAG* qui informe le *LMA* de son domaine (1). Ce dernier transfère le message au *LMA* du réseau à visiter pour demander une adresse IP pour le noeud mobile en déplacement. Dans le cadre des *NBMMP*, cette adresse est appelée (*Proxy CoA - PCoA*) (2). Des algorithmes basés sur la direction du noeud mobile ou sur la force du signal reçu sont utilisés pour déterminer le *LMA* le mieux adapté à recevoir le noeud mobile. C'est donc vers cet *LMA* que le message est envoyé [Fu2014]. A la réception du message, le *LMA* du domaine visité envoie une nouvelle adresse (*PCoA*) au *MAG* du noeud mobile (3). Cette adresse est assignée au noeud mobile (4). A partir de ce moment le *LMA* du domaine visité envoie un message de mise à jour appelé *Proxy Binding Update (PBA)* à l'ancien *LMA* du noeud mobile (5) qui répondra par un message d'accusé de réception (6) appelé *Proxy Binding Acknowledge (PBA)*.

Rapide Proxy Mobile IPv6 (*Fast Proxy Mobile IPv6 - FPMIPv6*) : Le rapide *proxy mobile IPv6* a été établi pour améliorer le *proxy mobile IPv6* [Zoh2014]. Il réduit le délai de *handover*. Il a deux modes de fonctionnement comme c'est le cas avec le *FMIPv6* : les modes prédictif et réactif. Dans le prédictif, lorsque le signal du noeud mobile commence à s'affaiblir, son *MAG* lance une recherche d'une *proxy CoA* avant même que le *handover* ne commence réellement. Cette anticipation permet au noeud mobile de disposer d'une adresse plus tôt (avant le déclenchement du *handover*). Dans le mode réactif, il faut que le *handover* se déclenche pour que le *MAG* commence la recherche d'une nouvelle adresse pour son noeud mobile.

2.2.3 Discussion

Nous présentons (dans un tableau comparatif) non seulemnt les forces et les faiblesses de chacune des deux familles de protocoles, à savoir le $HBMMP$ et $NBMMP$, mais aussi les différentes terminologies utilisées dans chacune d'elle.

	HBMMP	**NBMMP**
Ancrage de mobilité	HA, FA	LMA, MAG
Adresses de tunnel	CoA	$PCoA$
Messages de contrôle d'enregistrement	BU	PBU
Mise à jour	NM vers HA	MAG vers LMA
Déclenchement de *handover*	Manuel, par l'utilisateur	Automatique, par le MAG
Coûts de signalisation	élévé	faibles
Délais de latence	faibles	plus faibles

TABLE 2.1 – Comparaison entre le HBMMP et le NBMMP

Ce tableau montre que le $NBMMP$ offre de meilleures performances en termes de coûts de signalisation et de délais de latence, comme l'ont montré les travaux [Kha2014, Zho2013]. En effet, dans le $HBMMP$, pour chaque mouvement de noeud mobile (NM) , des messages de mise à jour sont échangés entre le NM et le HA. Cela peut créer des goulets d'étranglement au niveau des HA et augmente la latence, en particulier dans les systèmes où il existe plusieurs NM très mobiles. En revanche,

dans le cadre du $NBMMP$, pour les déplacements à l'intérieur d'un même MAG, il n'y a pas de messages de mise à jour à échanger. Cela permet aux LMA d'être décongestionnés. Les messages de mise à jour se produisent uniquement lorsque les NMs se déplacent entre différents LMA. En outre, dans $HBMMP$, lorsqu'il y a *handover*, l'enregistrement du nouvel emplacement d'un NM est effectué manuellement. En d'autres termes, une fois dans le réseau étranger, il revient à l'utilisateur de redémarrer la demande de connexion. Dans le $NBMMP$, il est automatiquement géré par le MAG dès qu'il détecte que le NM se déplace. Tous ces avantages du $NBMMP$ par rapport à $HBMMP$ expliquent le fait que de plus en plus, les recherches sur la gestion de la mobilité dans IPv6 évoluent vers des solutions du $NBMMP$.

2.3 Problèmes inhérents à la mobilité IPv6

Dans les communications mobiles, les utilisateurs se trouvent toujours dans des zones géographiques couvertes soit par une station de base, soit par un point d'accès sans fil, etc. En ayant la possibilité de communiquer tout en se déplaçant, les utilisateurs traversent très souvent plusieurs espaces géographiques dont le réseau de couverture n'est pas le même. Ce qui entraîne souvent les interruptions de communication, les pertes de paquets, et donc une mauvaise qualité du service car les différents réseaux traversés n'ont pas toujours les mêmes technologies d'accès. La mobilité dans internet a été introduite pour assurer la gestion des déplacements d'un ordinateur sur internet, c'est-à-dire le passage d'un réseau local à un autre. Ces nouvelles fonctionnalités ont été mieux prises en compte dans le protocole IPv6 qui donne la possibilité à un noeud mobile de se mouvoir entre plusieurs réseaux tout en conservant son adresse [Sil2006]. En effet, avec la mobilité sur IPv6, un noeud dispose au moins d'une adresse appelée *home address* et de la *care-of-address*. La *home address* est obtenue via son *Home Agent* ou son *LMA* selon que le noeud est dans un réseau dont la gestion est basée sur le mobile (*HBMMP*) ou qu'il est dans un réseau dont la gestion est basée sur le réseau (*NBMMP*). La *home address* ne change pas même lorsque le mobile change de réseau. C'est seulement la *CoA* qui change en fonction du réseau d'appartenance. Au niveau de chaque *Home Agent* et de chaque *LMA*, une correspondance est faite entre la *home address* du mobile et sa *care-of-address*. A chaque fois que le mobile change de réseau et que sa *CoA* change, une mise à jour est effectuée au niveau des *Home Agent* et des *LMAs*.

Cependant, il faut noter que la possibilité pour un noeud mobile de se déplacer à travers différents domaines tout en conservant sa *home address* n'a pas que des avantages. En effet, pour des mesures de sécurité, un administrateur peut empêcher l'accès à son domaine depuis l'extérieur. Il suffit juste pour cela de ne pas ajouter l'adresse de ce domaine à la table de routage de la passerelle. Dans ce cas, un noeud mobile ayant obtenu sa *home address* dans ce domaine (filtré), lorsqu'il se déplace vers un autre domaine (qui est accessible de l'extérieur), ne sera toujours pas accessible de l'extérieur car il garde sa *home address* du domaine filtré. En réalité dans la mobilité IPv6, avec l'optimisation de la route, les entêtes de routage sont utilisés pour l'envoi de messages. Les paquets sont envoyés avec la *home address* du noeud mobile [Sil2006]. Ce qui permet aux messages d'arriver au niveau des *Home*

Agent ou des *LMAs* qui, après vérification au niveau de la table de correspondance, acheminent le message au noeud mobile grâce à sa *care-of-address*.
Puisque le noeud mobile n'est plus dans le domaine filtré, il ne doit pas continuer à subir les restrictions de ce dernier. En d'autres termes, le noeud doit pouvoir être accessible depuis l'extérieur car il vient d'intégrer un nouveau réseau pour lequel l'accessibilité n'est pas interdite. Pour permettre cela, les réseaux pair-à-pair sont une solution. Car lorsque des noeuds sont connectés via un réseau virtuel, même lorsqu'un d'entre eux se déplace, les communications se poursuivent par l'intermédiaire des connexions logiques.

Toutefois, l'utilisation des réseaux pair-à-pair dans un environnement mobile suppose la mise en place de solutions qui tiennent compte des deux types de mobilités : la mobilité physique et la mobilité logique (cf. section 3). En effet, dans les réseaux *P2PSIP*, les noeuds arrivent et quittent le réseau à tout moment de manière incontrôlée, ce qui rend plus complexe la gestion des phases de *handover*. De plus, les réseaux *P2PSIP* sont caractérisés par une organisation plate. Or, dans la réalité, les noeuds n'ont pas toujours les mêmes performances. Il y a des noeuds dont les capacités de stockage ou de traitement sont supérieures aux autres. En les mettant tous sur le même niveau, les noeuds de plus faibles capacités peuvent anéantir les performances des autres.

Par ailleurs, l'introduction de la téléphonie dans les réseaux pair-à-pair *SIP* a fait de ces réseaux des systèmes de communication temps-réel dans lesquels le gain de temps est capital. Car, plus le délai de localisation est réduit, plus le service est fiable. Ainsi, l'utilisation d'une architecture pair-à-pair hiérarchique optimise la gestion de la mobilité, car permettant d'éviter que les noeuds de faibles capacités détériorent les performances des autres noeuds de plus grande capacité.

3 Gestion de la mobilité et réseaux pair-à-pair

La gestion de la mobilité dans les réseaux pair-à-pair obéit à un certain nombre d'exigences qu'il faut considérer. Il s'agit entre autres de :

- Mobilité d'un noeud : une bonne gestion des communications téléphoniques à travers un réseau *P2PSIP* doit permettre à un noeud de pouvoir changer de point d'attachement (ou de réseau) sans interruption de la communication.
- Mobilité d'un groupe de noeuds : il s'agit ici de la gestion de la mobilité de l'ensemble des noeuds mobiles avec leur point d'attachement. Pour être efficace, une solution de gestion de la mobilité doit inclure un tel scénario.
- *Handover* durant la localisation : il n'est pas rare qu'un utilisateur initie une communication en lançant un message *SIP INVITE*, et avant de recevoir une acceptation (message *200 OK*) de son correspondant, l'appelant change de point d'attachement (respectivement de réseau) à cause d'un changement de la topologie du réseau (à cause d'un déplacement). Par ailleurs, le message *INVITE* envoyé contient l'ancienne adresse du noeud mobile. Dans ce cas, la réponse *200 OK* de l'utilisateur appelé pourrait être envoyée à l'ancienne adresse de l'appelant. Ce qui bien sûr serait un échec car l'appelant s'est déplacé et a obtenu une autre adresse. La bonne solution serait que la réponse

200 OK soit directement acheminée vers la nouvelle adresse de l'appelant.

- *Handover* durant la communication : un des défis majeurs des communications mobiles dans les systèmes *P2PSIP* est de donner aux usagers la possibilité de communiquer sans interruption tout en changeant de point d'attachement ou de réseau physique.
- Latence du *handover* : dans les systèmes de communications en temps réel, le gain de temps reste un véritable challenge. La réduction du délai d'attente lors des phases de *handover* est nécessaire, plus ce délai est court mieux est la solution.
- Coûts de signalisation : le plus souvent, lors d'un *handover* beaucoup de messages sont échangés entre les différents participants à la communication. Cela génère des coûts importants en nombre de messages et peut impacter négativement sur le délai de *handover*. Il est donc important de mettre en place une stratégie pour minimiser la quantité de messages échangés.
- Perte de paquets : une perte excessive de paquets contribue à la dégradation de la qualité de la communication. Elle est donc à éviter.
- Surcharge des éléments qui jouent un rôle central : les éléments tels que les *LMAs*, les *MAGs*, les *Home Agents*, les *Foreign Agents* sont toujours au coeur du processus de *handover*. Ils sont sollicités lorsqu'un noeud mobile demande une adresse de connexion ou lorsqu'il entre dans la phase de *handover*, ou encore lorsqu'il est en période prédictive [2], etc. Ils sont souvent très surchargés. Pour mieux gérer la mobilité, il est important de ne pas les surcharger ce qui évitera les goulots d'étranglement.

La prise en compte de ces aspects entraîne celle des deux types de mobilité, à savoir la mobilité physique et la mobilité logique.

3.1 Mobilité physique

La mobilité physique est le déplacement d'un noeud d'une zone vers une autre ou d'une technologie d'accès vers une autre [John2004, Thie2002]. Ce mouvement entraîne un changement de la *care-of-address* du noeud. Des mises à jour doivent donc s'opérer au niveau du réseau pair-à-pair pour tenir compte de la nouvelle localisation du noeud (figure 2.4).

Le point d'attachement dispose d'une table dans laquelle est faite la correspondance entre la *home address* du noeud mobile et sa *care-of-address* avant le déplacement (*CoA 1*). Lorsque le noeud mobile bouge et se retrouve dans le réseau physique 2, il conserve sa *home address*, mais obtient une nouvelle *care-of-address* (*CoA 2*). Le point d'attachement doit donc mettre à jour sa table de correspondance pour tenir compte de la nouvelle *CoA* du noeud. Une solution de gestion de la mobilité doit gérer un tel scénario et permettre au noeud mobile (en déplacement) de continuer à conserver son lien logique avec le point d'attachement.

2. Période prédictive : c'est la période qui précède celle de handover

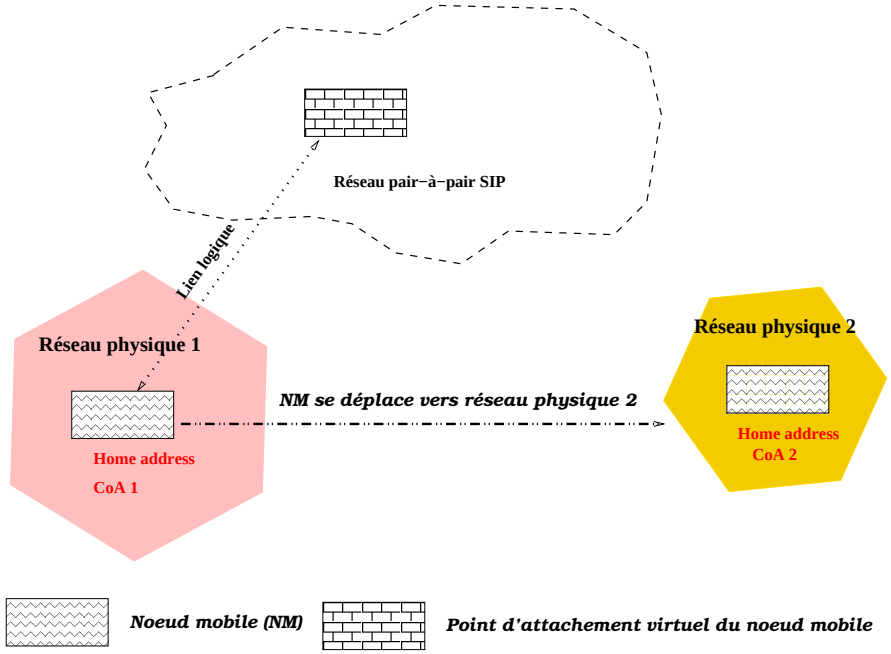


FIGURE 2.4 – Mobilité physique d'un noeud

3.2 Mobilité logique

Dans la mobilité logique, le noeud est sur place, mais à cause d'un changement de la topologie du réseau, il change de point d'attachement [John2004]. Ce changement de la topologie est souvent occasionné par des noeuds qui intègrent ou qui quittent le réseau. Un exemple est illustré à la figure 2.5.

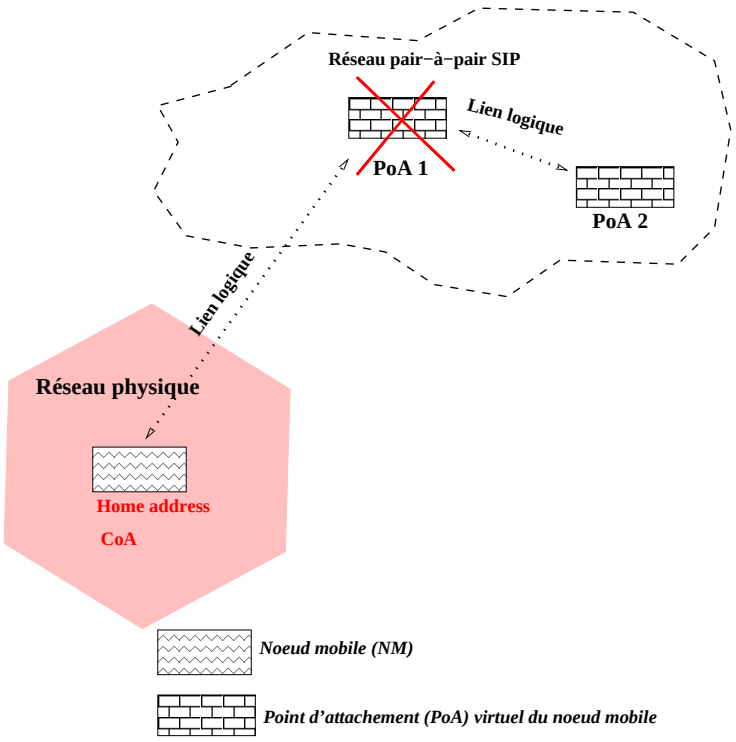


FIGURE 2.5 – Mobilité logique d'un noeud

Dans la figure 2.5, *PoA 1* est le point d'attachement du noeud mobile au niveau du réseau virtuel. A partir d'un certain moment, il quitte le réseau pair-à-pair. Le noeud mobile perd ainsi son lien logique et doit se connecter à un autre point d'attachement virtuel (*PoA 2*). Une solution de gestion de la mobilité dans un réseau pair-à-pair doit permettre ce changement de point d'attachement, mais surtout assurer la continuité du service que le noeud mobile entreprenait avant le départ de son point d'attachement.

4 Conclusion

Dans ce chapitre, nous avons rappelé les nombreux avantages du protocole internet version 6 (IPv6) sur la version 4 (IPv4) notamment l'infinité d'adresses, l'optimisation de la route, le gain de temps au niveau des routeurs, etc. Nous avons détaillé le fonctionnement des protocoles *FMIPv6* et *FPMIPv6* qui ont été mis en place pour améliorer les délais de latence lors des phases de *handover*, notamment avec leur méthodes prédictive et réactive. Nous avons également parcouru les différents types d'architectures pair-à-pair *SIP* hiérarchiques et avons expliqué les deux types de mobilité à gérer dans un environnement pair-à-pair mobile.

Ainsi, le chapitre suivant est consacré à des études comparatives d'une part, entre les modes prédictif et réactif, d'autre part, entre les différentes architectures. L'objectif étant de trouver la meilleure combinaison à mettre en place pour avoir une solution efficace de gestion de la mobilité dans un réseau *P2PSIP* hiérarchique. En d'autres termes, quel mode (prédictif ou réactif) combiner avec quel type d'architecture pour avoir une solution, optimisée, de gestion de la mobilité dans un réseau pair-à-pair *SIP* hiérarchique ?

Chapitre 3

Etudes comparatives des solutions de mobilité sur IPv6 et des architectures P2PSIP hiérarchiques

1 Introduction

Après avoir détaillé, au chapitre précédent, le fonctionnement des solutions de gestion de la mobilité dans IPv6 et des architectures *P2PSIP* hiérarchiques, nous menons dans ce chapitre présent des comparaisons analytiques entre d'une part, les solutions de la mobilité dans IPv6, et d'autre part, entre les architectures. En effet, tous les travaux qui se sont intéressés à la mobilité dans IPv6 se sont contentés de préciser qu'il existe le mode prédictif et le mode réactif et ont montré que la méthode prédictive permet de gagner du temps lors du *handover*, notamment grâce à l'anticipation sur le *handover*. Cependant, ils ne se sont pas préoccupés des nombreux messages de mise à jour générés lors des phases de *handover*. Pourtant, ils représentent un facteur déterminant dans la gestion du *handover* puisque, lorsqu'il y a trop de messages générés, cela impacte négativement sur l'occupation de la bande passante et par la même occasion sur le délai de localisation. Or, les réseaux pair-à-pair *SIP* hiérarchiques, n'utilisant pas les mêmes protocoles de routage et de localisation, n'ont pas toujours les mêmes performances en termes de surcharge du réseau.

En combinant une solution efficace de gestion de la mobilité avec un réseau *P2PSIP* hiérarchique qui surcharge beaucoup la bande passante, cela risque d'annihiler les performances de la solution. C'est pourquoi, nous menons ces études comparatives afin de voir la meilleure combinaison, c'est-à-dire, lequel des modes (prédictif ou réactif) coupler avec quel type d'architecture *P2PSIP* hiérarchique, permet de mettre en place une solution efficace.

Ainsi, à la section 2, nous faisons la comparaison des solutions de gestion de la mobilité dans IPv6 en termes de délais de *handover* et de coûts de mise à jour. Dans la section 3, nous comparons les différents types d'architectures pair-à-pair *SIP* hiérarchiques en termes de coûts de localisation. La section 4 est consacrée à synthèse des résultats issus des comparaisons. Dans la section 5, nous donnons une conclusion et des perspectives.

2 Etude comparative des solutions de la mobilité dans IPv6

Le *FMIPv6* et le *FPMIPv6* sont deux modes de fonctionnement beaucoup plus proches de la couche physique que de la couche application. Or, la plupart des communications mobiles actuelles utilisent des applications pair-à-pair qui tournent au niveau de la couche application. Il est donc nécessaire de voir comment combiner la mobilité dans IPv6 avec une architecture pair-à-pair hiérarchique, étant entendu que la hiérarchisation des noeuds permet d'améliorer le temps de latence dans la localisation mais aussi dans le traitement des requêtes. Dans ce cas, les *LMAs* et les *MAGs* représentent les super noeuds (SN) pour des solutions utilisant le *FPMIPv6*. Tandis que pour des solutions fonctionnant avec le *FMIPv6*, ce sont les *HAs* et les *FAs* qui jouent le rôle de super noeuds. Les noeuds mobiles simples et les points d'accès représentent quant à eux les noeuds mobiles.

Remarque 2.1 *Une adresse IPv6 est composée de* 128 *bits regroupés en huit (08) champs de 16 bits chacun. Les champs sont représentés par des chiffres hexadécimaux et séparés par deux points (" :"). Une adresse IPv6 se présente sous la forme suivante :*

Adresse IPv6 de 128 bits

n bits	*m bits*	$(128 - n - m)$ *bits*
ID de site	*ID de sous réseau*	*ID d'interface*

Dans le cadre de la mobilité dans IPv6, les ID de sites sont souvent réservés aux *LMAs*, *HAs* et *FAs*. Tandis que les ID de sous réseau sont pour les *MAGs* et les ID d'interface pour les noeuds mobiles (NMS et PA).

Pendant et après une phase de *handover*, plusieurs informations de localisation doivent être mises à jour, entraînant ainsi des coûts supplémentaires. Nous définissons les coûts de mise à jour comme le temps nécessaire à un noeud pour traiter une unité d'information. Ils s'expriment en secondes par bit (s/bit).
Durant toute notre analyse, nous utilisons les notations suivantes :

- $T_{X/Y}$: Délai de transmission entre le noeud X et le noeud Y.
- T_{update} : Temps nécessaire à un noeud mobile pour mettre à jour son adresse.
- $C_{X/Y}$: Coûts de mise à jour entre le noeud X et le noeud Y.
- $C_{update} = \frac{T_{update}}{S}$: Coût nécessaire à un noeud mobile simple (NMS) pour mettre à jour son adresse.
- S : Nombre de bits à mettre à jour au niveau de l'address IPv6 lorsqu'un NMS se déplace ($S \leq 128$ bits).
- Pre : mode prédictif.
- $Reac$: mode réactif.
- $AvHO$: Avant *handover*.
  X,$Y \in \{NM, MAG, LMA, HA, FA\}$

Hypothèses : Sans pour autant perdre la généralité, nous supposons que :

1. Le délai de transmission entre un NMS et un point d'accès (PA) est négligeable.

2. Le délai de transmission entre les MAG, LMA, HA, FA est le même. En d'autres termes : $T_{MAG/MAG} = T_{MAG/LMA} = T_{HA/FA} = T_{LMA/LMA}$
3. $T_{NMS/HA} = T_{NMS/FA}$ et $T_{PA/HA} = T_{PA/FA}$
4. Quel que soit X,$Y \in \{NM, MAG, LMA, HA, FA\}$, $T_{X/Y} = T_{Y/X}$ et $C_{X/Y} = C_{Y/X} = \frac{T_{X/Y}}{S}$

Nous ne considérons que les mouvements inter-domaines de noeuds mobiles (noeuds mobiles simples et points d'accès). Pour chacun des cas suivants, nous calculons le délai de *handover* et les coûts de mise à jour :

- Un NMS se déplace de son point d'attachement à un autre se trouvant dans un autre domaine.
- Un point d'accès (PA) se déplace de son point d'attachement à un autre se trouvant dans un autre domaine.

2.1 Calcul des coûts de mise à jour et des délais de handover dans FMIPv6

Le calcul des coûts s'appuie sur le fonctionnement défini à la figure 2.1, page 28.

2.1.1 Handover d'un noeud mobile simple (NMS)

Calcul du délai de handover : Ce délai correspond au temps qui sépare la déconnexion du noeud mobile simple de son *HA* et sa connexion à son nouveau *FA*.

- Avec le mode prédictif : Dans ce mode, un NMS envoie un message au *FA* pour demander une nouvelle adresse ($T^{AvHO}_{NMS/FA}$). Le *FA* répond en envoyant une nouvelle adresse au NMS ($T^{AvHO}_{FA/NMS}$). Le NMS achemine ce message à son *HA* ($T^{AvHO}_{NMS/HA}$) pour lui notifier qu'il dispose désormais d'une nouvelle adresse et doit entrer en phase de *handover*. Tous ces messages se font avant la déconnexion du NMS. Lorsque le *handover* s'achève, le *FA* envoie un message au *HA* du noeud mobile ($T_{FA/HA}$) afin que ce dernier mette à jour ces informations. Le *FA* notifie le NMS par un message ($T_{FA/NMS}$) qu'il l'a bien enregistré. Ainsi, le délai du *handover* est :

$$T^{Pre}_{FMIPv6} = T^{AvHO}_{NMS/FA} + T^{AvHO}_{FA/NMS} + T^{AvHO}_{NMS/HA} + T_{FA/HA} + T_{FA/NMS} + T_{update}$$

  Toutefois, les messages permettant d'anticiper sur le *handover* ne sont pas comptabilisés pour la durée du *handover* car étant envoyés avant même que le NMS se déconnecte de son *HA*. En d'autres termes, $T^{AvHO}_{X/Y} = 0$. Donc,

$$T^{Pre}_{FMIPv6} = T_{FA/HA} + T_{FA/NMS} + T_{update} \quad (3.1)$$

- Avec le mode réactif : Dans ce mode, dès que le *handover* est déclenché, le noeud mobile simple envoie un message à son *HA* ($T_{NMS/HA}$) pour l'informer qu'il est entré en phase de *handover*, le *HA* envoie un message au *FA* ($T_{HA/FA}$) pour lui demander une nouvelle adresse pour ce NMS. Ce message contient les informations du noeud mobile simple. Le *FA* envoie la nouvelle adresse au noeud mobile simple en déplacement ($T_{FA/NMS}$).

$$T^{Reac}_{FMIPv6} = T_{NMS/HA} + T_{HA/FA} + T_{FA/NMS} + T_{update}$$

Nous rappelons que, $T_{NMS/HA} = T_{NMS/FA}$, selon l'hypothèse 3). Donc,

$$T^{Reac}_{FMIPv6} = 2 \times T_{NMS/HA} + T_{HA/FA} + T_{update} \tag{3.2}$$

Calcul des coûts de mise à jour : Alors que pour le calcul du délai de *handover*, il fallait considérer uniquement le temps effectif du *handover*, pour les coûts de mise à jour par contre, doivent être pris en compte tous les messages envoyés avant, pendant et après la phase de *handover* et ayant contribué à la gestion de ce *handover*. Car ces messages participent à la surcharge du réseau.

- Avec le mode prédictif : En plus des messages envoyés pour anticiper sur le *handover* ($C^{AvHO}_{NMS/FA}$, $C^{AvHO}_{FA/NMS}$ et $C^{AvHO}_{NMS/HA}$), il faut ajouter le message envoyé par le *HA* au *FA* ($C_{HA/FA}$) pour lui signifier qu'un de ses NMS se dirige vers lui et celui répondu par le *FA* ($C_{FA/HA}$) au *HA* pour informer ce dernier qu'il a effectivement enregistré le NMS et qu'il doit mettre à jour sa table d'index. Il y a également le message envoyé par le *FA* au NMS ($C_{FA/NMS}$) pour l'informer qu'il l'a bien enregistré comme étant un de ses NMS. Le NMS répondra par un message d'accusé de réception ($C_{NMS/FA}$). Ainsi, les coûts de mise à jour sont :

$$C^{Pre}_{FMIPv6} = C^{AvHO}_{NMS/FA} + C^{AvHO}_{FA/NMS} + C^{AvHO}_{NMS/HA} + C_{HA/FA} + C_{FA/NMS} + C_{NMS/FA} + C_{FA/HA} + C_{update}$$

Puisque $T_{NMS/HA} = T_{NMS/FA}$ (hypothèse 3), alors
$C_{NMS/HA} = \frac{T_{NMS/HA}}{S} = \frac{T_{NMS/FA}}{S} = C_{NMS/FA} = C_{FA/NMS}$,
D'où,

$$C^{Pre}_{FMIPv6} = 5 \times C_{NMS/HA} + 2 \times C_{HA/FA} + C_{update} \tag{3.3}$$

- Avec le mode réactif : De la même manière, le NMS informe son *HA* qu'il est entré en phase de *handover* ($C_{NMS/HA}$), le *HA* demande au FA de fournir une adresse à son NMS ($C_{HA/FA}$), le *FA* fournit une adresse et l'envoie au NMS ($C_{FA/NMS}$), le NMS répond au *FA* par un accusé de réception ($C_{NMS/FA}$), le *FA* l'enregistre comme un de ses NMS puis informe son ancien *HA* ($C_{FA/HA}$) afin qu'il mette à jour sa table d'index. Ainsi, les coûts de mise à jour s'obtiennent par la formule ci-après :

$$C^{Reac}_{FMIPv6} = C_{NMS/HA} + C_{HA/FA} + C_{FA/NMS} + C_{NMS/FA} + C_{FA/HA} + C_{update}$$

$$C^{Reac}_{FMIPv6} = 3 \times C_{NMS/HA} + 2 \times C_{HA/FA} + C_{update} \tag{3.4}$$

2.1.2 Handover d'un point d'accès (PA)

Le point d'accès se déplace avec l'ensemble des noeuds mobiles simples qui lui sont rattachés. Durant la phase de *handover*, les messages sont échangés uniquement entre le point d'accès et les *HA* et *FA*. Les noeuds mobiles simples n'interviennent que lorsque le *handover* du point d'accès est achevé. Dans ce cas, tous les noeuds mobiles simples du point d'accès doivent mettre à jour leur adresse.

Nous supposons qu'il y a k noeuds mobiles simples rattachés au point d'accès en déplacement.

Calcul du délai de handover :

- Avec le mode prédictif : le même processus que celui avec un noeud mobile simple (section précédente) se produit. La seule différence est que pour ce cas présent, les messages sont échangés entre le point d'accès et le *HA* ou le *FA*. En plus de ce délai engendré par ces messages, nous devons tenir compte du temps nécessaire à chaque noeud mobile simple pour mettre à jour son adresse. Sans enfreindre la généralité, nous supposons que tous les noeuds mobiles simples utilisent sensiblement le même délai (T_{update}) pour mettre à jour leur adresse.

  $$T^{Pre}_{FMIPv6} = T^{AvHO}_{PA/FA} + T^{AvHO}_{FA/PA} + T^{AvHO}_{PA/HA} + T_{HA/FA} + T_{FA/PA} + k \times T_{update}$$

  De la même manière, $T^{AvHO}_{X/Y} = 0$, car les messages sont envoyés avant le *handover*, donc ils ne sont pas comptabilisés pour la durée du *handover*.

  $$T^{Pre}_{FMIPv6} = T_{HA/FA} + T_{FA/PA} + k \times T_{update} \tag{3.5}$$

- Avec le mode réactif : de façon similaire, nous donnons le délai du *handover* avec le mode réactif par la formule suivante :

  $$T^{Reac}_{FMIPv6} = T_{PA/HA} + T_{HA/FA} + T_{FA/PA} + k \times T_{update}$$

  $T_{PA/HA} = T_{PA/FA}$, (hypothèse 3). Ceci donne :

  $$T^{Reac}_{FMIPv6} = 2 \times T_{PA/HA} + T_{HA/FA} + k \times T_{update} \tag{3.6}$$

Calcul des coûts de mise à jour : les coûts de mise à jour d'un point d'accès s'obtiennent de la même manière qu'avec un noeud mobile simple en tenant compte des k noeuds mobiles simples rattachés au point d'accès.

- Avec le mode prédictif :

  $$C^{Pre}_{FMIPv6} = C^{AvHO}_{PA/FA} + C^{AvHO}_{FA/PA} + C^{AvHO}_{PA/HA} + C_{HA/FA} + C_{FA/PA} + C_{PA/FA} + C_{FA/HA} + k \times C_{update}$$

  $C_{PA/HA} = \frac{T_{PA/HA}}{S} = \frac{T_{PA/FA}}{S} = C_{PA/FA} = C_{FA/PA}$, Donc,

  $$C^{Pre}_{FMIPv6} = 5 \times C_{PA/HA} + 2 \times C_{HA/FA} + k \times C_{update} \tag{3.7}$$

- Avec le mode réactif :

  $$C^{Reac}_{FMIPv6} = C_{PA/HA} + C_{HA/FA} + C_{FA/PA} + C_{PA/FA} + C_{FA/HA} + k \times C_{update}$$

  $$C^{Reac}_{FMIPv6} = 3 \times C_{PA/HA} + 2 \times C_{HA/FA} + k \times C_{update} \tag{3.8}$$

2.2 Calcul des coûts de mise à jour et des délais de handover dans FPMIPv6

Nous procédons de la même manière comme nous l'avons fait pour le *FMIPv6*. Il suffit juste d'ajouter les coûts de mise à jour et de délai entre les noeuds mobiles (NMS et PA) et les *MAGs* d'une part, et entre les *MAGs* et les *LMAs* d'autre part. Les coûts se calculent suivant le fonctionnement défini à la figure 2.3, page 29. Nous rappelons que dans le *PMIPv6*, les *HAs* et *FAs* sont remplacés par les *LMAs*. Ceci donne les coûts et les délais suivants :

2.2.1 Handover d'un noeud mobile simple (NMS)

Calcul du délai de handover :

- Avec le mode prédictif :

$$T^{Pre}_{FPMIPv6} = T_{LMA/LMA} + T_{LMA/MAG} + T_{MAG/NMS} + T_{update} \quad (3.9)$$

- Avec le mode réactif :

$$T^{Reac}_{FPMIPv6} = 2 \times T_{MAG/NMS} + 2 \times T_{LMA/MAG} + T_{LMA/LMA} + T_{update} \quad (3.10)$$

Calcul des coûts de mise à jour :

- *Avec le mode prédictif* :

$$C^{Pre}_{FPMIPv6} = 5 \times C_{NMS/MAG} + 6 \times C_{MAG/LMA} + 2 \times C_{LMA/LMA} + C_{update} \quad (3.11)$$

- *Avec le mode réactif* :

$$C^{Reac}_{FPMIPv6} = 3 \times C_{NMS/MAG} + 4 \times C_{MAG/LMA} + 2 \times C_{LMA/LMA} + C_{update} \quad (3.12)$$

2.2.2 Handover d'un point d'accès (PA)

Le point d'accès se déplace avec l'ensemble de ses k noeuds mobiles simples.

Calcul du délai de handover :

- Avec le mode prédictif :

$$T^{Pre}_{FPMIPv6} = T_{LMA/LMA} + T_{LMA/MAG} + T_{MAG/PA} + k \times T_{update} \quad (3.13)$$

- Avec le mode réactif :

$$T^{Reac}_{FPMIPv6} = 2 \times T_{MAG/PA} + 2 \times T_{LMA/MAG} + T_{LMA/LMA} + k \times T_{update} \quad (3.14)$$

Calcul des coûts de mise à jour :

- Avec le mode prédictif :

$$C^{Pre}_{FPMIPv6} = 5 \times C_{PA/MAG} + 6 \times C_{MAG/LMA} + 2 \times C_{LMA/LMA} + k \times C_{update} \quad (3.15)$$

- Avec le mode réactif :

$$C^{Reac}_{FPMIPv6} = 3 \times C_{PA/MAG} + 4 \times C_{MAG/LMA} + 2 \times C_{LMA/LMA} + k \times C_{update} \tag{3.16}$$

Remarque 2.2 *En observant l'expression des coûts de mise à jour ($C_{X/Y} = \frac{T_{X/Y}}{S}$) avec $0 < S \leq 128$, nous pourrions penser que, plus la valeur de S est grande, moins grands seront les coûts de mise à jour. Pourtant il n'en est rien. En effet, lorsque S tend vers 128 bits, cela signifie que les mises à jour atteignent les LMAs ou les HAs. Or, les mises à jour commencent depuis le noeud mobile (NM). Dans ce cas, les coûts sont : $C_{NM/LMA} = C_{NM/MAG} + C_{MAG/LMA} = \frac{T_{NM/MAG}}{128-n} + \frac{T_{MAG/LMA}}{128}$*

Alors que si la valeur de S est petite, seuls sont concernés les IDs de sous réseau et d'interface. Les mises à jour n'atteignent pas les LMAs ni les HAs. Elles s'arrêtent au niveau des MAGs. C'est dans le cadre des mouvements intra-domaine par exemple. Dans ce cas, le coût est : $C_{NM/MAG} = \frac{T_{NM/MAG}}{128-n}$.
Donc $C_{NM/LMA} > C_{NM/MAG}$

Ceci montre que pour les mouvements inter-domaines, les coûts de mise à jour sont plus importants que dans le cadre de mouvements intra-domaines.

2.3 Comparaison des modes prédictif et réactif dans FMIPv6 et FPMIPv6

Les équations 3.1 et 3.2 d'une part et, 3.5 et 3.6 d'autre part, montrent que dans le *FMIPv6* le délai de *handover* en mode prédictif est plus faible qu'en mode réactif. Ceci est valable aussi bien pour les déplacements de noeuds mobiles simples que les déplacements de points d'accès.
Par contre les couples d'équations (3.3 , 3.4) et (3.7, 3.8) montrent que le mode prédictif surcharge beaucoup plus la bande passante que le mode réactif.

Les mêmes remarques sont faites pour le *FPMIPv6* avec d'une part les équations 3.9 et 3.10 et d'autre part, les équations 3.11 , 3.12.

Le tableau 3.1 résume les relations de comparaison des différents délais et des coûts de mise à jour calculés.

	FMIPv6		***FPMIPv6***	
	Prédictif	Réactif	Prédictif	Réactif
Délai de handover (NMS, PA)	$T^{Pre}_{FMIPv6} < T^{Reac}_{FMIPv6}$		$T^{Pre}_{FPMIPv6} < T^{Reac}_{FPMIPv6}$	
Coûts de mise à jour (NMS,PA)	$C^{Pre}_{FMIPv6} > C^{Reac}_{FMIPv6}$		$C^{Pre}_{FPMIPv6} > C^{Reac}_{FPMIPv6}$	

TABLE 3.1 – Tableau comparatif des modes prédictif et réactif

3 Etude comparative des différents types d'architectures P2PSIP hiérarchiques

Dans cette section, nous menons une étude théorique afin de déterminer le nombre de messages exact générés dans chacune des trois types d'architectures décrites à la section 6 du chapitre 1. Contrairement à [Art2007] où les auteurs ont fait une comparaison entre les systèmes plats et les systèmes hiérarchiques et ont montré que la hiérarchisation était bénéfique, dans notre cas, nous comparons les systèmes pair-à-pair SIP hiérarchiques entre eux. Comme il a été prouvé dans les travaux de Bryan et al. [Bry2008] et de Zoels et al. [Zoels2006], le nombre de messages générés dépend fortement de la méthode de localisation utilisée. En effet, dans [Bry2008] les auteurs ont donné de manière générique le nombre de messages échangés par un noeud en fonction de la méthode de recherche (itérative, récursive ou semi-récursive). Quant à Zoels et al. dans [Zoels2006], ils ont déterminé les coûts de recherche et de maintenance seulement dans une architecture hiérarchique à un seul domaine (Figure 1.10). Pour les coûts de recherche, ils font la différence entre coûts de recherche des super noeuds (SN) et coûts de recherche des noeuds ordinaires (NO). Ils utilisent le nombre de messages échangés par chaque noeud sans pour autant fournir sa véritable valeur. Dans notre étude, nous calculons de manière formelle cette valeur en nous appuyant sur [Bry2008]. Ensuite, nous utilisons cette expression pour trouver une formule générale donnant le nombre total de messages générés dans chaque architecture. Enfin, dans chacune des architectures, nous déterminons formellement le coût de chaque méthode de routage en termes de nombre de messages échangés.

Dans notre cas, nous nous focalisons sur le nombre de messages de localisation. Pour ce faire, nous utilisons Chord comme table de hachage distribuée qui est celle employée dans [Zoels2006].

3.1 Architectures P2PSIP hiérarchiques à un domaine de recouvrement

3.1.1 Architecture hiérarchique à seul domaine avec une seule couche

L'architecture hiérarchique à seul domaine avec une seule couche (*Hierarchical Single Domain Architecture : H-SDA*) est celle illustrée à la Figure 1.10. Nous rappelons ici certaines formules utilisées dans [Zoels2006]. Dans celles-ci, les termes qui nous intéressent sont ($R_{LKP,ON}$ et $R_{LKP,SN}$). Ces derniers représentent respectivement le nombre de messages envoyés/reçus par un noeud ordinaire (*Ordinary Node - ON*) respectivement un super noeud durant la procédure de recherche (*lookup (LKP)*) de noeud. Ainsi, en nous appuyant des travaux de [Bry2008] sur les coûts de méthodes de routage, nous déterminons ces termes. Dans la suite de la section, nous utilisons les notations suivantes :

- $\mathcal{N}_{SN}$: nombre de Super Noeuds (SN).
- $\mathcal{N}_{ON}$: nombre de noeuds ordinaires (*Ordinary Nodes - ON*).
- $\mathcal{N}_{ON_i}$: nombre de noeuds ordinaires rattachés au $SN\ i$.
- $\mathcal{C}_{LKP}^{H-SDA}$: coût total de recherche (*lookup*) dans *H-SDA*.
- $\mathcal{C}_{LKP,ON}^{H-SDA}$: coût total de recherche pour tous les noeuds ordinaires dans *H-SDA*.

- $\mathcal{C}_{LKP,SN}^{H-SDA}$: coût total de recherche pour tous les SN dans *H-SDA*.

D'après [Zoels2006], le coût de recherche dans *H-SDA* est obtenu comme suit :

$$\mathcal{C}_{LKP}^{H-SDA} = \mathcal{C}_{LKP,ON}^{H-SDA} + \mathcal{C}_{LKP,SN}^{H-SDA} \tag{3.17}$$

Coûts de recherche des noeuds ordinaires : Pour déterminer le coût de recherche des noeuds ordinaires, nous utilisons les notations suivantes :

- $\mathcal{R}_{LKP,ONj}$: nombre de messages envoyés/reçus par le noeud ordinaire j.
- $\mathcal{C}_{ONj}^{e}$: coût généré par le noeud ordinaire j pour l'envoi d'un message.
- $\mathcal{C}_{ONj}^{r}$: coût généré par le noeud ordinaire j pour la réception d'un message.

Selon [Zoels2006], le coût pour tous les noeuds ordinaires est donné par l'équation suivante :

$$\mathcal{C}_{LKP,ON}^{H-SDA} = \sum_{j=1}^{\mathcal{N}_{ON}} (\mathcal{C}_{ON_j}^{e} + \mathcal{C}_{ON_j}^{r}) \times \mathcal{R}_{LKP,ON_j} \tag{3.18}$$

Coûts de recherche des super noeuds : Pour déterminer le coût de recherche des noeuds ordinaires, nous utilisons les formules ci-après.

- $\mathcal{R}_{LKP,SN_i}$: nombre de messages envoyés/reçus par le super noeud i.
- $\mathcal{N}_{mess}^{H-SDA}$: nombre de messages envoyés/reçus par tous les noeuds dans *H-SDA*.
- $\mathcal{C}_{SN_i}^{e}$: coûts générés par le super noeud i pour l'envoi d'un message.
- $\mathcal{C}_{SN_i}^{r}$: coûts générés par le super node i for la réception d'un message.

Dans [Zoels2006], il est montré que le nombre total de messages générés par chaque super noeud i (SN_i) (respectivement chaque noeud ordinaire j (ON_j)), qui lance une recherche dans un anneau chord est donné respectivement par :

$$\mathcal{M}_{(SN_i)} = \mathcal{R}_{LKP,SN_i} \times \log_2(\mathcal{N}_{SN}) \tag{3.19}$$

$$\mathcal{M}_{(ON_j)} = \mathcal{R}_{LKP,ON_j} \times [\log_2(\mathcal{N}_{SN}) + 1] \tag{3.20}$$

Ainsi,utilisant les équations (3.19) et (3.20), nous calculons le nombre de messages total échangés (noté $\mathcal{N}_{mess}^{H-SDA}$) par tous les noeuds (super noeuds et noeuds ordinaires) dans *H-SDA* comme suit :

$$\mathcal{N}_{mess}^{H-SDA} = \sum_{i=1}^{\mathcal{N}_{SN}} \mathcal{M}_{(SN_i)} + \sum_{i=1}^{\mathcal{N}_{SN}} \sum_{j=1}^{\mathcal{N}_{ON_i}} \mathcal{M}_{(ON_j)} \tag{3.21}$$

$$\mathcal{N}_{mess}^{H-SDA} = \sum_{i=1}^{\mathcal{N}_{SN}} \mathcal{R}_{LKP,SN_i} \times \log_2(\mathcal{N}_{SN}) + \sum_{i=1}^{\mathcal{N}_{SN}} \sum_{j=1}^{\mathcal{N}_{ON_i}} \mathcal{R}_{LKP,ON_j} \times [\log_2(\mathcal{N}_{SN}) + 1]$$

Finalement, après réorganisation, le nombre total de messages générés dans *H-SDA* s'obtient par la formule suivante :

$$\mathcal{N}_{mess}^{H-SDA} = \log_2(\mathcal{N}_{SN}) \times \sum_{i=1}^{\mathcal{N}_{SN}} \mathcal{R}_{LKP,SN_i} + [\log_2(\mathcal{N}_{SN}) + 1] \times \sum_{i=1}^{\mathcal{N}_{SN}} \sum_{j=1}^{\mathcal{N}_{ON_i}} \mathcal{R}_{LKP,ON_j} \tag{3.22}$$

Puisque le nombre de messages échangés dépend de la méthode de recherche utilisée, après avoir déterminé une formule générale (équation 3.22), nous nous intéressons aux coûts de chacune des méthodes de recherche. Notre analyse se focalise sur les méthodes les moins coûteuses, à savoir les méthodes itérative, récursive et semi-récursive (cf Chapitre 1 - section 3.3). Dans ce qui suit, nous utilisons les notations suivantes :

- Ite=itérative
- Rec=récursive
- SRec=semi-récursive

Recherche par la méthode itérative : Comme nous le rappelions plus haut (cf Chapitre 1 - section 3.3), le nombre de messages générés par noeud avec la méthode itérative est de $2 \times (n-1)$, où n est le nombre de noeuds participant à la distribution (au routage). En appliquant cette théorie dans les différents termes $\mathcal{R}_{LKP}$, nous obtenons :

$$\mathcal{R}_{LKP,SN_i} = 2(\mathcal{N}_{SN} - 1) \tag{3.23}$$

$$\mathcal{R}_{LKP,ON_j} = 2(\mathcal{N}_{SN} - 1) + 2 = 2\mathcal{N}_{SN} \tag{3.24}$$

Remarque 3.1 *Pour les noeuds ordinaires, puisque la recherche est effectuée par les super noeuds, il y a donc* $2(\mathcal{N}_{SN} - 1)$ *[Zoels2006]. En plus, il est nécessaire de considérer les messages envoyé/reçu au/du super noeud (c'est-à-dire* $+2$*).*

Pour tous les super noeuds, le nombre total de messages générés durant le processus de localisation est :

$$\sum_{i=1}^{\mathcal{N}_{SN}} \mathcal{R}_{LKP,SN_i} = \sum_{i=1}^{\mathcal{N}_{SN}} 2(\mathcal{N}_{SN} - 1) = 2 \times \mathcal{N}_{SN} \times (\mathcal{N}_{SN} - 1) \tag{3.25}$$

Pour tous les noeuds ordinaires, le nombre total de messages générés durant le processus de localisation est :

$$\sum_{j=1}^{\mathcal{N}_{ONi}} \mathcal{R}_{LKP,ON_j} = \sum_{j=1}^{\mathcal{N}_{ON_i}} 2\mathcal{N}_{SN} = 2 \times \mathcal{N}_{ON_i} \times \mathcal{N}_{SN} \tag{3.26}$$

En injectant les équations (3.25) et (3.26) dans l'équation (3.22), nous déduisons la formule suivante qui donne le nombre total de messages générés (noté $\mathcal{N}_{mess}^{H-SDA}(Ite)$).

$$\mathcal{N}_{mess}^{H-SDA}(Ite) = \log_2(\mathcal{N}_{SN}) \times 2 \times \mathcal{N}_{SN} \times (\mathcal{N}_{SN} - 1) + [\log_2(\mathcal{N}_{SN}) + 1] \times \sum_{i=1}^{\mathcal{N}_{SN}} 2 \times \mathcal{N}_{ON_i} \times \mathcal{N}_{SN}$$

Finalement, après réorganisation et simplification, nous obtenons :

$$\mathcal{N}_{mess}^{H-SDA}(Ite) = 2 \times \mathcal{N}_{SN} \times [(\mathcal{N}_{SN} - 1) \times \log_2(\mathcal{N}_{SN}) + [\log_2(\mathcal{N}_{SN}) + 1] \times \sum_{i=1}^{\mathcal{N}_{SN}} \times \mathcal{N}_{ON_i}] \tag{3.27}$$

Recherche par la méthode récursive : Dans la méthode récursive, puisque le nombre de messages générés par un noeud est également de $2\times(n-1)$ (cf Chapitre 1 - section 3.3), alors le nombre total de messages générés par tous les noeuds (noté $\mathcal{N}_{mess}^{H-SDA}(Rec)$) est le même que pour la méthode itérative.

$$\mathcal{N}_{mess}^{H-SDA}(Ite) = \mathcal{N}_{mess}^{H-SDA}(Rec) = \mathcal{N}_{mess}^{H-SDA}(Ite/Rec) \tag{3.28}$$

Recherche par la méthode semi-récursive : Contrairement aux méthodes itérative et récursive, dans la semi-récursive, le nombre de messages générés par un noeud est de n, avec n le nombre de noeuds participant à la distribution. De la même manière, en appliquant dans les différents termes $\mathcal{R}_{LKP}$, cela donne :

$$\mathcal{R}_{LKP,SN_i} = \mathcal{N}_{SN}$$

$$\mathcal{R}_{LKP,ON_j} = \mathcal{N}_{SN} + 2$$

Ainsi :

$$\sum_{i=1}^{\mathcal{N}_{SN}} \mathcal{R}_{LKP,SN_i} = \sum_{i=1}^{\mathcal{N}_{SN}} \mathcal{N}_{SN} = (\mathcal{N}_{SN})^2 \tag{3.29}$$

$$\sum_{j=1}^{\mathcal{N}_{ON_i}} \mathcal{R}_{LKP,ON_j} = \sum_{j=1}^{\mathcal{N}_{ON_i}} (\mathcal{N}_{SN}+2) = \mathcal{N}_{ONi} \times (\mathcal{N}_{SN}+2) \tag{3.30}$$

En injectant les équations (3.29) et (3.30) dans l'équation (3.22), nous déduisons la formule donnant le nombre total de messages générés dans la méthode semi-récursive (noté $\mathcal{N}_{mess}^{H-SDA}(SRec)$).

$$\mathcal{N}_{mess}^{H-SDA}(SRec) = (\mathcal{N}_{SN})^2 \times \log_2(\mathcal{N}_{SN}) + (\mathcal{N}_{SN}+2) \times [\log_2(\mathcal{N}_{SN})+1] \sum_{i=1}^{\mathcal{N}_{SN}} \mathcal{N}_{ON_i} \tag{3.31}$$

Remarque 3.2 *Dans le cas particulier où le nombre de noeuds ordinaires est le même pour chaque super noeud (i.e. $\mathcal{N}_{ON_i} = \mathcal{N}_{ON/SN}$), cela donne :*

$$\sum_{i=1}^{\mathcal{N}_{SN}} \mathcal{N}_{ON_i} = \sum_{i=1}^{\mathcal{N}_{SN}} \mathcal{N}_{ON/SN} = \mathcal{N}_{SN} \times \mathcal{N}_{ON/SN} \tag{3.32}$$

Et donc :

- Dans les méthodes itérative ou récursive, le nombre de messages échangés est :

$$\mathcal{N}_{mess}^{H-SDA}(Ite/Rec) = 2\times\mathcal{N}_{SN}\times[(\mathcal{N}_{SN}-1)\times(\log_2(\mathcal{N}_{SN}))+\mathcal{N}_{SN}\times\mathcal{N}_{ON/SN}\times(\log_2(\mathcal{N}_{SN})+1)] \tag{3.33}$$

- Dans la méthode semi-récursive, le nombre de messages générés est :

$$\mathcal{N}_{mess}^{H-SDA}(SRec) = (\mathcal{N}_{SN})^2\times\log_2(\mathcal{N}_{SN})+\mathcal{N}_{SN}\times\mathcal{N}_{ON/SN}\times(\mathcal{N}_{SN}+2)\times[\log_2(\mathcal{N}_{SN})+1] \tag{3.34}$$

3.1.2 Architecture hiérarchique à seul domaine avec plusieurs couches

L'architecture hiérarchique à seul domaine avec plusieurs couches (*Hierarchical Multi-Layered Architecture : H-MLA*) est représentée à la Figure 1.11. Dans notre cas, les différentes couches utilisent Chord comme table de hachage distribuée. Pour la suite de cette section, nous utilisons les notations suivantes :

- K : nombre de sous niveaux.
- $\mathcal{N}_{P_i}$: nombre de noeuds dans le sous niveau i.
- $\mathcal{N}_{SN}$: nombre de super noeuds dans le niveau supérieur (le niveau des SNs)
- $\mathcal{R}_{SN_i}$: nombre de messages envoyés/reçus par un super noeud i.
- $\mathcal{R}_{P_i}$: nombre de messages envoyés/reçus par un noeud i.

Dans l'architecture *H-MLA*, tous les noeuds participent à la distribution dans leur niveau respectif. Donc en appliquant les formules de Zoels et al. [Zoels2006], nous obtenons :

- Le nombre de messages générés (noté $\mathcal{M}_{(SN_i)}$) par un super noeud i qui initie une recherche est :
$$\mathcal{M}_{(SN_i)} = \mathcal{R}_{SN_i} \times \log_2(\mathcal{N}_{SN})$$
- Le nombre de messages générés (noté $\mathcal{M}_{(P_j)}$) par un noeud j appartenant au sous niveau i est :
$$\mathcal{M}_{(P_j)} = \mathcal{R}_{P_j} \times \log_2(\mathcal{N}_{P_i})$$

Le nombre total de messages envoyés et reçus dans le niveau supérieur ($Nsup$) (noté $\mathcal{N}^{H-MLA}_{mess/Nsup}$) et dans le niveau i (noté $\mathcal{N}^{H-MLA}_{mess/i}$) sont respectivement donnés par les équations suivantes :

$$\mathcal{N}^{H-MLA}_{mess/Nsup} = \sum_{i=1}^{\mathcal{N}_{SN}} \mathcal{M}_{(SN_i)} = \log_2(\mathcal{N}_{SN}) \times \sum_{i=1}^{\mathcal{N}_{SN}} \mathcal{R}_{SN_i} \tag{3.35}$$

$$\mathcal{N}^{H-MLA}_{mess/i} = \sum_{j=1}^{\mathcal{N}_{P_i}} \mathcal{M}_{(P_j)} = \log_2(\mathcal{N}_{P_i}) \times \sum_{j=1}^{\mathcal{N}_{P_i}} \mathcal{R}_{P_j} \tag{3.36}$$

Selon les équations (3.35) et (3.36), le nombre total de messages générés dans H-MLA est :

$$\mathcal{N}^{H-MLA}_{mess} = \mathcal{N}^{H-MLA}_{mess/Nsup} + \sum_{i=1}^{k} \mathcal{N}^{H-MLA}_{mess/i} \tag{3.37}$$

Finalement, après réorganisation et simplification, la formule devient :

$$\mathcal{N}^{H-MLA}_{mess} = \log_2(\mathcal{N}_{SN}) \times \sum_{i=1}^{\mathcal{N}_{SN}} \mathcal{R}_{SN_i} + \sum_{i=1}^{k} (\log_2(\mathcal{N}_{P_i})) \times \sum_{j=1}^{\mathcal{N}_{P_i}} \mathcal{R}_{P_j} \tag{3.38}$$

Dans ce qui suit, nous déterminons le nombre total de messages générés pour chaque méthode de recherche (itérative, récursive et semi-récursive) dans *H-MLA*.

Recherche par les méthodes itérative ou récursive : Dans les méthodes itérative et récursive, le nombre de messages générés par un noeud est :

- dans chaque sous niveau i, tous les noeuds (en nombre de $\mathcal{N}_{P_i}$) participent à la distribution. Ce qui donne :

$$\mathcal{R}_{P_j} = 2 \times (\mathcal{N}_{P_i} - 1)$$

- dans le niveau supérieur, tous les noeuds (au nombre de $\mathcal{N}_{SN}$) participent à la distribution. Ceci donne :

$$\mathcal{R}_{SN_i} = 2 \times (\mathcal{N}_{SN} - 1)$$

Le nombre total de messages générés par tous les noeuds est donc :

- dans chaque sous niveau i :

$$\sum_{j=1}^{\mathcal{N}_{P_i}} \mathcal{R}_{P_j} = 2 \times \mathcal{N}_{P_i} \times (\mathcal{N}_{P_i} - 1)$$

- dans le niveau supérieur :

$$\sum_{i=1}^{\mathcal{N}_{SN}} \mathcal{R}_{SN_i} = 2 \times \mathcal{N}_{SN} \times (\mathcal{N}_{SN} - 1)$$

D'où :

- le nombre total de messages générés dans le niveau supérieur est :

$$\mathcal{N}_{mess/Nsup}^{H-MLA}(Ite/Rec) = 2 \times \mathcal{N}_{SN} \times (\mathcal{N}_{SN} - 1) \times (\log_2(\mathcal{N}_{SN})) \quad (3.39)$$

- le nombre de messages générés par l'ensemble des sous niveaux est :

$$\sum_{i=1}^{K} \mathcal{N}_{mess/i}^{H-MLA}(Ite/Rec) = 2 \times \sum_{i=1}^{K} \mathcal{N}_{P_i} \times (\mathcal{N}_{P_i} - 1) \times (\log_2(\mathcal{N}_{P_i})) \quad (3.40)$$

Utilisant les équations (3.39) et (3.40), nous calculons le nombre total de messages générés dans *H-MLA* par les méthodes itérative ou récursive par la formule ci-après :

$$\mathcal{N}_{mess}^{H-MLA}(Ite/Rec) = 2 \times \mathcal{N}_{SN} \times (\mathcal{N}_{SN} - 1) \times \log_2(\mathcal{N}_{SN}) + 2 \times \sum_{i=1}^{K} \mathcal{N}_{P_i} \times (\mathcal{N}_{P_i} - 1) \times \log_2(\mathcal{N}_{P_i}) \quad (3.41)$$

Recherche par la méthode semi-récursive : De la même manière, puisque tous les noeuds d'un niveau i respectivement du niveau supérieur participent à la distribution dans leur niveau respectif, en procédant de manière similaire, nous obtenons :

- dans chaque sous-niveau i :

$$\mathcal{R}_{P_j} = \mathcal{N}_{P_i} \quad (3.42)$$

- dans le niveau supérieur :

$$\mathcal{R}_{SN_i} = \mathcal{N}_{SN} \quad (3.43)$$

En injectant les équations (3.43) et (3.42) respectivement dans les équations (3.35) et (3.36), cela donne :

$$\mathcal{N}_{mess/Nsup}^{H-MLA}(SRec) = (\mathcal{N}_{SN})^2 \times (\log_2(\mathcal{N}_{SN})) \quad (3.44)$$

et

$$\sum_{i=1}^{K} \mathcal{N}_{mess/i}^{H-MLA}(SRec) = \sum_{i=1}^{K} (\mathcal{N}_{P_i})^2 \times (\ log_2(\mathcal{N}_{P_i})) \quad (3.45)$$

Nous déduisons ainsi le nombre total de messages générés avec la méthode semi-récursive dans l'architecture *H-MLA* par la formule ci-après :

$$\mathcal{N}_{mess}^{H-MLA}(SRec) = (\mathcal{N}_{SN})^2 \times (\log_2(\mathcal{N}_{SN})) + \sum_{i=1}^{K}(\mathcal{N}_{P_i})^2 \times (\log_2(\mathcal{N}_{P_i})) \quad (3.46)$$

Remarque 3.3 *Dans le cas particulier où nous avons dans tous les sous niveaux, le même nombre de noeuds (noté N_P), nous aurons :*

- Dans les méthodes itérative ou récursive :
$$\sum_{i=1}^{K}\mathcal{N}_{mess/i}^{H-MLA}(Ite/Rec) = 2 \times K \times \mathcal{N}_P \times (\mathcal{N}_P - 1) \times (\log_2(\mathcal{N}_P)) \quad (3.47)$$

Le nombre total de messages générés devient :

$$\mathcal{N}_{mess}^{H-MLA}(Ite/Rec) = 2\times\mathcal{N}_{SN}\times(\mathcal{N}_{SN}-1)\times(\log_2(\mathcal{N}_{SN}))+2\times K\times\mathcal{N}_P\times(\mathcal{N}_P-1)\times(\log_2(\mathcal{N}_P)) \quad (3.48)$$

- Dans la méthode semi-récursive :
$$\sum_{i=1}^{K}\mathcal{N}_{mess/i}^{H-MLA}(SRec) = K \times \mathcal{N}_P^2 \times (\log_2(\mathcal{N}_P)) \quad (3.49)$$

Le nombre de messages total générés est alors :

$$\mathcal{N}_{mess}^{H-MLA}(SRec) = \mathcal{N}_{SN}^2 \times (\log_2(\mathcal{N}_{SN})) + K \times \mathcal{N}_P^2 \times (log_2(\mathcal{N}_P)) \quad (3.50)$$

Remarque 3.4 *Un saut de message dans le réseau de recouvrement correspond à plusieurs sauts dans le réseau physique. Ainsi, nous pouvons minimiser la surcharge du réseau physique en agissant sur la structure du réseau de recouvrement. En effet, en limitant le nombre de messages dans ce dernier, il en sera de même dans le réseau physique.*

Par exemple, nous supposons avoir un réseau physique de N noeuds sur lequel nous souhaitons créer un réseau de recouvrement à plusieurs niveaux avec 5 super noeuds ($\mathcal{N}_{SN} = 5$). De plus, nous supposons que chaque sous niveau dispose de 150 noeuds ($\mathcal{N}_p = 150$). Nous voulons que dans le réseau de recouvrement, le nombre de messages de localisation ne dépasse jamais un million (1000000).

Question : En combien de niveaux pouvons nous organiser le réseau de recouvrement pour que le nombre maximal de messages échangés (lors de la localisation) ne dépasse pas 1000000 ?

Réponse :

- D'après l'équation 3.48, si c'est une des méthodes itérative ou récursive qui est mise en oeuvre, nous aurons $K = 4$ niveaux.
- D'après l'équation 3.50, si par contre c'est la méthode semi-récursive qui est implémentée, le nombre de niveaux est $K = 7$.

Posons :

- $\mathcal{N}_{ij}$ le nombre de sauts entre les noeuds i et j du réseau physique.
- $\mathcal{N}_{maxHops}$ le maximum des nombres $\mathcal{N}_{ij}$ quelque soit les noeuds i et j appartenant au réseau physique, $\mathcal{N}_{maxHops} = Max(\mathcal{N}_{ij})$

Alors, dans le réseau physique, le nombre de messages sera au plus $1000000 \times \mathcal{N}_{maxHops}$

Ceci montre qu'en limitant le nombre de messages dans le réseau de recouvrement (par exemple en agissant sur le nombre de niveaux à mettre en place), nous pouvons limiter le nombre de messages dans le réseau physique.

Dans le tableau suivant, nous montrons le nombre de niveaux K nécessaires pour un réseau de recouvrement pour lequel le nombre maximal de messages est fixé (exemple avec $\mathcal{N}_{SN} = 5$ et $\mathcal{N}_p = 150$).

Nombre maximal de messages	Méthodes itérative ou récursive	Méthode semi-récursive
1000000	$k = 4$	$k = 7$
2000000	$k = 7$	$k = 13$
3000000	$k = 10$	$k = 19$
5000000	$k = 16$	$k = 31$

TABLE 3.2 – Nombre de niveaux en fonction du nombre de messages

3.2 Architecture hiérarchique à plusieurs domaines de recouvrement

Une architecture hiérarchique à plusieurs domaines de recouvrement (*Hierarchical Multi-Domain Architecture : H-MDA*) est représentée à la figure 1.12, page 24.

Dans notre cas, nous utilisons Chord comme table de hachage distribuée aussi bien dans le domaine principal que dans les sous domaines.

- K : nombre de sous domaines.
- $\mathcal{N}_{P_i}$: nombre de noeuds dans le sous domaine i.
- $\mathcal{N}_{SN}$: nombre de super noeuds dans le Domaine Principal (DP).

En procédant de la même manière que précédemment et en nous fondant sur les mêmes assertions de [Bry2008], nous déterminons le nombre de messages échangés avec la formule ci-après :

$$\mathcal{N}_{mess}^{H-MDA} = \mathcal{N}_{mess/DP}^{H-MDA} + \sum_{i=1}^{k} \mathcal{N}_{mess/i}^{H-MDA} \tag{3.51}$$

En appliquant aux différentes méthodes, nous obtenons :

- dans les méthodes itérative ou récursive :

$$\mathcal{N}_{mess}^{H-MDA}(Ite/Rec) = 2\times\mathcal{N}_{SN}\times(\mathcal{N}_{SN}-1)\times(\log_2(\mathcal{N}_{SN}))+2\times K\times\mathcal{N}_P\times(\mathcal{N}_P-1)\times(\log_2(\mathcal{N}_P)) \tag{3.52}$$

- dans la méthode semi-récursive :

$$\mathcal{N}_{mess}^{H-MDA}(SRec) = (\mathcal{N}_{SN})^2 \times (\log_2(\mathcal{N}_{SN})) + K \times (\mathcal{N}_P)^2 \times (\log_2(\mathcal{N}_P)) \tag{3.53}$$

Comme dans l'architecture H-MLA, ces formules permettent de déterminer le nombre de sous domaines k qui peuvent constituer un réseau de recouvrement de type H-MDA de sorte à ne pas polluer la bande passante.

3.3 Comparaison des différentes architectures en fonction des méthodes de localisation

Dans cette section, nous donnons un tableau qui fournit le nombre de messages générés par chacune des architectures en fonction de la méthode de localisation utilisée. Notre objectif est d'observer l'évolution du nombre de messages chaque fois que le nombre de super noeuds varie.

Hypothèses :
Le nombre de noeuds est le même dans chacune des trois architectures. Nous supposons disposer d'un réseau de mille (1000) noeuds (c'est-à-dire que $\mathcal{N} = 1000$).
Nous faisons varier le nombre de super noeuds en respectant le même pas et observons l'évolution du nombre de messages échangés.
•$\mathcal{N}_{SN}$ = nombre de super noeuds
•$\mathcal{N}_{ON/SN}$ = nombre de noeuds ordinaires par super noeud
•K = nombre de sous niveaux (sous domaines)
•$\mathcal{N}_P$ = nombre de noeuds dans chaque niveau K

- Dans *H-SDA* : $\mathcal{N} = \mathcal{N}_{SN} + \mathcal{N}_{SN} \times \mathcal{N}_{ON/SN}$
  En choisissant arbitrairement les valeurs de $\mathcal{N}_{SN}$, nous calculons $\mathcal{N}_{ON/SN}$ et remplaçons dans les équations de $\mathcal{N}_{mess}^{H-SDA}(Ite/Rec)$ et $\mathcal{N}_{mess}^{H-SDA}(SRec)$ (équations 3.33 et 3.34 à la page 47)
- Dans *H-MLA* ou *H-MDA* : $\mathcal{N} = \mathcal{N}_{SN} + K \times \mathcal{N}_P$.
  K et $\mathcal{N}_P$ sont inversement proportionnels, puisque le nombre de noeuds est fixé égal à mille (1000).
  En choisissant arbitrairement les valeurs de $\mathcal{N}_{SN}$, nous calculons $\mathcal{N}_P$ de la même manière que $\mathcal{N}_{ON/SN}$. Ensuite nous déterminons K et remplaçons dans les équations de $\mathcal{N}_{mess}^{H-MLA}(Ite/Rec)$ et $\mathcal{N}_{mess}^{H-MLA}(SRec)$ (équations 3.48 et 3.50 à la page 50). Il faut noter que $\mathcal{N}_{mess}^{H-MLA}$ et $\mathcal{N}_{mess}^{H-MDA}$ ont la même formule.

Les valeurs obtenues sont répertoriées dans le tableau 3.3.

		Nombre de messages générés			
		Architecture H-SDA		Architectures H-MLA ou H-MDA	
		Ite/Rec	SRec	Ite/Rec	SRec
Nombre de super noeuds	40	502128, 5	263415, 2	219076, 4	114152, 6
	80	1204557, 3	616842, 5	152784, 3	80210, 4
	120	2019008, 4	1025514, 2	234220, 8	120579, 2
	160	2929036, 0	1481667, 6	394253, 4	200469, 6
	200	3374484, 9	1702601, 4	618050, 9	312154, 3
	240	5011373, 7	2524685, 7	916198, 5	461516, 9
	280	5564533, 9	2799880, 4	1274683, 9	640759, 3
	320	7426397, 5	3733759, 9	1703322, 3	855403, 5
	360	7115550, 8	3574500, 7	2196254, 2	1101824, 2
	400	8931186, 9	4484481, 1	2760318, 9	1384216, 9

TABLE 3.3 – Compaison des différentes architectures P2PSIP hiérarchiques

- Le tableau 3.3 montre que quel que soit le nombre de super noeuds et quel que soit le type d'architecture, les méthodes itérative ou récursive sont plus

coûteuses en messages que la méthode semi-récursive. Le nombre de messages croît en fonction du nombre de super noeuds. Ceci est dû au fait que plus le nombre de super noeuds augmente, plus l'architecture tend vers un réseau plat car le nombre de noeuds dans le système est fixé. En d'autres termes, le nombre de noeuds ordinaires est réduit puisque certains deviennent super noeuds. Ainsi le nombre de noeuds participant à la distribution devient plus important.

- Par ailleurs, si nous considérons les méthodes récursive ou itérative (colonnes 3 et 5 du tableau 3.3), nous constatons que l'architecture *H-SDA* génère plus de messages que les architectures *H-MLA* et *H-MDA*. Ce constat est valable pour la méthode semi-récursive (colonnes 4 et 6 du même tableau). En effet, dans *H-SDA*, seuls les super noeuds participent à la distribution. Or, en plus de leurs propres messages, les super noeuds doivent supporter tous les messages de leurs noeuds ordinaires. C'est ce qui explique que lorsqu'ils deviennent plus nombreux, le nombre de messages croît plus vite dans *H-SDA* que dans *H-MLA* ou *H-MDA*. Car dans les architectures *H-MLA* ou *H-MDA*, pour chaque noeud, sont pris en compte uniquement ses propres messages.
- Ces deux remarques sur le tableau montrent qu'en utilisant la méthode semi-récursive avec une architecture de type *H-MLA* ou *H-MDA*, la localisation est plus efficace.

Le tableau 3.4 est le résumé des comparaisons faites sur les différents types d'architectures et dont les valeurs sont répertoriées dans le tableau 3.3.

	Itérative ou Récursive	**Semi-récursive**
$H-SDA$	Très coûteux	Coûteux
$H-MLA$ ou $H-MDA$	Coûteux	Moins coûteux

TABLE 3.4 – Résumé des comparaisons des différents types d'architectures

4 Synthèse des études comparatives

Avec le tableau 3.1, nous constatons que le mode prédictif offre de meilleures performances en termes de délai de *handover* que le mode réactif dans les mouvements de noeuds mobiles simples et de points d'accès. Ceci est valable aussi bien dans le *FMIPv6* que dans le *FPMIPv6*. Cela s'explique par l'anticipation qui est faite sur le *handover* dans le cas du mode prédictif. Le noeud mobile commence la recherche d'une *care-of-address* avant même le déclenchement effectif du *handover*, ce qui lui permet de gagner du temps. Lorsque le *handover* commence, il aura déjà obtenu son adresse et il ne lui reste qu'à se connecter à son nouveau point d'attachement.

Toutefois, dans le contexte des coûts de mise à jour, nous observons le contraire. Le mode prédictif présente des performances médiocres comparées à celles du mode réactif. Cela s'explique par les nombreux messages échangés dans le mode prédictif. En effet, avant même que le *handover* ne soit nécessaire, lorsque la force du signal

du noeud mobile décroît jusqu'à une certaine valeur seuil, ce dernier (ou son point d'attachement) envoie un message pour rechercher une *care-of-address*. Le point d'attachement sollicité envoie une *care-of-address* au noeud mobile (ou à son point d'attachement). A son tour, le noeud mobile (respectivement son *MAG*) informe son *HA* (respectivement son *LMA*) de l'obtention d'une nouvelle adresse. Tous ces messages sont échangés avant que le *handover* ne démarre. En d'autres termes, ces messages n'existent pas dans le mode réactif dans lequel l'échange de messages ne commence qu'avec le démarrage effectif du *handover*.

D'autre part, le tableau 3.4 montre qu'en utilisant la méthode semi-récursive dans une architecture à plusieurs niveaux (*H-MLA*, ou *H-MDA*), cela permet d'avoir des coûts de localisation plus réduits.

Ainsi, en combinant les résultats des tableaux 3.1 et 3.4, cela permet de tirer les conclusions suivantes :
Pour mettre en place une solution de gestion de la mobilité dans un réseau *P2PSIP* hiérarchique basé sur IPv6, le concepteur a le choix entre, combiner d'un côté, une architecture à plusieurs couches ou à plusieurs domaines implémentant la méthode semi-récursive et, d'un autre côté,

1. une solution de gestion de la mobilité (*FMIPv6* ou *FPMIPv6*) implémentant le mode prédictif, s'il souhaite mettre la priorité sur des délais de *handover* faibles.
2. une solution de gestion de la mobilité (*FMIPv6* ou *FPMIPv6*) utilisant le mode réactif, si toutefois il souhaite privilégier la réduction des coûts de mise à jour.

5 Conclusion

Dans ce chapitre nous avons mené deux études comparatives : une, sur les modes utilisés dans les solutions de gestion de la mobilité dans IPv6, et une autre sur les architectures pair-à-pair *SIP* hiérarchiques. Ces études nous ont permis de proposer les possibilités de combinaisons à mettre en place pour avoir une solution optimisée selon les cas (faibles délais de *handover*, faibles coûts de mise à jour, faibles coûts de localisation).

Partant de ces conclusions, nous proposons au chapitre suivant, une architecture *P2PSIP* hiérarchique basée sur IPv6 pour gérer la mobilité des noeuds.

Chapitre 4

$HPAMM^{IPv6}$: une architecture pour la gestion de la mobilité dans un réseau P2PSIP hiérarchique basé sur IPv6

1 Introduction

A la suite de l'analyse faite dans les chapitres précédents, notamment dans les chapitres 2 et 3, il devient évident que l'utilisation du protocole IPv6 peut améliorer la gestion de la mobilité dans un système pair-à-pair SIP.

Ainsi dans ce chapitre, nous présentons une nouvelle architecture pour la gestion de la mobilité dans un réseau peer to peer hiérarchique basé sur le protocole IPv6, appelé $HPAMM^{IPv6}$. Les objectifs visés à travers cette architecture sont de gérer la mobilité physique et logique, la mobilité lors de la localisation et lors de la communication et la réduction des temps de *handover*. Pour atteindre ces objectifs, $HPAMM^{IPv6}$ s'appuie sur une structuration hiérarchique des noeuds (*LMAs*, *MAGs* et NMSs), un nouveau système d'adressage IPv6 et un serveur distribué pour la gestion des *handovers* logique et physique.

2 Synthèse des travaux connexes

Dans les réseaux pair-à-pair mobiles, un noeud peut subir deux types de mobilité [Diat2016] : une mobilité physique et/ou une mobilité logique. Dans le premier cas, le noeud se déplace physiquement d'un réseau à un autre entraînant un changement de sa *care-of-address*. Dans le second cas (mobilité logique), le noeud est sur place, mais à cause d'un changement de la topologie du réseau, il doit changer de point d'attachement.
Une solution de gestion de la mobilité via un réseau pair-à-pair doit considérer ces deux types de mobilité afin de prendre en compte les nombreuses exigences qui en ressortent.

D'après [Myu2014], dans un système où des terminaux utilisent à la fois les protocoles *TCP* et *RTP/UDP*, une gestion efficace de la mobilité n'est possible qu'en mettant ensemble le *proxy mobile IPv6 (PMIPv6)* et le protocole SIP. Or, le protocole SIP fonctionne en mode centralisé. Une combinaison du *PMIPv6* et du P2PSIP permet alors non seulement d'avoir une gestion efficace de la mobilité mais aussi et surtout de prendre en compte la tolérance aux pannes des noeuds en évitant la centralisation. Partant de ces éléments et sachant que les architectures pair-à-pair SIP hiérarchiques (H-P2PSIP) sont une optimisation des réseaux P2PSIP, nous avons montré dans [Diat2016] qu'une combinaison des réseaux H-P2PSIP et du *NBMMP* (protocole de gestion de la mobilité basé sur le réseau) permettait d'avoir une solution optimale de gestion de la mobilité. Par ailleurs, le modèle architectural du *NBMMP* est le *PMIPv6*. Le *FPMIPv6* n'est qu'un mode de fonctionnement introduit pour réduire le délai de *handover*. En d'autres termes, une solution optimisée de gestion de la mobilité met ensemble une architecture H-P2PSIP et le *PMIPv6*.

Cependant, à notre connaissance, très peu de travaux ont pris en compte ces deux aspects à la fois. La plupart des solutions s'intéressent à ces notions de manière séparée. Les unes traitant la mobilité dans IPv6 [Abd2016, Chu2011, Kha2014, Kim2016, Lee2010, Kris2016, Leb2016, Mus2012, Par2012, Sor2012, Mor2003, Lei2016, Zoh2014] et les autres la mobilité dans un environnement pair-à-pair [Cos2010, Far2006, Waq2013, Eun2014, Yel2009, Mat2008]. Ainsi, dans la section suivante, nous mettons en exergue les travaux qui se sont intéressés à les combiner, en faisant ressortir les défis relevés dans leurs solutions. L'objectif étant de montrer qu'il reste beaucoup à faire dans ce domaine de la mobilité dans un réseau H-P2PSIP.

Dans [Luo2011], les auteurs ont mis en place un système nommé SARP (*Scalable And Robust PMIPv6*). Leur solution est basée sur Chord et est composée de super noeuds (les *MAGs* et les *LMAs*) qui forment un anneau et des noeuds mobiles attachés aux super noeuds. Leurs *MAGs* et *LMAs* jouent des rôles similaires. Chacun d'eux dispose d'une table dans laquelle sont stockées les informations de ses noeuds mobiles, avec pour chaque noeud mobile son *MAG* d'attachement et son *LMA* d'amorçage.

Lorsqu'un noeud mobile se déplace, son *MAG* d'attachement envoie un message de mise à jour au *LMA* d'amorçage. Le *LMA* met à jour le préfixe réseau de l'adresse du noeud mobile puis répond au *MAG* par un accusé de réception. Les *LMAs* sont donc très sollicités car sont au coeur de tout processus de *handover*. Puisque leur solution utilise Chord, les *MAG* et *LMA* d'un noeud peuvent souvent être géographiquement très éloignés. C'est pourquoi l'envoi de messages de mise à jour et d'accusé de réception entre eux, chaque fois qu'un noeud se déplace, contribue grandement à la surcharge du réseau. Cela constitue une des limites de cette solution. De plus, dans SARP, la mobilité logique durant la communication n'est pas prise en compte. La solution ne considère pas le cas d'un noeud mobile en communication pour lequel le *MAG* d'attachement quitte le réseau. C'est un scénario à prendre en considération puisque dans la *care-of-address* du noeud mobile se trouve le préfixe de sous réseau de son *MAG*.

Une architecture SIP basée sur le *PMIPv6* a été réalisée dans [Myu2014] et permet un *handover* durant la communication. Les auteurs utilisent le *PMIPv6*

pour gérer la mobilité d'un terminal et le protocole SIP pour la mobilité de la session centralisée. Leur solution assure la continuité de la communication même lorsque le mobile traverse différentes technologies d'accès. Cependant ils n'ont pas tenu compte de la possibilité qu'un terminal ayant lancé un message *INVITE* puisse se déplacer vers un autre réseau avant même de recevoir la réponse *200 OK*. C'est-à-dire que le *handover* d'un mobile pendant qu'il cherche à localiser un autre n'est pas considéré.

En combinant une architecture H-P2PSIP avec le *PMIPv6*, les auteurs dans [Song2015] ont mis en place un système à deux niveaux : un premier niveau appelé *home level* géré par un *Home Gateway (HG)* et un second niveau nommé *local level* géré par un *Local Gateway (LG)*. Les *LGs* jouent le rôle des *MAGs* tandis que les *HGs* représentent les *LMAs*. Les *LGs* et les *HGs* sont interconnectés entre eux et les noeuds mobiles leur sont attachés. Lorsqu'un noeud effectue un déplacement, il envoie un message en *multicast* pour rechercher un *local gateway* auquel se connecter. Un *LG* ou un *HG* recevant le message *multicast*, répond en envoyant un message à l'ancien *LG* du noeud mobile qui se chargera de l'acheminer. Cependant, la solution proposée surcharge le réseau car étant non structurée, elle utilise le routage par inondation. En effet, le message *multicast* est reçu par plusieurs *LGs* et *HGs* qui répondront chacun pendant que le noeud mobile change de position. A l'inverse, leur solution prend en compte le déplacement d'un noeud durant la communication. Comme tous les autres travaux du domaine, ils n'ont pas géré le *handover* durant la phase de localisation.

Dans [Ana2015] par contre, la mobilité est gérée pour un groupe de noeuds mobiles. En effet, leur solution permet à un ensemble de noeuds mobiles connectés à un point d'accès de se déplacer ensemble. Lorsqu'un point d'accès passe d'un *MAG* à un autre avec tous les noeuds mobiles qui lui sont rattachés, les mises à jour concernent l'ensemble des noeuds. Chaque noeud du groupe doit mettre à jour son préfixe de sous réseau en tenant compte de son nouveau *MAG*. Beaucoup de messages de mise à jour sont échangés entre le *LMA* et le *MAG* du groupe. Ce qui contribue à surcharger la bande passante notamment pour les mouvements inter *LMAs*.

Dans le même ordre d'idées, une architecture, basée sur le *PMIPv6* pour une mise à jour groupée d'un ensemble d'objets connectés, a été conçue dans [Gua2016] afin de réduire les coûts de signalisation. Leur solution ne s'intéresse qu'au déplacement physique d'un noeud.

Le résumé de ces travaux connexes est illustré dans le tableau 4.1. Il met en évidence, les limites des solutions proposées dans le domaine, en mettant un accent particulier sur la latence, la surcharge du réseau et le *handover* lors des phases de localisation et de communication.

Remarque : Dans le tableau 4.1, *PA* signifie point d'accès.

Le tableau 4.1 montre que plusieurs défis restent encore non résolus. Le *handover* durant la localisation n'est nullement pris en compte. Il est indispensable pour des applications temps-réel. En effet, la prise en charge de ce *handover*, devrait éviter

	"An approach for building scalable proxy mobile ipv6 domains" [Luo2011]	"Performance Analysis of A Novel Inter-Networking Architecture" [Myu2014]	"On pmipv6-based mobility support for HP2P-SIP" [Song2015]	"Efficient Mobility Management Signalling in Network Mobility Supported PMIPV6" [Ana2015]	"The PMIPv6-Based Group Binding Update for IoT Devices" [Gua2016]
Mobilité d'un noeud	pris en compte	pris en compte	pris en compte	pris en compte	pris en compte
Mobilité d'un ensemble de noeuds	**Non pris en compte**	**Non pris en compte**	**Non pris en compte**	pris en compte	pris en compte
Handover durant la communication	**Non pris en compte**	pris en compte	pris en compte	**Non pris en compte**	**Non pris en compte**
Handover durant la localisation	**Non pris en compte**	**Non pris en compte**	**Non pris en compte**	**Non pris en compte**	**Non pris en compte**
latence du *handover*	**Non résolu**	**Non résolu**	**résolu**	- résolu pour le déplacement d'un noeud simple **- Non résolu pour le déplacement d'un PA**	**Non résolu**
coûts de signalisation	**Non résolu**	**Non résolu**	**Non résolu**	- résolu pour le déplacement d'un noeud simple **- Non résolu pour le déplacement d'un PA**	**Non résolu**
Perte de paquets	**Non résolu**	**Non résolu**	**Non résolu**	- résolu pour le déplacement d'un noeud simple **- Non résolu pour le déplacement d'un PA**	**Non résolu**
Surcharge des éléments centraux	**Non résolu**	**Non résolu**	**Non résolu**	- résolu pour le déplacement d'un noeud simple **- Non résolu pour le déplacement d'un PA**	**Non résolu**

TABLE 4.1 – Résumé de l'état de mise en oeuvre des exigences dans la mobilité à travers un réseau H-P2PSIP basé sur le PMIPv6

l'envoi de nouvelles demandes de communication.

D'autre part, les défis tels que le délai de latence, les coûts de signalisation, les pertes de paquets, la surcharge des éléments jouant un rôle central (*LMA*, *MAG*, *HA*, *FA*) sont toujours non résolus. En effet, dans tous les travaux présentés [Song2015, Ana2015, Gua2016, Luo2011, Myu2014], le déplacement d'un noeud mobile simple ou d'un groupe de noeuds s'accompagne toujours de plusieurs messages de mise à jour. Ce qui augmente les coûts de signalisation et par ricochet le délai de latence. De même, toutes les opérations de mise à jour faites au niveau des *LMAs* et *MAGs* contribuent à les surcharger. La solution est de trouver un moyen de les désengorger ce qui leur permettra d'être plus performants dans le traitement des requêtes, entraînant ainsi une réduction du délai d'attente et par le fait même une réduction de la perte de paquets.

3 Structuration de $HPAMM^{IPv6}$

Dans cette section, nous présentons le rôle de chaque type de noeuds et le système d'adressage utilisé pour la gestion de la couche de recouvrement. Nous détaillons la procédure d'intégration des noeuds au réseau, la stratégie de localisation et la gestion de la mobilité.

3.1 Fonctionnement

Dans [Diat2015], nous avons montré que les architectures P2PSIP hiérarchiques sont organisées en multi-domaines [Isla2011], en domaine unique avec une seule couche [Zoels2006] et en domaine unique avec plusieurs couches [Lekuo2007]. Nous avons démontré que les multi-domaines ou les domaines uniques à plusieurs couches sont plus efficaces dans la localisation des noeuds. Par ailleurs, les architectures multi-domaines sont caractérisées par deux types de noeuds : les super noeuds (SN) qui forment le domaine principal et les noeuds mobiles simples (NMS) qui forment les sous domaines. Chaque sous domaine de noeuds mobiles simples est rattaché à un super noeud. A l'inverse, lorsqu'il existe plus de deux catégories de noeuds, les architectures en domaine unique avec plusieurs couches sont mieux adaptées. Car elles distribuent le réseau de recouvrement en autant de niveaux que de types de noeuds. Diané et *al.* ont proposé un tel modèle d'architecture dans [Dian2010] pour la tolérance aux pannes. Le premier niveau est formé par des super noeuds complexes, le deuxième par les super noeuds et le niveau le plus bas est constitué de noeuds ordinaires simples.
Cependant, leur solution ne gère pas les communications mobiles. En d'autres termes, ils n'ont pas pris en compte la possibilité qu'un noeud se déplace pendant qu'il est en communication.
Nous nous inspirons de leur modèle architectural pour construire notre architecture P2PSIP hiérarchique en nous basant sur le *PMIPv6* pour la gestion de la mobilité(Figure 4.1).

Dans notre système, les super noeuds complexes, les super noeuds et les noeuds ordinaires dans [Dian2010] représentent respectivement les *LMAs*, les *MAGs* et les noeuds mobiles simples (NMS). Chaque noeud mobile simple dispose d'un *MAG* et

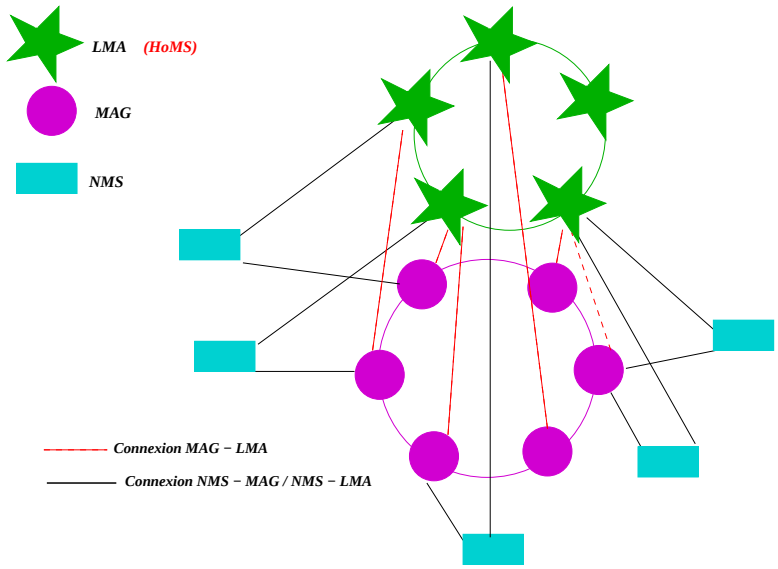


FIGURE 4.1 – Architecture P2PSIP hiérarchique basée sur le PMIPv6

d'un *LMA* responsables auquels il se rattache. De même, chaque *MAG* possède un *LMA* responsable auquel il est rattaché. Les *LMAs* et les *MAGs* sont respectivement organisés en anneau.

3.1.1 Rôle des noeuds

En raison de la hiérarchisation du système, tout noeud peut exécuter toutes les tâches réservées aux noeuds de niveau inférieur.

- NMSs : ils ne font pas de routage. Ils peuvent exécuter des tâches comme :
  - initier des requêtes SIP (*INVITE*, *REGISTER*, *180 RINGING*, *200 OK*, *ACK*, *CANCEL*, *BYE*, etc.) ;
  - recevoir des réponses SIP.
- *MAGs* : en plus des tâches des noeuds mobiles simples, les *MAGs* assurent :
  - le routage des messages SIP (requêtes et réponses) ;
  - la détection des mouvements de leurs noeuds mobiles simples fils et de leurs successeurs. Pour ce faire, chaque *MAG* envoie un message *HELLO* à intervalle de temps régulier de t secondes à tous ses noeuds fils et à ses successeurs. Lorsqu'un noeud ne répond, pas il est considéré comme s'étant déplacé ;
  - lorsqu'un *MAG* détecte le mouvement d'un noeud, il informe son *LMA* qui déclenche le processus de gestion du *handover*.
- *LMAs* : en plus des tâches des *MAGs* et des noeuds mobiles simples, les *LMAs* :
  - maintiennent une base de données de tous ses *MAGs* et NMSs fils ;
  - détiennent le *Handover Management Server (HoMS)* qui gère la mobilité.

3.1.2 Handover Management Server (HoMS)

Pour une bonne gestion de la mobilité dans un réseau P2PSIP, il est nécessaire de considérer à la fois les *handovers* logique et physique. Comme nous le rappelions plus haut, le *handover* logique est le fait qu'un noeud, bien qu'étant sur la même position, change de point d'attachement à cause d'un changement de la topologie

du réseau. Tandis que le *handover* physique est le fait qu'un noeud quitte son réseau physique vers un autre. Ce changement de réseau entraîne des changements d'adresse IP notamment la *care-of-address* du noeud. Pour prendre en compte ces deux types de *handover* (physique et logique) et gérer surtout le *handover* durant la localisation, nous proposons un serveur distribué nommé *Handover Management Server (HoMS)*. Il est distribué à tous les *LMAs*. Pour bien remplir sa mission, nous l'équipons d'une table appelée *Communicating and Locating Node (CLN) table* et contenant tous les noeuds en communication et ceux ayant lancé un message SIP *INVITE* mais n'ayant pas encore reçu de réponse de la part de leur correspondant. La table *CLN* se présente ainsi qu'il suit :

Table CLN								
$srcSIP@$	$destSIP@$	$srcIPv6@$	$destIPv6@$	$srcID$	$destID$	$Call-ID$	$srcType$	$destType$

- $srcSIP@$: adresse SIP du noeud source[1] du message *INVITE* ;
- $destSIP@$: adresse SIP du destinataire du message *INVITE* ;
- $srcIPv6@$: adresse IPv6 du noeud source ;
- $destIPv6@$: adresse IPv6 du noeud destinataire ;
- $srcID$: identificateur du noeud source au niveau du réseau de recouvrement ;
- $destID$: identificateur du noeud destinataire au niveau du réseau de recouvrement ;
- $CALL-ID$: identificateur de la session. Il est généré par le protocole SIP. Il identifie la session de manière unique ;
- $srcType$: type (*LMA*, *MAG* ou *NMS*) du noeud source ;
- $destType$: type du noeud destinataire.

Les messages *INVITE* sont envoyés à la couche des *LMAs*. Chaque *LMA* qui reçoit un message *INVITE* ajoute dans la table *CLN* les informations suivantes : $srcSIP@$, $destSIP@$, $srcIPv6@$, $srcID$, $CALL-ID$, et $srcType$. A ce stade, les champs $destIPv6@$, $destID$ et $destType$ sont vides car tant que le destinataire ne répond pas, son type et adresse IP ne sont pas connus. C'est grâce à ces champs que le *LMA* saura qu'un noeud est en phase de localisation ou en communication. Lorsque $destIPv6@$, $destID$ et $destType$ sont vides, cela signifie que le noeud source a lancé un message *INVITE* mais n'a pas encore reçu de réponse (localisation). Lorsque ces champs sont remplis, cela signifie que le noeud est en communication.

Dès qu'un *LMA* détecte le mouvement d'un noeud, il consulte sa table *CLN*.
Si le noeud, en déplacement, y est enregistré en tant que source, alors le *LMA* cherche le champ $destID$. Si ce dernier est vide, le *LMA* initie le processus de gestion du *handover* durant la localisation. Sinon, le *LMA* initie le processus de gestion du *handover* durant la communication.
Par contre, si le noeud en déplacement est dans la table *CLN* en tant que destinataire, cela signifie qu'il est en communication, alors le *LMA* déclenche le processus de gestion du *handover* durant la communication.
Les processus de gestion de handover seront détaillés à la section 4.

1. noeud source : noeud ayant initié le message

Considérant la logique de Chord, où chaque noeud a une liste de $r = log_2(n)$ successeurs [Stoi2001], avec n le nombre de noeuds dans l'anneau, nous définissons :

- $r_{lma} = log_2(N_{lma})$: taille de la liste des successeurs d'un LMA sur son anneau ;
- $r_{mag} = log_2(N_{mag})$: taille de la liste des successeurs d'un MAG sur son anneau ;
- N_{lma} et N_{mag} représentent respectivement le nombre de $LMAs$ et de $MAGs$.

Remarque 3.1 *Le successeur (respectivement prédécesseur) d'un LMA ou d'un MAG est son voisin direct sur son anneau dans le sens des aiguilles d'une montre (respectivement dans le sens contraire des aiguilles d'une montre).*

Dans le système :

- chaque LMA contient dans sa table de routage les adresses IP et les identificateurs de ses r_{lma} premiers successeurs. Il détient des informations sur tous les noeuds fils de niveau inférieur ($MAGs$ et NMSs) connectés à lui ;
- chaque MAG contient dans sa table de routage les adresses IP et les identificateurs de ses r_{mag} premiers successeurs et des r_{lma} premiers successeurs de son LMA responsable. Il détient des informations sur tous les noeuds mobiles simples fils connectés à lui ;
- chaque NMS contient dans sa table de routage les adresses IP et les identificateurs des r_{lma} premiers successeurs de son LMA et des r_{mag} premiers successeurs de son MAG. Ces adresses et identificateurs sont envoyés au NMS respectivement par son LMA et son MAG responsables.

Les identificateurs des noeuds sont obtenus par hachage avec la fonction sha1 comme le montre la section suivante.

3.1.3 Système d'adressage

L'architecture $HPAMM^{IPv6}$ consiste en trois couches de recouvrement (LMA, MAG et NMS) et fonctionne avec une couche physique IPv6. Une adresse IPv6 est composée de trois champs : l'identificateur de site (IDsite), l'identificateur du sous réseau (IDdesousreseau) et l'identificateur de l'interface (IDinterface).

n bits	m bits	$(128 - n - m)$ bits
IDdesite	IDdesousreseau	IDinterface

Ainsi, dans $HPAMM^{IPv6}$, les identificateurs des noeuds sont obtenus grâce à la fonction de hachage sha1 de la manière suivante :

- $LMA - ID = sha1(IDdesite)$;
- $MAG - ID = sha1(IDdesite + IDdesousreseau)$;
- $NMS - ID = sha1(IDdesite + IDdesousreseau + IDinterface)$.

Avec ce système d'adressage, il est possible d'avoir $2^n \times 2^{n+m} \times 2^{128}$ identificateurs.

Remarque 3.2 *La fonction sha1 est appliquée sur les home address et non sur les care-of-address. Ceci permet aux noeuds de conserver des identificateurs fixes et de pouvoir être joignables même lorsque leur home address est filtrée car la localisation se fait grâce aux identificateurs.*

Les $LMAs$ sont des noeuds de grandes capacités de stockage et de vitesses de traitement. Les $MAGs$ sont également des noeuds de grandes capacités (plus petites

que les *LMAs*) et aussi d'assez bonnes vitesses de traitement. Tandis que les noeuds mobiles simples ont de faibles capacités de stockage et de traitement.

3.2 Construction du réseau

Lorsqu'un noeud souhaite intégrer le système, il doit d'abord évaluer sa catégorie (*LMA*, *MAG* ou NMS) . Pour cela, le noeud évalue ses propres performances en termes de capacité de stockage, de capacité mémoire, de vitesse de traitement du processeur, et autres caractéristiques par rapport à des valeurs références. En fonction des performances trouvées, le noeud se déclare *LMA*, *MAG* ou noeud mobile simple. Pour les capacités de stockage, la valeur référence pourrait être de deux Terra octets ($2To$) qui est de nos jours presque la capacité maximale des disques durs [2]. Pour les capacités mémoires, une valeur référence pourrait être $2 \times 8Go$ car la plupart des nouveaux modèles [3] $DDR4$ sont à $8Go$. Quant à la vitesse de traitement du processeur, une valeur de 4 GHz servirait de référence.

Sur une échelle de zéro à cent, si les performances réunies du noeud sont inférieures ou égales à 50%, le noeud se déclare noeud mobile simple. Si elles sont entre 50% exclus et 80% inclus, le noeud se déclare *MAG*. Si par contre, elles dépassent les 80%, le noeud est un *LMA*. Par exemple, un noeud de capacité de stockage de $500Go$ (25%), de capacité mémoire de $8Go$ (50%), de $3GHz$ (75%) de vitesse du processeur et de 50% pour les autres caractéristiques réunies aura des performances globales de 50%. Donc ce noeud se déclarera noeud mobile simple.

Remarque 3.3 *Avec la technologie qui évolue très vite, ces valeurs références peuvent changer au cours du temps. Cela dépendra des capacités des nouveaux matériels qui seront construits. Toutefois, cela n'empêchera le système de fonctionner, il suffira juste de mettre à jour ces valeurs au niveau de l'application en tenant compte des nouvelles capacités de l'heure de chacune des caractéristiques.*

Une fois que le noeud détermine sa catégorie (*LMA*, *MAG* ou NMS), la fonction *sha*1 est alors appliquée sur la *home address* du noeud comme nous l'avons indiqué plus haut pour déterminer son identificateur au niveau du réseau de recouvrement. En raison du rôle primordial joué par les *LMAs*, dès qu'il y a au moins deux noeuds dans le système, il doit nécessairement exister un *LMA* entre eux. En d'autres termes, même si les deux premiers noeuds ne sont pas des *LMAs*, celui des deux ayant les plus grandes capacités se déclare *LMA*. De même, si les deux premiers noeuds sont tous des *LMAs*, celui détenant les capacités les plus faibles devient un *MAG* car à tout moment le caractère hiérarchique du système doit être conservé.

Comme tout réseau *P2P*, $HPAMM^{IPv6}$ est un réseau *peer to peer* construit au dessus d'un réseau physique. Les connexions d'un réseau physique sont souvent générées de manière aléatoire. Il existe plusieurs modèles de génération de graphes aléatoires, le modèle de Molloy et Reed [Mol1998], de Erdös et Rényi [Erd1959], de Albert et Barabási [Bar1999], etc. Le modèle de Molloy et Reed génère un graphe aléatoire ayant une distribution de degré voulu. Il est simple mais ne correspond pas

2. https ://cours-informatique-gratuit.fr/cours/disque-dur-et-ram/

3. https ://www.topachat.com/pages/g_cat_est_micro_puis_page_est_-memoire-pc.html https ://www.topachat.com/pages/produits_cat_est_micro_puis_rubrique_est_wme_ddr4.tml

à la réalité à cause de son faible clustering [Fab2004]. Pour le modèle de Albert et Barabási, les sommets s'ajoutent un à un en s'attachant aux sommets de plus fort degré. Ce qui peut engendrer un problème d'équilibre de charge. Enfin, pour Erdös et Rényi, leur modèle place un nombre prédéfini d'arêtes au hasard sur l'ensemble des arêtes possibles, respectant ainsi la densité du graphe. Tous les sommets ont donc un degré similaire et permet un meilleur équilibrage. Notre solution s'appuie sur un réseau physique pour lequel les connexions sont générées suivant ce modèle.

Après avoir déterminé sa catégorie et s'être appliqué la fonction *sha1*, le noeud arrivant cherche à intégrer le réseau de recouvrement. Il lance un message SIP *REGISTER* de demande de connexion. Le message contient son adresse SIP, son adresse IP, son identificateur (ID) et sa nature (LMA, MAG ou NMS). De l'envoi du message sip *REGISTER* jusqu'à l'intégration du noeud dans le système, plusieurs messages peuvent être échangés. Le tableau suivant donne des détails sur ces messages.

Messages	Champs	Rôle
$REGISTER$	$adresseSIP$, ID, $adresseIP$ $natureduNoeud$	envoyé par tout noeud qui cherche à intégrer le réseau
$NMSLooksForMAG-ForRegisterMSG$	$adresseSIP$, ID, $adresseIP$	envoyé par un NMS qui est dans le système et n'a pas encore de MAG
$NMSLooksForLMA-ForRegisterMSG$	$adresseSIP$, ID, $adresseIP$	envoyé par un NMS qui est dans le système et n'a pas encore de LMA
$MAGLooksForMAG-ForRegisterMSG$	$adresseSIP$, ID, $adresseIP$	envoyé par un MAG qui est dans le système et n'a pas encore intégré l'anneau des $MAGs$
$MAGLooksForLMA-ForRegisterMSG$	$adresseSIP$, ID, $adresseIP$	envoyé par un MAG qui est dans le système et n'est pas encore connecté à un LMA
$LMALooksForLMA-ForRegisterMSG$	$adresseSIP$, ID, $adresseIP$	envoyé par un MAG ou un NMS qui reçoit un message $REGISTER$ de la part d'un LMA cherchant à intégrer le réseau
$UPDATE\_SUCCES-SOR\_LIST\_MSG$	ID, $adresseIP$	Envoyé par un MAG/LMA à ces prédécesseurs lorsqu'il intègre le réseau pour qu'ils mettent à jour la liste de leurs successeurs

TABLE 4.2 – Signification des messages

Le point de départ est l'envoi du message *REGISTER*. Selon la nature du noeud arrivant et selon celle du noeud recevant le message *REGISTER*, plusieurs scénarios sont observés.

Notations :

- d : le degré de connectivité moyen des noeuds ;
- A : noeud arrivant ;
- B : noeud qui reçoit le message de demande de connexion du noeud A ;
- $N_{fils}^{connectes}$: nombre de noeuds de niveau inférieur connectés à un noeud ;

- N_{fils}^{max} : nombre de noeuds maximal de niveau inférieur que peut supporter un noeud.

Nous expliquerons en détails comment obtenir N_{fils}^{max} au chapitre 5 à la section dimensionnement de l'architecture (section 2).

Si le noeud arrivant est un potentiel LMA

Il doit s'insérer dans l'anneau des *LMAs*. Lorsqu'il réussit à s'insérer, il envoie un message à ses prédécesseurs avec un *time-to-live* $TTL = r_{lma}$ qui est décrémenté à chaque saut. Lorsqu'il atteint la valeur zéro (0), le message est supprimé. Ce message contient l'identificateur et l'adresse IP du *LMA* arrivant et permet aux prédécesseurs d'ajouter ce dernier comme leur $(r_{lma} - TTL + 1)^{ieme}$ successeur.
Le noeud arrivant (A) envoie un message sip *REGISTER* en multicast à la recherche d'un *LMA* auquel se connecter. (i) Si un *LMA* reçoit le message, il répond à (A) en précisant ses informations de connexion. (A) décide de se connecter au premier *LMA* ayant répondu. Il envoie un message de confirmation à ce dernier et puis se connecte entre ce *LMA* et son successeur. (A) envoie ensuite un message à ses prédécesseurs avec un $TTL = r_{lma}$ pour que ces derniers mettent à jour la liste de leur r_{lma} successeurs. (ii) Si par contre, un *MAG* ou un NMS reçoit le message *REGISTER* de (A), alors si ce *MAG* ou ce NMS a un *LMA* responsable, il lui transmet le message et les étapes du point (i) sont reprises. Dans le cas où le *MAG* ou le NMS n'a pas de *LMA*, cela veut dire qu'il est le premier noeud du système et que (A) est le premier *LMA* venu. Les deux noeuds se connectent entre eux et (A) devient le *LMA* responsable de ce *MAG* ou de ce NMS. Ces étapes sont décrites de manière formelle dans l'algorithme 1.

Algorithme 1 : Noeud arrivant est un potentiel LMA

```
1 :  (A) envoie un message sip REGISTER en multicast
2 :  Si (B) est un LMA, Alors
3 :  |  (B) répond à (A) et (A) envoie une confirmation à (B)
4 :  |  (A) se connecte entre (B) et son successeur
5 :  |  Successeur de (B) envoie sa liste de successeurs à (A)
     |  /* Pour permettre à (A) de construire sa liste de r_lma successeurs */
6 :  |  TTL = r_lma
7 :  |  (A) envoie à ses prédécesseurs un message UPDATE_SUCCESSOR_LIST_MSG
     |  /*(pour que ces derniers mettent à jour la liste de leur r_lma successeurs)*/
8 :  |  Pour chaque LMA recevant le message UPDATE_SUCCESSOR_LIST_MSG
9 :  |  |  Si (TTL > 0) alors
10 :  |  |  TTL = TTL − 1
11 :  |  |  Il envoie à son tour le message à son prédécesseur
12 :  |  |  Sinon il supprime le message UPDATE_SUCCESSOR_LIST_MSG
13 :  Sinon si (B) est un MAG ou un NMS, alors
14 :  |  Si (B) est connecté à un LMA, alors
15 :  |  |  (B) envoie un message LMALooksForLMAForRegisterMSG à son LMA
16 :  |  Sinon   /*(Cela veut dire qu'il n'y a pas encore de LMA dans le système)*/
17 :  |  |  (A) se connecte dans ce cas à (B)
18 :  |  FinSi
19 :  |  Lorsqu'un LMA reçoit un message LMALooksForLMAForRegisterMSG
20 :  |  |  l'algorithme reprend à la ligne 2
21 :  FinSi
```

Chaque fois qu'un *LMA* intègre le réseau, son successeur lui envoie la liste de ses successeurs pour que le *LMA* arrivant initialise la sienne.

Complexité 1 *: Il s'agit de la complexité en nombre de messages échangés. Nous rappelons que (A) dispose d'un degré de connexion égal à d. Le pire des cas est obtenu lorsque (B) est un MAG ou NMS possédant un LMA (ligne* 14 *de l'algorithme 1). Dans ce cas, en plus des* $2d$ *messages envoyés/reçus par (A) à/de ses voisins (message multicast), il y a le message de confirmation envoyé par (A) au premier LMA ayant répondu, il y a également le message que (A) envoie à ses* r_{lma} *prédécesseurs avec le* $TTL = r_{lma}$ *pour la mise à jour de la liste des successeurs. De plus, il y a le message que le MAG ou le NMS envoie à son LMA pour lui transférer le message* $LMALooksForLMAForRegisterMSG$ *de (A) et le message retour envoyé par le LMA au MAG ou au NMS. Enfin, il y a le message que le successeur de (A) lui envoie pour initialiser la liste de ses successeurs. La complexité est donc de* $(2d + 4 + r_{lma})$ *messages, soit* $O(log(N_{lma}))$.

Si le noeud arrivant est un potentiel MAG

Il doit s'insérer dans l'anneau des *MAGs* et se connecter nécessairement à un *LMA* si au moins il y en a un dans le système. De la même manière qu'un *LMA*, lorsque le *MAG* réussit à s'insérer, il envoie un message à ses prédécesseurs avec un *time-to-live* $TTL = r_{mag}$ qui est décrémenté à chaque saut. Lorsqu'il atteint la valeur zéro (0), le message est supprimé. Ce message contient l'identificateur et l'adresse IP du *MAG* arrivant et permet aux prédécesseurs d'ajouter ce dernier comme leur $(r_{mag} - TTL + 1)^{ieme}$ successeur.

Chaque noeud de niveau supérieur (*LMAs* et *MAGs*), en fonction de ses capacités, possède un nombre maximal de noeuds fils de niveau inférieur qu'il peut supporter, c'est-à-dire qui peuvent se connecter à lui. Ainsi, les *LMAs* ont un nombre maximum de *MAGs* et de noeuds mobiles simples qu'ils peuvent supporter et les *MAGs* ont également un nombre limité de noeuds mobiles simples susceptibles de se connecter à eux.
Le *MAG* arrivant (*A*) envoie un message sip *REGISTER* en *multicast* à la recherche d'un *LMA* et d'un *MAG* auxquels se connecter.

- Si un *LMA* reçoit le message, (i) si le nombre des fils qui lui sont déjà rattachés ($N_{fils}^{connectes}$) est inférieur à N_{fils}^{max}, alors il répond au *MAG* (*A*) en lui envoyant ses informations de connexion. (*A*) décide de se connecter au premier *LMA* ayant répondu et envoie à ce dernier un message de confirmation. Les deux noeuds se connectent entre eux. Le *MAG* doit nécessairement intégrer l'anneau des *MAGs*. Ainsi, lorsqu'il réussit à se connecter à un *LMA*, s'il n'est pas encore inséré dans l'anneau, il envoie à son *LMA* un message intitulé $MAGLooksForMAGForRegisterMSG$. Ce message qui veut dire "un *MAG* cherche un *MAG* pour se connecter" contient les adresses IP et SIP de (*A*), son identificateur, etc. A la réception de ce message, un *LMA* le transfère à un *MAG* connecté à lui s'il en dispose et les étapes du point 2 sont exécutées, sinon il le transfère à son successeur qui fait les mêmes vérifications. (ii) A l'inverse, si le *LMA* a atteint le nombre limite de fils qu'il peut supporter, il envoie le message de (*A*) à son successeur qui fait les mêmes vérifications. Toutes ces étapes sont décrites par l'algorithme 2.

Algorithme 2 : Noeud arrivant est un potentiel MAG (1/3)

```
1 :  Si (B) est un LMA, alors
2 :  |  Si (N_fils^connectes < N_fils^max) alors
3 :  |  |  (B) répond à (A) qui lui envoie un message de confirmation
4 :  |  |  (A) se connecte à (B)
5 :  |  |  Si (A) n'est pas encore connecté à un autre MAG, alors
6 :  |  |  |  (A) envoie à (B) un message MAGLooksForMAGForRegisterMSG
7 :  |  |  FinSi
8 :  |  Sinon
9 :  |  |  (B) transfère le message à son successeur qui fait les mêmes vérifications
10 :  |  FinSi
11 :  |  Lorsque (B) reçoit le message MAGLooksForMAGForRegisterMSG
12 :  |  |  |  iterator it = list.begin()    /* list est la liste des noeuds fils connectés à (B)*/
13 :  |  |  |  Tant que ((it!= list.end()) && (aucun MAG trouvé dans la liste))
14 :  |  |  |  |  it++
15 :  |  |  |  Fin Tant que
16 :  |  |  |  Si (un MAG a été trouvé dans la liste), alors
17 :  |  |  |  |  (B) achemine le message MAGLooksForMAGForRegisterMSG à ce dernier
18 :  |  |  |  |  (A) se connecte entre ce MAG et son successeur
19 :  |  |  |  Sinon
20 :  |  |  |  |  (B) transfère le message à son successeur qui fait les mêmes vérifications
21 :  |  |  |  FinSi
22 :  |  TTL = r_mag
23 :  |  (A) envoie à ses prédécesseurs un message UPDATE_SUCCESSOR_LIST_MSG
      |  /*(pour que ces derniers mettent à jour la liste de leur r_mag successeurs)*/
24 :  |  Pour chaque MAG recevant le message UPDATE_SUCCESSOR_LIST_MSG
25 :  |  |  Si (TTL > 0) alors
26 :  |  |  |  |  TTL = TTL − 1
27 :  |  |  |  |  Il envoie à son tour le message à son prédécesseur
28 :  |  |  FinSi
29 :  FinSi
```

Le pire des cas est obtenu au niveau des lignes 9 et 20 de l'algorithme 2. Dans ces cas, les messages peuvent faire le tour des *LMAs* à la recherche d'un *LMA* possédant un *MAG* fils ($2 \times N_{lma}$ messages), avec N_{lma} le nombre de *LMAs*. A ces messages sont ajoutés les $2d$ messages envoyés/reçus par le *MAG* arrivant à/de ses voisins en *multicast*, le message qu'un *LMA* envoie à son *MAG* lorsque la recherche est fructueuse, la réponse du *MAG* trouvé au *MAG* arrivant, le message de confirmation, les messages envoyés par le successeur et le *LMA* du *MAG* arrivant pour lui fournir la liste de leurs successeurs et le message envoyé par le nouveau venu à ses r_{mag} prédécesseurs. Le nombre de messages est donc $(2 \times N_{lma} + 2d + r_{mag} + 5)$ messages.

- Si un *MAG* reçoit le message de (A) : si le *MAG* possède un *LMA*, il répond à (A) en précisant ses informations de connexion. (A) décide de se connecter au premier *MAG* ayant répondu. Il envoie un message de confirmation à ce dernier et puis se connecte entre ce *MAG* et son successeur. (A) envoie ensuite un message à ses prédécesseurs avec un $TTL = r_{mag}$ pour que ces derniers mettent à jour la liste de leur r_{mag} successeurs. Si par contre le MAG ne possède pas encore de *LMA*, cela veut dire que ces deux noeuds sont les tout premiers du système. Dans ce cas, le noeud de plus grandes capacités devient *LMA*. L'algorithme 3 traduit le déroulement de ce processus.

Algorithme 3 : Noeud arrivant est un potentiel MAG (2/3)

```
1 : Si (B) est un MAG, alors
2 :   Si (B) possède un LMA
3 :     (B) répond à (A) qui lui envoie un message de confirmation
4 :     (A) se connecte entre (B) et son successeur
5 :      TTL = r_mag
6 :      (A) envoie un message UPDATE_SUCCESSOR_LIST_MSG à ses prédécesseurs
7 :      Pour chaque MAG recevant le message UPDATE_SUCCESSOR_LIST_MSG
8 :      Si (TTL > 0) alors
9 :        TTL = TTL − 1
10 :       il envoie le message UPDATE_SUCCESSOR_LIST_MSG à son prédécesseur
11 :     FinSi
12 :     Si (A) n'est pas encore connecté à un LMA
13 :       (A) envoie un message MAGLooksForLMAForRegisterMSG à son successeur
14 :       Le successeur achemine le message à son LMA
15 :     FinSi
16 :     Lorsque le LMA reçoit le message MAGLooksForLMAForRegisterMSG alors
17 :       S'il n'a pas atteint le nombre limite de fils, alors (A) se connecte à ce LMA
18 :       Sinon LMA transmet le message à son successeur
19 :   Sinon   /*(Cela veut dire qu'il n'y a pas encore de LMA dans le système)*/
20 :     Le noeud de plus grandes capacités entre (A) et (B) devient LMA
21 :   FinSi
22 : FinSi
```

Le pire des cas est obtenu au niveau de la ligne 18 de l'algorithme 3. Dans ces cas, le message peut faire le tour des *LMAs* à la recherche d'un *LMA* qui peut connecter le *MAG* arrivant (N_{lma} messages). A ces messages sont ajoutés les $2d$ messages envoyés/reçus par le *MAG* arrivant à/de ses voisins en *multicast*, la réponse du *LMA* trouvé au *MAG* arrivant, le message de confirmation, le messages envoyé par le *LMA* au *MAG* pour lui fournir la liste de ses successeurs et le message envoyé par le nouveau venu à ses r_{mag} prédécesseurs. Le nombre de messages est donc $(N_{lma} + 2d + r_{mag} + 3)$ messages.

- Si par contre un NMS reçoit le message de (A), alors si le NMS a un *MAG*, il lui transmet le message et les étapes de l'algorithme 3 sont reprises. Dans le cas où le NMS n'a pas encore de *MAG*, s'il a un *LMA*, il lui envoie le message et on reprend l'algorithme 2. Si le NMS n'a ni *LMA* ni *MAG*, cela veut dire que ces deux noeuds sont les premiers du système, alors le *MAG* arrivant se déclare *LMA*. Ceci est montré à l'algorithme 4.

Algorithme 4 : Noeud arrivant est un potentiel MAG (3/3)

```
1 : Si (B) est un NMS, alors
2 :   Si (B) est connecté à un MAG, alors
3 :     (B) achemine le message REGISTER à son MAG (on reprend l'algorithme 3)
4 :   Sinon si (B) est connecté à un LMA, alors
5 :     (B) achemine le message REGISTER à son LMA (on reprend l'algorithme 2)
6 :   Sinon   /* Cela veut dire que ces noeuds sont les deux premiers du système */
7 :     (A) devient LMA et se connecte au NMS
8 :   FinSI
9 : FinSi
```

Le pire des cas s'obtient lorsque le NMS n'a pas encore de *MAG* mais possède

un *LMA* (ligne 4 de l'algorithme 4). Dans ce cas, le nombre de messages est le nombre obtenu avec l'algorithme 2 ajouté au message envoyé par le NMS à son *LMA*, c'est-à-dire $(2 \times N_{lma} + 2d + r_{mag} + 6)$ messages.

Complexité 2 *Les algorithmes* 2, 3 *et* 4 *sont exécutés dans le cadre de l'arrivée d'un MAG dans le système. Ainsi, pour ce cas, le pire des cas donne un nombre de messages de* $(2 \times N_{lma} + 2d + r_{mag} + 6)$ *messages, soit* $O(N_{lma} + log(N_{mag}))$.

Si le noeud arrivant est un potentiel NMS

Le NMS doit nécessairement se connecter à un LMA et à un MAG s'il y en a dans le système. Lorsqu'il se connecte, son LMA (respectivement son MAG) lui envoie la liste de ses r_{lma} (respectivement r_{mag}) successeurs. Nous décrivons ces étapes à l'algorithme 5.
Le NMS arrivant (A) envoie un message sip REGISTER en multicast à la recherche d'un LMA et d'un MAG auxquels se connecter.

Algorithme 5 : Noeud arrivant est un potentiel NMS (1/2)

```
1 :  Si (B) est un LMA, alors
2 :  |  Si (N_fils^connectes < N_fils^max), alors
3 :  |  |  (B) répond à (A)
4 :  |  |  (A) envoie à (B) une confirmation et les deux se connectent entre eux
5 :  |  |  Si ((A) n'est pas encore connecté à un MAG) alors
6 :  |  |  |  (A) envoie à (B) un message NMSLooksForMAGForRegisterMSG
7 :  |  |  FinSi
8 :  |  Sinon
9 :  |  |  (B) envoie le message à son successeur
10 :  |  FinSi
11 :  |  Lorsque le LMA reçoit le message NMSLooksForMAGForRegisterMSG
12 :  |  |  iterator it = list.begin()   /* list est la liste des noeuds fils connectés à (B)*/
13 :  |  |  Tant que ((it != list.end()) && (un MAG n'est pas trouvé dans la liste)) faire
14 :  |  |  |  it++
13 :  |  |  Fin tant que
14 :  |  |  Si (un MAG a été trouvé dans la liste), alors
15 :  |  |  |  (B) lui transfère le message NMSLooksForMAGForRegisterMSG
16 :  |  |  |   L'algorithme 6 est exécuté à la ligne 21
17 :  |  |  Sinon
18 :  |  |  |  (B) envoie le message à son successeur qui fait les mêmes vérifications
19 :  |  |  FinSi
20 :  FinSi
```

Le pire des cas est obtenu aux niveau des lignes 9 et 18 car dans ces situations les messages peuvent faire le tour des *LMAs*, d'une part à la recherche d'un *LMA* pouvant le connecter et d'autre part à la recherche d'un *LMA* contenant dans sa liste un *MAG* $(2 \times N_{lma})$. Lorsqu'un tel *LMA* est trouvé, transfère le message au *MAG* qui lui est connecté. Le *MAG* doit vérifier s'il a atteint la limite des noeuds fils qu'il peut supporter. Le cas échéant, il transfère le message à son successeur. Dans ce cas également, le message peut faire le tour des *MAG* à la recherche d'un *MAG* pouvant connecter le NMS (N_{mag}), avec N_{mag} le nombre de *MAG* dans le réseau. A ces messages il faut ajouter les $2d$ messages envoyé et reçus par le NMS arrivant, les réponses envoyées au noeud arrivant par les *LMA* et *MAG* prêts à le connecter, les

messages de confirmation que le NMS leur envoie. Ainsi, le nombre de messages est $(2 \times N_{lma} + N_{mag} + 2d + 5)$.

Algorithme 6 : Noeud arrivant est un potentiel NMS (2/2)

```
21 : Si ((B) est un MAG) alors
22 :   Si (N_fils^connectes < N_fils^max), alors
23 :     (B) répond à (A)
24 :     (A) envoie à (B) une confirmation et les deux se connectent entre eux
25 :     Si ((A) n'est pas encore connecté à un LMA) alors
26 :       (A) envoie à (B) un message NMSLooksForLMAForRegisterMSG
27 :     FinSi
28 :   Sinon
29 :     (B) envoie le message à son successeur
30 :   FinSi
31 :   Lorsque le MAG reçoit le message NMSLooksForLMAForRegisterMSG
32 :     il le transfère à son LMA et on reprend l'algorithme 5
33 : Sinon si (B) est un NMS, alors
34 :   Si (B) est connecté à un LMA, alors
35 :     (B) envoie le message REGISTER à son LMA et on reprend l'algorithme 5
36 :     Si ((A) n'est pas encore connecté à un MAG) alors
37 :       (A) envoie un message NMSLooksForMAGForRegisterMSG à (B)
38 :     FinSi
39 :     Lorsque le NMS reçoit le message NMSLooksForMAGForRegisterMSG
40 :       il le transfère à son MAG et on reprend depuis la ligne 21
41 :   Sinon   /*(Cela veut dire qu'il n"y a pas encore de LMA dans le système)*/
42 :     Le noeud de plus grandes capacités entre (A) et (B) devient LMA
43 :     Les deux se connectent entre eux
44 :   FinSi
45 : FinSi
```

L'algorithme comporte deux grandes parties : de la ligne 21 à la ligne 32 puis de la ligne 33 à la ligne 45. Le pire des cas, dans la première partie, est obtenu au niveau de la ligne 29 (le message peut faire le tour des MAG) et de la ligne 32 (complexité de l'algorithme 5) et le message que le MAG transfère à son LMA. Donc le nombre de messages est $(2 \times N_{lma} + 2 \times N_{mag} + 2d + 6)$.

Pour la deuxième partie, le pire des cas est obtenu au niveau des lignes 35 (complexité de l'algorithme 5), 37 et 40 (complexité de la première partie). A ces messages il faut ajouter les messages que le NMS envoie à (B) pour chercher un MAG lorsqu'il constate qu'il n'est pas connecté à un MAG, mais aussi le message qu'il transfère à son MAG lorsqu'il reçoit le message $NMSLooksForMAGForRegisterMSG$ et celui qu'il transfère à son LMA lorsqu'il reçoit le message $REGISTER$. Ainsi le nombre de messages est $(4 \times N_{lma} + 3 \times N_{mag} + 4d + 14)$.

Complexité 3 *Les algorithmes* 5 *et* 6 *sont exécutés lorsqu'un noeud mobile simple intègre le réseau. Donc le pire des cas, pour l'arrivée d'un noeud mobile simple donne* $(4 \times N_{lma} + 3 \times N_{mag} + 4d + 14)$ *messages, soit* $O(N_{lma} + N_{mag})$.

3.3 Routage et localisation

Dans $HPAMM^{IPv6}$, chaque *MAG* et chaque *NMS* sont nécessairement rattachés à un *LMA* qui les enregistre dans la table de ses noeuds fils. Ainsi, pour la procédure de localisation, lorsqu'un *LMA* souhaite communiquer, il vérifie d'abord dans la table de ses noeuds fils (via l'adresse sip du destinataire) si le destinataire est rattaché à lui, afin de lui envoyer directement l'invitation. Si tel n'est pas le cas, il envoie le message *INVITE* à son successeur. Par ailleurs, s'il s'agit d'un *MAG* ou d'un NMS qui veut entrer en communication, le message *INVITE* est directement envoyé à la couche des *LMAs* car le destinataire est soit un LMA soit un noeud connecté à un *LMA*, et dans ce dernier cas il figure dans la table des noeuds fils connectés à ce *LMA*. Tout *LMA* recevant un message *INVITE*, vérifie si le destinataire est connecté à lui. Si oui, il lui transmet le message. Sinon, il envoie le message à son successeur. De plus, chaque *LMA* qui reçoit un message *INVITE*, remplit sa table *CLN* (*Communicating and Locating Node table*) avec les informations du noeud source et de l'identificateur de l'appel (*call-id*) généré par le protocole SIP. Lorsque le destinataire est trouvé, les messages 180 *RINGING* et 200 *OK* empruntent le même chemin retour vers le noeud source afin de permettre aux *LMAs* de compléter leur table *CLN* avec les informations du destinataire (*destIPv6@*, *destID*, et *destType*). Par contre, le message *ACK* est directement envoyé au destinataire puisqu'un canal est créé entre les deux.

Exemple : Flux de communication entre un noeud source (*sip :awa.samb-@gmail.com*) et un destinataire (*sip : issa.dia@ucad.edu.sn*). Figure 4.2 montre comment sont connectés les noeuds et Figure 4.3 donne les détails sur les messages échangés.

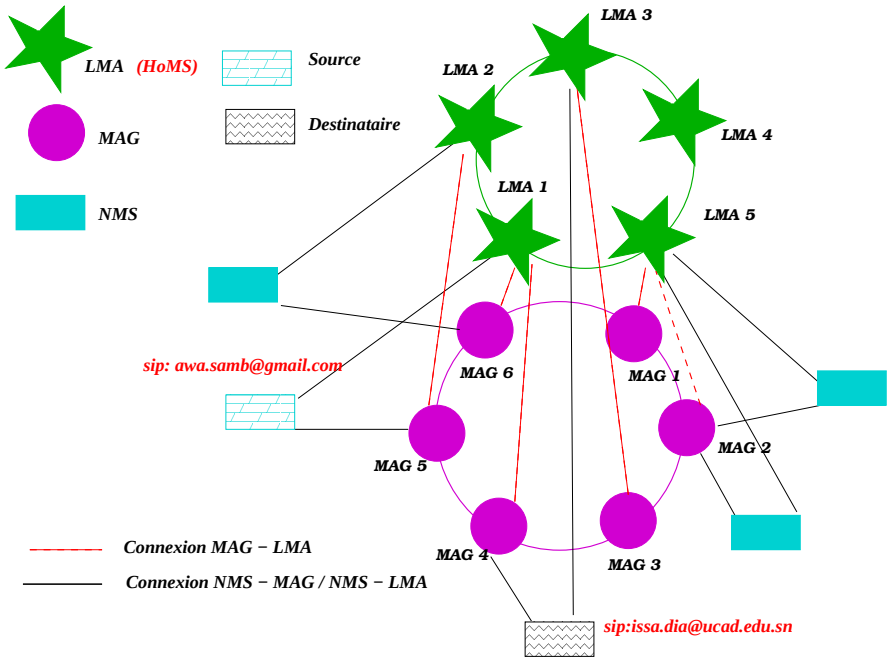


FIGURE 4.2 – Procédure de localisation dans $HPAMM^{IPv6}$

Remarque 3.4 *Lorsqu'un message INVITE revient au LMA qui avait initié l'appel (cela veut dire que le destinataire n'est pas trouvé), celui-ci est supprimé. Par ailleurs, lorsqu'un noeud fils (MAG ou NMS) initie une requête INVITE, son LMA*

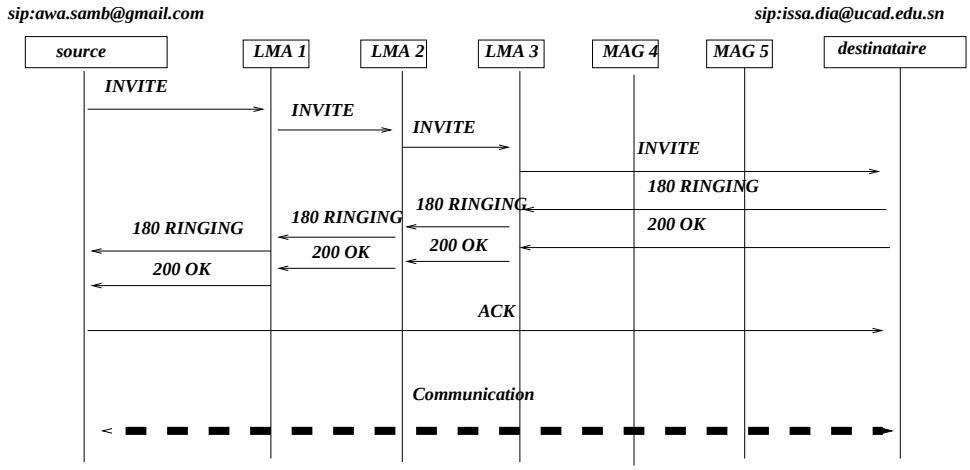


FIGURE 4.3 – Messages SIP échangés dans $HPAMM^{IPv6}$ durant la localisation

estampille le message en y mettant son identificateur. Ainsi, lorsque ce message fait le tour des LMAs et revient au LMA qui l'avait estampillé (cela veut dire que le destinataire n'est pas trouvé), il est supprimé.

Complexité 4 *Lorsqu'un LMA reçoit un message INVITE, il vérifie si le destinataire est rattaché à lui pour lui transmettre le message, sinon le message est acheminé au successeur. Dans le pire des cas, le message fait le tour des LMAs jusqu'à l'avant dernier LMA ($N_{lma} - 1$). Si le destinataire est dans le réseau, il sera trouvé obligatoirement. La réponse prendra le même chemin retour ($N_{lma} - 1$)). Toutefois, le pire des cas est obtenu lorsque le message INVITE a été initié par un MAG ou un NMS. Aux messages précédents, il faut adjoindre les messages "NMS/MAG vers LMA" et "LMA vers NMS/MAG" pour la requête et la réponse . Par conséquent si nous ne considérons que la procédure de localisation (qui commence à l'envoi du message INVITE et se termine lorsque le destinataire est trouvé), alors le nombre de messages est de ($N_{lma} + 1$) dans le pire des cas. Par contre si nous considérons jusqu'au retour de la réponse, le nombre de messages est de ($2 \times N_{lma}$) soit $O(N_{lma})$ dans le pire des cas.*

4 Gestion de la mobilité

Dans cette section, nous décrivons la procédure utilisée dans $HPAMM^{IPv6}$ pour gérer la mobilité logique et la mobilité physique.

4.1 Gestion de la mobilité logique

C'est le cas où le noeud, même lorsqu'il ne change pas de position physique, change de point d'attachement en raison d'un changement de la topologie du réseau.

Gestion de la localisation durant un handover logique

Nous prenons un exemple d'un noeud mobile simple qui envoie un message *INVITE* à un autre noeud du réseau. Le *handover* logique de ce noeud mobile simple durant la localisation est le fait que son *LMA* quitte le réseau avant que le NMS

ne reçoive la réponse 200 *OK* du destinataire. Le NMS doit changer de *LMA* et la réponse doit être envoyée à son nouveau *LMA*.
Exemple (se référer à la Figure 4.2) : L'utilisateur source Awa(*sip :awa.samb@gmail.com*) lance un message *INVITE* à destination de Issa (*sip :issa.dia@ucad.edu.sn*). Avant la réponse de Issa, le *LMA* auquel est rattaché le périphérique de Awa ($LMA1$) quitte le réseau. Le périphérique de Awa doit donc se connecter au successeur de $LMA1$ (c'est-à-dire $LMA2$) et la réponse de Issa doit être envoyée au nouveau *LMA* de Awa, à savoir $LMA2$. Les flux de messages échangés sont illustrés à la figure 4.4.

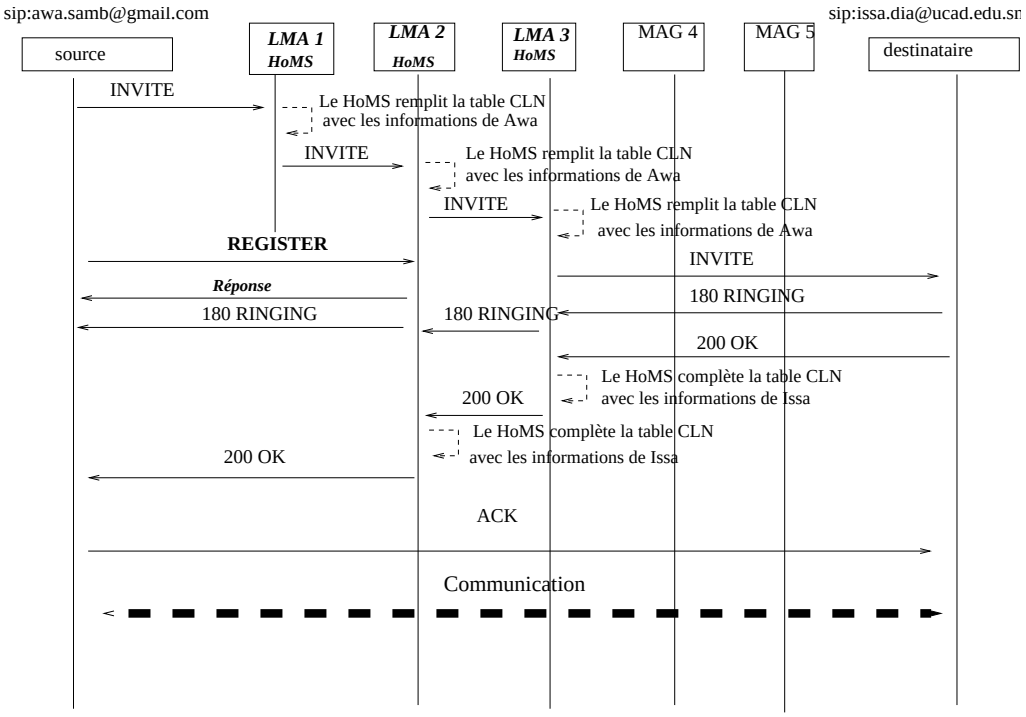


FIGURE 4.4 – Handover logique durant la localisation

Nous avons précisé plus haut, à la section 3.2, que lorsqu'un noeud de niveau inférieur (*MAG* ou NMS) se connecte à un autre de niveau supérieur (*LMA* ou *MAG*), ce dernier lui envoie la liste de ses successeurs. Ainsi, le périphérique de Awa connaît les adresses IP et *IDs* des successeurs de son *LMA*, en l'occurrence $LMA2$, $LMA3$, etc (jusqu'à r_{lma} successeurs). C'est pourquoi, lorsqu'il constate que son *LMA* ($LMA1$) a quitté le réseau (nous montrerons, à la section 3 du chapitre 5, comment un noeud parvient à détecter le départ d'un autre), il envoie un message *REGISTER* aux r_{lma} successeurs de $LMA1$ (c'est-à-dire $LMA2$, $LMA3$, etc.) pour demander une connexion. Les *LMAs* ayant reçu le message *REGISTER* du périphérique de Awa lui répondent et ce dernier décide de ce connecter au premier ayant répondu en envoyant une confirmation. Les deux noeuds se connectent entre eux et $LMA2$ l'ajoute dans la table des noeuds fils connectés à lui.
Puisque Awa a envoyé son message *INVITE* avant que son *LMA* ne quitte le réseau, pendant qu'il envoie le message *REGISTER*, le message *INVITE* poursuit son chemin. Chaque *LMA* recevant le message *INVITE*, son *HoMS* remplit la table *CLN* avec les informations (adresse IP, adresse sip, etc.) du périphérique de Awa. Ainsi, lorsque la réponse 200 *OK* de Issa arrive au niveau de $LMA2$, le périphérique de

Awa est déjà connecté à $LMA2$ qui le trouve dans sa table des noeuds fils connectés. Le $HoMS$ de $LMA2$ complète alors la table CLN avec les informations du périphérique de Issa, et puis achemine le message à Awa.

La figure 4.4, nous n'avons pas montré les messages *REGISTER* envoyés aux autres successeurs de $LMA1$ pour ne pas surcharger la figure. Nous nous sommes limités à montrer un cas en supposant que $LMA2$ est le premier LMA à répondre.

Gestion de la communication durant un handover logique

C'est le cas où, un noeud est en communication pendant que, son point d'attachement, son successeur ou son prédécesseur quitte le réseau. Dans ces cas, lorsque la communication est établie, les noeuds concernés communiquent directement sans passer par leur point d'attachement ni par leur successeur ou prédécesseur. La communication n'est donc pas interrompue. De plus, dans $HPAMM^{IPv6}$, tout noeud est connecté à au moins deux autres. Les $MAGs$ sont rattachés aux $LMAs$ et à d'autres $MAGs$ (prédécesseur et successeur), les $NMSs$ aux $LMAs$ et aux $MAGs$. Lorsqu'un MAG est en communication et que son LMA quitte le réseau, ce MAG reste connecté au réseau grâce à son prédécesseur et à son successeur. Si c'est un NMS qui communique et son LMA quitte le système, grâce à son MAG, ce NMS reste connecté. Dans tous les cas, le noeud reste connecté au réseau et peut relancer une nouvelle demande de connexion pour remplacer le noeud partant.
Si un LMA est en communication, et que son successeur quitte le réseau, grâce à son prédécesseur il reste dans le réseau. Il envoie un message *REGISTER* à son deuxième successeur. Si par contre, c'est son prédécesseur qui quitte le réseau pendant qu'il communique, il reste connecté au réseau grâce à son successeur. Son deuxième prédécesseur lui envoie un message *REGISTER* pour se connecter à lui.

En résumé, le *handover* logique durant la communication n'est pas un problème. Les problèmes apparaissent lors du *handover* physique.

4.2 Gestion de la mobilité physique

La mobilité physique, comme nous l'avons rappelé plus haut, est le fait qu'un noeud quitte son réseau physique vers un autre. Ce changement de réseau entraîne un changement de sa *care-of-address* et donc nécessite des mises à jour au niveau des tables de correspondance et surtout au niveau de la table CLN pour que les réponses qui viendront après que le noeud ait bougé, soient acheminées vers la nouvelle position du noeud en déplacement.

Gestion de la localisation durant un handover physique

Dans ce cas ,il s'agit d'un noeud qui envoie un message *INVITE*, et avant de recevoir la réponse, il change de réseau. Or, le message *INVITE* envoyé contient son ancienne adresse IP. La réponse du destinataire risque d'être envoyée à l'ancienne adresse du noeud source. Ce qui bien évidemment serait un échec car la source vient de se déplacer. Il faut donc mettre en place une bonne stratégie pour rediriger la réponse vers la nouvelle position du noeud source de sorte que l'utilisateur n'ait pas à tenter une nouvelle fois une autre demande de connexion.

Pour ce faire, nous mettons en place un nouveau type de message appelé *Changing Address During Location (CADL)* ou changement d'adresse durant la locali-

sation. Ce message est envoyé à son *LMA* par le noeud source lorsque ce dernier est dans un autre réseau physique. Il contient les éléments suivants : la nouvelle adresse IP du noeud, le *call-ID* de la session, l'adresse sip et la nature du noeud source, l'adresse sip du destinataire. Lorsqu'un *LMA* reçoit un message $CADL$, il vérifie dans la table *CLN* une session dont le *call-id* correspond à celui contenu dans le message $CADL$. En cas de recherche fructueuse, le *LMA*, met à jour la table *CLN* en remplaçant l'ancienne adresse IP du noeud en déplacement par la nouvelle adresse et puis, envoie un nouveau message *INVITE* avec les nouvelles informations du noeud source. Ce message *INVITE* est envoyé au même destinataire. Ainsi, les réponses seront réorientées vers la nouvelle position de la source.

Remarque 4.1 *Nous précisons que le noeud change de réseau physique tout en restant connecté. C'est le cas par exemple de changement d'une zone de couverture à une autre. Le noeud reste connecté et donc son point d'attachement au niveau du réseau de recouvrement ne change pas.*

La Figure 4.5 illustre les flux de messages échangés pour gérer ce type de *handover* (se référer à la Figure 4.2).

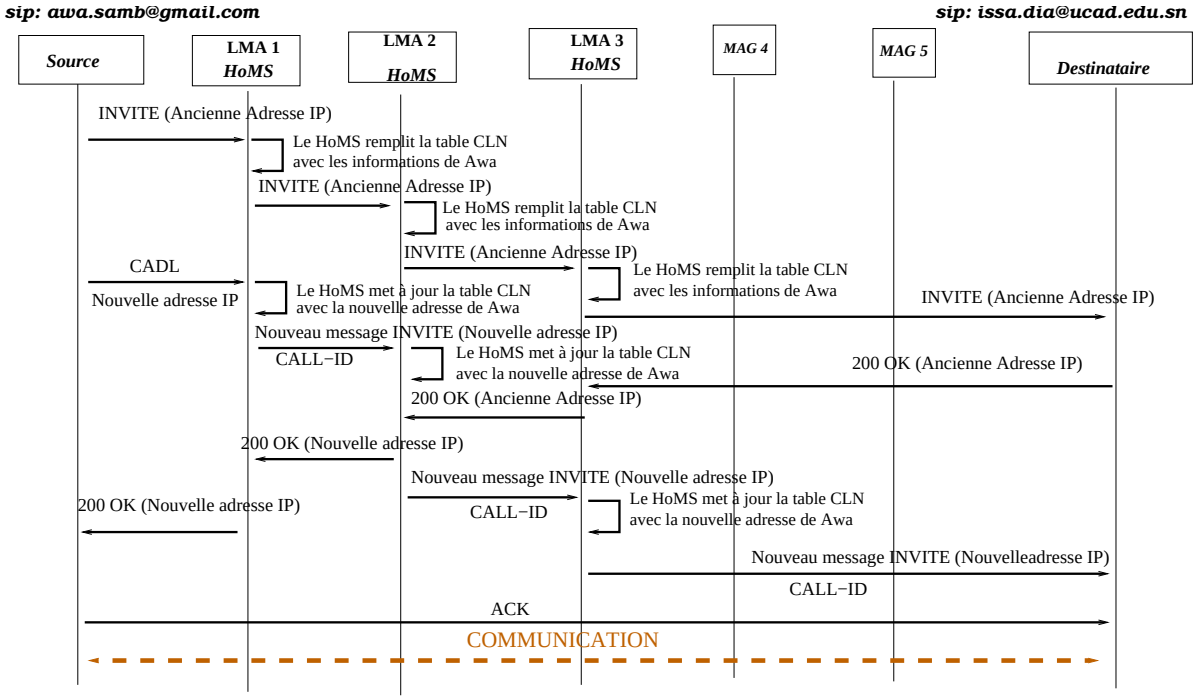


FIGURE 4.5 – Handover physique durant la localisation

La source envoie via son *LMA* ($LMA1$) un message *INVITE* à destination de Issa. Le $HoMS$ de $LMA1$ remplit la table *CLN* et le message est acheminé ensuite au successeur de $LMA1$ ($LMA2$). Pendant que le message poursuit son chemin, le noeud source se déplace vers un autre réseau et obtient une nouvelle *care-of-address*. Cependant, il reste rattaché à $LMA1$ (car en changeant de réseau il ne s'est pas déconnecté). Il envoie un message $CADL$ à $LMA1$. Lorsque $LMA1$ reçoit le message $CADL$ il cherche dans la table *CLN* un *call-id* correspondant à celui dans le message $CADL$. Il le trouve, mais constate que les champs $destIPv6@$, $destID$ et $destType$ sont vides. Cela veut dire que la source est en phase de localisation. Alors,

le $HoMS$ de $LMA1$ met à jour le champs $srcIPv6@$ de la table CLN avec la nouvelle adresse IP se trouvant dans le message $CADL$ et puis, un nouveau message $INVITE$ (avec le même *call-id*) est envoyé au successeur de $LMA1$ (c'est-à-dire $LMA2$) avec la nouvelle adresse du noeud source. Les autres $LMAs$ recevant le message $INVITE$ feront les mises à jour de l'adresse grâce au *call-id*.
A la réception d'une réponse 200 OK dont le *call-id* est le même, tous les $LMAs$ ayant déjà mis à jour l'adresse de la source, mettent dans le champ $destIPv6@$ la nouvelle adresse du noeud et l'achemine. De ce fait, lorsque la réponse arrive à $LMA1$, elle est acheminée à la nouvelle adresse de la source.

Gestion de la communication durant un handover physique

Le *handover* physique durant la communication consiste au déplacement d'un noeud d'un réseau physique vers un autre, tout en étant en communication. Puisque la session a été établie avec l'ancienne adresse du noeud en déplacement, sa nouvelle adresse doit être envoyée à son correspondant [4] pour que les paquets soient redirigés vers sa nouvelle position. Tous les $LMAs$ pour lesquels le serveur $HoMS$ contient le noeud en déplacement dans sa table CLN, doivent mettre à jour la table avec la nouvelle adresse.
Pour le faire, nous concevons un nouveau type de message : *Changing Address During Communication (CADC)* ou changement d'adresse durant la communication. Le message $CADC$ est envoyé à son LMA par un noeud, lorsque ce dernier est dans un autre réseau. Lorsqu'un LMA en communication se déplace, il envoie le message $CADC$ à son successeur. Le message contient la nouvelle adresse IP du noeud, le *call-id* de la session, l'adresse sip et la nature du noeud source, l'adresse sip du destinataire. Lorsqu'un LMA reçoit un message $CADC$, son serveur $HoMS$ vérifie dans la table CLN, une session correspondant au *call-id* contenu dans le message $CADC$. En cas de succès, le $HoMS$ met à jour l'ancienne adresse du noeud en déplacement. La Figure 4.6 montre les flux de messages échangés dans un tel scénario (se référer à la Figure 4.2)

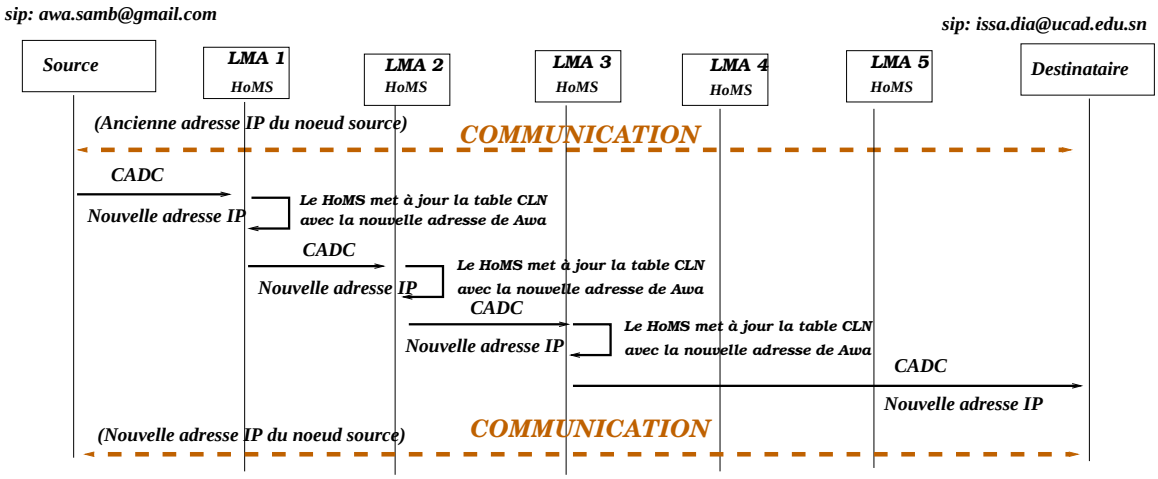


FIGURE 4.6 – Handover physique durant la Communication

Avant le déplacement du noeud, la communication se fait avec l'ancienne adresse

4. Noeud avec lequel on est en communication

IP. Une fois que le message $CADC$ est envoyé au correspondant via les $LMAs$, les nouveaux paquets sont envoyés avec l'adresse actuelle du noeud en déplacement.

5 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle architecture P2PSIP hiérarchique pour gérer les communications mobiles en nous basant sur les principes du protocole IPv6. Le serveur $HoMS$ que nous avons mis en place, nous a permis de gérer les problèmes liés aux *handovers* physique et logique d'un noeud, grâce surtout à la table CLN dont il est doté et aux nouveaux types de messages créés. Nous avons montré comment, à partir des capacités réelles d'un noeud, l'architecture se construit. Étant constituée de trois niveaux, nous avons proposé des fonctionnalités dans les $LMAs$ et les $MAGS$, qui leur permettent d'assurer les exigences inhérentes à la mobilité H-P2PSIP basée sur IPv6.

Dans l'architecture, chaque noeud de niveau inférieur est nécessairement rattaché à un autre de niveau supérieur. Par conséquent, la panne ou la surcharge d'un noeud de niveau supérieur a inévitablement des conséquences sur les noeuds qui lui sont rattachés. Ainsi, dans le chapitre suivant, nous présentons des stratégies de dimensionnement du réseau et ensuite, montrer une solution de gestion de tolérance aux pannes dans $HPAMM^{IPv6}$.

Chapitre 5

Stratégies de dimensionnement et de tolérance aux pannes de $HPAMM^{IPv6}$

1 Introduction

L'architecture $HPAMM^{IPv6}$ fonctionne avec des noeuds qui sont classés suivant leur performances. Selon ses capacités réelles, un noeud a des limites en termes de données à stocker et d'informations à traiter par unité de temps. Cela montre que dans $HPAMM^{IPv6}$, tout noeud de niveau supérieur, en fonction de sa capacité de stockage, dispose d'un nombre maximum de noeuds de niveau inférieur qu'il est en mesure de supporter. Car le nombre de noeuds de niveau inférieur rattachés à un autre de niveau supérieur influe négativement sur le délai de réponse de ce dernier. Plus ce nombre est élevé, plus le noeud de niveau supérieur prendra du temps à traiter les requêtes.
De plus, la panne d'un *LMA* ou d'un *MAG* a des répercutions sur le fonctionnement de son successeur, de son prédécesseur et des noeuds de niveau inférieur qui lui sont rattachés.

Nous précisons qu'un noeud est considéré en panne lorsqu'il ne réagit pas aux messages qui lui sont envoyés. Ceci peut être occasionné par le fait que le noeud est physiquement tombé en panne ou parce qu'il a quitté le réseau.

2 Dimensionnement de l'architecture

Dans cette section, nous nous intéressons à deux types de caractéristiques : les capacités mémoires et les capacités de traitement d'un noeud.

2.1 Dimensionnement en termes de capacités mémoire

Notations :
- $\mathcal{N}_{nms}^{mag}$: nombre de NMSs connectés à un MAG ;
- $\mathcal{N}_{nms}^{lma}$: nombre de NMSs connectés à un LMA ;
- $\mathcal{N}_{mag}^{lma}$: nombre de MAGs connectés à un LMA ;
- $\mathcal{N}_{com/loc}$: nombre de couples de noeuds en communication ou en localisation ;

- X_c : capacité mémoire du noeud de type X ;
$X \in \{NMS, MAG, LMA\}$

Pour les NMS

A la section 3.2 du chapitre 4, nous avons précisé que lorsqu'un noeud mobile simple intègre le réseau, son LMA et son MAG lui envoient respectivement la liste de leurs r_{lma} respectivement r_{mag} successeurs. Ainsi, un noeud mobile simple détient deux tables : une pour indexer les r_{lma} successeurs de son LMA et une autre pour les r_{mag} successeurs de son MAG.
Pour chaque successeur de son LMA, le NMS conserve l'adresse IP (128 bits), l'ID (256 bits), l'ID du tunnel de l'interface via laquelle le NMS contacte le successeur de son LMA au cas où son LMA est en panne (256 bits) ; ce qui donne globalement 640 bits. A ce nombre de bits, il faut ajouter $(1600 + 128 \times s)$ bits comme il est précisé dans [Luo2011] pour stocker un seul successeur de son LMA ; avec s le nombre de bits de l'ID d'interface, c'est-à-dire $(s = 128 - n - m)$ bits, où n et m représentent respectivement les nombres de bits de l'ID de site et de l'ID de sous réseau. Donc pour stocker tous les r_{lma} successeurs de son LMA, un noeud mobile simple a besoin d'un espace mémoire de $[(2240 + 128 \times s) \times r_{lma}]$ bits.
En procédant de la même manière, nous montrons qu'un noeud mobile simple a besoin, pour stocker les r_{mag} successeurs de son MAG, d'un espace mémoire de $[(2240 + 128 \times s) \times r_{mag}]$ bits.

Donc, un NMS a finalement besoin d'un espace mémoire d'au moins :

$$NMS_c = (2240 + 128 \times s) \times (r_{lma} + r_{mag}) \tag{5.1}$$

Selon [Stoi2001], dans un réseau Chord de N noeuds, chaque noeud peut indexer jusqu'à $log_2(N)$ autres noeuds. Dans notre système, nous nous sommes basés sur le système d'adressage utilisé dans Chord, en l'occurrence la fonction $sha1$ appliquée à l'adresse IP. Nous nous basons sur cette théorie pour permettre à un LMA respectivement un MAG de pouvoir indexer sur leur propre anneau jusqu'à $log_2(\mathcal{N}_{lma})$ respectivement $log_2(\mathcal{N}_{mag})$ autres noeuds. Or, les noeuds indexés par un LMA/MAG sur son anneau ne sont rien d'autres que ses r_{lma}/r_{mag} successeurs. Ceci donne $r_{lma} = log_2(\mathcal{N}_{lma})$ et $r_{mag} = log_2(\mathcal{N}_{mag})$.

Pour les MAGs

Lorsqu'un MAG intègre le réseau, son LMA lui envoie la liste de ses r_{lma} successeurs. Un MAG dispose de trois tables pour indexer, ses r_{mag} propres successeurs, les r_{lma} successeurs de son LMA et l'ensemble des noeuds mobiles simples qui lui sont rattachés. De la même manière, un MAG a besoin d'un espace mémoire de $(2240 + 128 \times s) \times (r_{lma} + r_{mag})$ bits pour indexer ses r_{mag} successeurs et r_{lma} successeurs de son LMA.
Pour indexer un NMS, en plus de l'adresse IP, de l'ID du noeud, de l'ID du tunnel de l'interface de connexion et des $(1600+128\times s)$; ce qui donne $(2240+128\times s)$ bits, un MAG a aussi besoin de l'adresse sip (256 bits) de son noeud mobile simple. Donc il lui faut au total un espace mémoire de $(2496 + 128 \times s)$ bits pour l'indexation des informations d'un seul NMS. Pour indexer tous les noeuds mobiles simples rattachés à lui, il faudra au MAG un espace mémoire de $[(2496 + 128 \times s) \times \mathcal{N}_{nms}^{mag}]$ bits.

Par conséquent, pour indexer l'ensemble de ses r_{mag} successeurs, des r_{lma} successeurs de son *LMA* et de tous les NMS rattachés à lui, un *MAG* doit disposer d'une capacité mémoire de

$$MAG_c = (2240 + 128 \times s) \times (r_{lma} + r_{mag}) + (2496 + 128 \times s) \times \mathcal{N}_{nms}^{mag} \quad (5.2)$$

Toutefois, pour les *MAGs* la valeur de s s'obtient par $s = 128 - n = (m + 64)$ bits. En remplaçant s par son expression dans l'équation 5.2, nous obtenons

$$MAG_c = NMS_c + 128 \times m \times (r_{lma} + r_{mag}) + (2496 + 128 \times s) \times \mathcal{N}_{nms}^{mag} \quad (5.3)$$

En considérant des NMS de capacité mémoire de $2Go$ et des *MAGs* de capacité mémoire de $4Go$, l'équation 5.3 montre qu'un *MAG* peut indexer les informations de plus de 1348920 NMS.

Pour les LMAs

Un *LMA* possède quatre tables pour indexer, ses r_{lma} successeurs, l'ensemble des *MAGs* qui lui sont rattachés, l'ensemble des NMS connectés à lui et l'ensemble des noeuds en communication ou en localisation, c'est-à-dire les noeuds se trouvant dans la table *CLN*.
En procédant de la même manière, nous trouvons qu'un *LMA* a besoin de $(2240 + 128 \times s) \times r_{lma}$ bits pour indexer tous ses successeurs, de $[(2496 + 128 \times s) \times \mathcal{N}_{mag}^{lma}]$ bits pour tous ses *MAGs* et de $[(2496 + 128 \times s) \times \mathcal{N}_{nms}^{lma}]$ bits pour l'ensemble de ses NMS. Il faut cependant noter que pour les LMAs, $s = 128$ bits.

Pour les noeuds de la table *CLN*, un *LMA* a besoin de l'adresse IP du noeud source (128 bits), de celle du destinataire (128 bits), la nature du noeud source (256 bits) et du noeud destinataire (256 bits), du *call-id* de l'appel (256 bits).
Il faut donc à un *LMA* un espace mémoire de 1024 bits pour indexer un couple de noeuds en communication ou dont l'un est en phase de localisation.

La capacité mémoire totale dont a besoin un *LMA* pour indexer l'ensemble de toutes ses informations est donc

$$LMA_c = (2240 + 128 \times s) \times r_{lma} + (2496 + 128 \times s) \times (\mathcal{N}_{mag}^{lma} + \mathcal{N}_{nms}^{lma}) + 1024 \times \mathcal{N}_{com/loc} \quad (5.4)$$

Nous rappelons qu'un *MAG* peut indexer jusqu'à plus de 1348920 NMS.

En considérant un système dans lequel :

- Chaque noeud peut entretenir jusqu'à dix (10) communications simultanées comme dans Skype [1] par exemple pour les communications vidéo ;
- Il y un seul *LMA*, un seul *MAG* avec ses 1348920 NMS ;
- Tous les noeuds sont soit en communication ou en localisation.

$\mathcal{N}_{com/loc}$ aura pour valeur

$$\mathcal{N}_{com/loc} = 10 \times [1LMA + 1MAG + 1348920NMS] = 13489220 \quad (5.5)$$

L'équation 5.5 montre que si nous avons un réseau constitué d'un seul *LMA*, d'un seul *MAG* et de 1348920 NMS qui sont tous, soit en communication soit en localisation, le nombre de couples en communication/localisation est de $\mathcal{N}_{com/loc} = 13489220$.

1. http ://www.clubic.com/telecharger/logiciel-voip/skype/aide-astuce-tuto/skype-realiser-une-visioconference-reunissant-jusqu-a-10-participants-14838.html

En considérant des NMS de capacité mémoire de $2Go$, des *MAGs* de $4Go$ de mémoire et des *LMAs* de $6Go$ de mémoire ($2Go+4Go$), l'équation 5.4 montre qu'un *LMA* n'ayant pas de NMS ($\mathcal{N}_{nms}^{lma} = 0$), peut prendre jusqu'à 1998230 *MAGs*.

Chaque fois que le *LMA* n'a pas autant de *MAGs*, le nombre de *MAGs* manquants donne à ce *LMA* la possibilité de prendre un nombre de NMS égal à ($1348920 \times nombre\_de\_MAG\_manquants$). Par exemple si le *LMA* possède un million (1000000) de *MAGs*, il pourra en plus de ces *MAGs*, prendre ($1998230 - 1000000 = 998230$) × 1348920 NMS. Ce qui lui donnera 1000000 *MAGs* et 1346532411600 NMS.

$HPAMM^{IPv6}$ est donc un système qui n'a pas de problèmes de passage à l'échelle.

Remarque 2.1 *Les valeurs que peuvent supporter un MAG et un LMA dépendent des capacités mémoires de ces derniers. Plus la différence des capacités entre les catégories de noeuds est grande, plus un noeud pourra indexer d'autres noeuds de niveau inférieur.En d'autres termes, plus la différence de capacités entre les MAGs et les NMSs est importante, plus les MAGs seront en mesure de prendre des NMSs. De même, plus celle entre les LMAs et les MAGs est élevée, plus les LMAs pourront indexer des MAGs et des NMSs.*

Toutefois, les nombres de *MAGs* et de NMSs que peut supporter un *LMA* sont inversement proportionnels. Plus un *LMA* prendra des *MAGs*, moins il en prendra des NMSs et inversement.

2.2 Dimensionnement en termes de capacités de traitement

D'après [Luo2011], un processeur de $3GHz$ peut vérifier 2048 bits en $100us$ (microsecondes).

Lorsqu'un *MAG* cherche, si un NMS donné est connecté à lui, la vérification se fait sur l'ID (256 bits) du NMS ou sur l'adresse IP (128 bits). Donc elle se fait au plus sur 256 bits. Ainsi un *MAG* dont la vitesse du processeur est de $3GHz$ peut vérifier 80000 NMS en une seconde sur la table des noeuds mobiles simples qui lui sont connectés.
Avec de nos jours le multi-coeur et les traitements parallèles, il est très probable que ce nombre de $80000NMS/s$ soit revu à la hausse, ce qui augmente les performances du système.

Pour les *LMAs*, la recherche de localisation (pour vérifier si un *MAG* ou un NMS est connecté à lui) se fait également sur l'ID ou l'adresse IP du noeud. De la même manière, un *LMA* de $3GHz$ de vitesse du processeur vérifiera 80000 NMSs ou *MAGs* par seconde. Par ailleurs, lors d'un *handover* d'un *MAG* ou d'un NMS, le *LMA* doit vérifier au niveau de la table *CLN*, si le noeud en déplacement n'est pas en communication ou en localisation. Cette vérification se fait sur deux champs : les adresses IP de la source et du destinataire, donc sur 256 bits. Toute la table *CLN* doit être vérifiée car un noeud peut entretenir plusieurs communications à la fois. Plus le nombre de noeuds en communication/localisation est élevé, plus le *LMA* prendra du temps à vérifier la table *CLN*. Ceci montre qu'un *LMA*, pour jouer efficacement son rôle, doit nécessairement disposer d'un processeur dont la vitesse est assez grande. En d'autres termes, la vitesse de $3GHz$ est trop faible pour un *LMA*.

2.3 Stratégies de déploiement

De nos jours, avec les périphériques de communications mobiles qui augmentent à un rythme exponentiel, il est primordial de mettre en place un système qui puisse supporter cette croissance. C'est un des objectifs de $HPAMM^{IPv6}$. A la remarque 2.1, nous avons précisé que, la différence de capacités entre les types de noeuds est déterminante, par rapport au nombre de noeuds que peut supporter un autre. Par ailleurs, l'équation 5.3 nous montre qu'avec une différence de capacités de $2Go$ entre les *MAGs* et les NMS, nous pouvons avoir un nombre assez élevé de NMS par *MAG* (plus de 1348920 NMS). De la même manière, avec l'équation 5.4, nous avons montré qu'avec une différence de capacités de $2Go$ entre les *LMAs* et les *MAGs*, cela permet aux *LMAs* de pouvoir prendre un nombre important de *MAGs* (plus de 1998230 *MAGs*). Dans le même ordre d'idées, l'équation 5.4 nous montre que le nombre de communications ($N_{com/loc}$), qui peuvent être indexées par un *LMA* dans sa table *CLN*, dépend de la capacité mémoire de ce dernier. De plus, en plus des *MAGs*, les *LMAs* doivent supporter également des NMS. Pour toutes ces raisons, il est conseillé d'élargir la différence de capacité entre les *MAGs* et les *LMAs* à $2\times 2Go$. C'est-à-dire, si nous souhaitons mettre en place le système avec des NMS dont les capacités mémoires sont au plus de $4Go$ par exemple, pour jouer efficacement leur rôle, les *MAGs* respectivement les *LMAs* doivent disposer de $6Go$ respectivement $8Go$ de mémoire. En outre, pour les *LMAs*, il est conseillé d'avoir un processeur de grande vitesse ($2 \times 4GHz$ par exemple). Il est claire que ces valeurs sont minimales. Chaque fois que nous pouvons avoir des valeurs élevées pour les *MAGs* et *LMAs*, les performances du systèmes seront davantage améliorées.

3 Stratégie de tolérance aux pannes

Un noeud est considéré être en panne, s'il ne répond pas pendant un temps donné aux messages qui lui sont envoyés. Ceci peut être occasionné, soit par une panne physique soit par un départ du noeud. Selon la nature du noeud en panne, les perturbations au niveau du réseau sont différentes.

A un intervalle de temps régulier de t secondes, chaque *LMA* et chaque *MAG* envoie un message $HELLO$ à son successeur et aux noeuds de niveau inférieur (*MAGs* et NMSs) connectés à lui. A chaque envoi du message $HELLO$, le *LMA* ou le *MAG* relève l'instant du dernier envoi. De même, à chaque réception de la réponse $HELLO$ de son successeur ou d'un des noeuds de niveau inférieur rattachés à lui, il relève l'instant d'arrivée de la dernière réponse. Chaque *LMA* ou chaque *MAG* qui reçoit un message $HELLO$ de son prédécesseur, marque l'instant de réception du dernier $HELLO$ de son prédécesseur. Il en est de même pour les *MAGs* et les NMSs qui marquent l'instant de réception du dernier $HELLO$ de leur *LMA* et/ou de leur *MAG* responsable.

3.1 Détection d'une panne par un LMA

- Lorsqu'un *LMA*, ayant envoyé au préalable un message $HELLO$, reste jusqu'au prochain envoi sans recevoir une réponse de la part d'un noeud, il considère ce dernier comme étant en panne. Si le noeud en panne est un *MAG* ou

un NMS, le *LMA* le supprime de la liste de ses noeuds fils. Si par contre il est son successeur, le *LMA* doit donc chercher un nouveau successeur. Pour ce faire, il envoie un message *LOOKINGFORNEWSUCCESSOR* à tous ses autres successeurs (se trouvant dans la liste de ses r_{lma} successeurs). Ce message est constitué de l'ID du noeud qui cherche un nouveau successeur, de son adresse IP et de sa nature. La différence entre le message *LOOKINGFORNEWSUCCESSOR* et le message $REGISTER$ est que tout noeud qui envoie un message $REGISTER$ à un autre noeud de même nature que lui, cherche à intégrer le réseau. Tandis qu'un noeud qui envoie le message *LOOKINGFORNEWSUCCESSOR* est déjà dans le système. Un *LMA* qui reçoit un message $REGISTER$ de la part d'un autre LMA, insère ce dernier entre lui et son successeur. Alors que, s'il reçoit un message *LOOKINGFORNEWSUCCESSOR* de la part d'un autre *LMA*, il "sait" que ce dernier est déjà dans le système, qu'il a déjà un prédécesseur et qu'il ne cherche qu'un successeur. C'est pourquoi parmi, les *LMAs* qui vont recevoir le message *LOOKINGFORNEWSUCCESSOR*, seuls ceux n'ayant pas de prédécesseur vont répondre car ils sont les seuls à pouvoir satisfaire la demande du *LMA* qui cherche un nouveau successeur ;

- Lorsqu'il obtient son nouveau successeur, le *LMA* met à jour la liste de ses successeurs ;
- Il envoie un message à tous les *MAGs* et NMSs connectés à lui pour une mise à jour ;
- Le nouveau successeur de ce *LMA* envoie à son prédécesseur un message ; $UPDATE\_SUCCESSOR\_LIST\_MSG$ avec un $TTL = r_{lma}$ pour permettre aux prédécesseurs de mettre à jour la liste de leurs successeurs. Le message est envoyé à chaque prédécesseur jusqu'à ce que le TTL s'annule.

3.2 Détection d'une panne par un MAG

Nous rappelons que chaque *MAG* possède la liste des r_{lma} successeurs de son *LMA*.

- Lorsqu'un *MAG*, ayant envoyé au préalable un message $HELLO$, reste jusqu'au prochain envoi sans recevoir une réponse de la part d'un noeud, il considère ce dernier comme étant en panne. Si le noeud en panne est un NMS, le *MAG* le supprime de la liste de ses noeuds fils. Si par contre il est son successeur, le *MAG* lance la recherche d'un nouveau successeur en envoyant un message *LOOKINGFORNEWSUCCESSOR* à tous ses autres successeurs de la liste de ses r_{mag} successeurs ;
- Lorsqu'un *MAG* obtient son nouveau successeur, il met à jour la liste de ses successeurs
- Il envoie un message à tous les NMSs connectés à lui pour une mise à jour ;
- Le nouveau successeur de ce *MAG* envoie à son prédécesseur un message ; $UPDATE\_SUCCESSOR\_LIST\_MSG$ avec un $TTL = r_{mag}$ pour permettre aux prédécesseurs de mettre à jour la liste de leurs successeurs. Le message est envoyé à chaque prédécesseur jusqu'à ce que le TTL s'annule ;
- Lorsqu'un *MAG* reste pendant un temps supérieur à t secondes sans recevoir le message $HELLO$ de son LMA, il le considère en panne. Dans ce cas il

envoie un message *REGISTER* à tous les r_{lma} successeurs de son *LMA* à la recherche d'un nouveau *LMA*. Un *LMA* qui reçoit le message *REGISTER* du *MAG*, s'il n'a pas atteint le nombre maximum de *MAGs* qu'il est en mesure de connecter, envoie une réponse au *MAG*. Le *MAG* envoie une confirmation au premier *LMA* ayant répondu et les deux se connectent entre eux. Le *MAG* vient donc de posséder un nouveau *LMA*.

3.3 Détection d'une panne par un NMS

Le noeud mobile simple n'initie pas de message $HELLO$, il ne fait que répondre à son *LMA* et à son *MAG* qui lui en envoient.

- Lorsqu'il reste pendant un temps supérieur à t secondes, sans recevoir le message $HELLO$ de son *LMA*, il le considère en panne. Dans ce cas il envoie un message *REGISTER* à tous les r_{lma} successeurs de son *LMA* (puisqu'il en possède la liste) pour rechercher un nouveau *LMA*. Un *LMA* qui reçoit le message *REGISTER* du NMS, s'il n'a pas atteint le nombre maximum de NMSs qu'il peut supporter, envoie une réponse à ce NMS qui envoie à son tour une confirmation au premier *LMA* ayant répondu. Les deux noeuds se connectent entre eux et le NMS possède désormais un nouveau *LMA* ;
- De même, lorsqu'un NMS reste pendant un temps supérieur à t secondes, sans recevoir le message $HELLO$ de son *MAG*, il le considère en panne. Il envoie alors un message *REGISTER* à tous les r_{mag} successeurs de son *MAG* afin de chercher un nouveau *MAG*. Chaque *MAG* recevant le message *REGISTER* du NMS, s'il est en mesure de le connecter, envoie une réponse à ce NMS. Ce dernier envoie une confirmation au premier *MAG* à répondre et les deux se connectent. Le NMS dispose ainsi d'un nouveau *MAG*.

4 Coûts de la stratégie de tolérance aux pannes

Lorsqu'un noeud est en panne, les conséquences se font ressentir au niveau de sa couche et des couches inférieures car les noeuds de niveau supérieur possèdent éventuellement des noeuds de niveau inférieur connectés à eux. Plusieurs messages sont échangés pour reconstituer le réseau.

Posons $\mathcal{N}^{mess}_{X/Faulty}$, le nombre de messages échangés pour la reconstitution du réseau lors de la panne d'un noeud de type X.
$X \in \{LMA, MAG, NMS\}$.

4.1 Panne d'un LMA

La panne d'un *LMA* est détectée, d'une part par son prédécesseur (lorsqu'il ne reçoit pas la réponse au message $HELLO$), d'autre part, par tous les *MAGs* et NMS connectés à lui (quand ils restent pendant un temps supérieur à t secondes sans recevoir le message $HELLO$ de leur *LMA*). Dans ce cas, un *LMA* qui constate que son successeur est en panne, envoie à tous les $(r_{lma} - 1)$ successeurs restants un message *LOOKINGFORNEWSUCCESSOR*. Ceux n'ayant pas de prédécesseurs répondent. Au plus, il y en a $(r_{lma} - 1)$; donc $2 \times (r_{lma} - 1)$ messages.

Le *LMA* qui reçoit la réponse de ses successeurs envoie une confirmation (1 message) au premier *LMA* à répondre. Il envoie à ses ($\mathcal{N}_{nms}^{lma}$) NMSs et à ses ($\mathcal{N}_{mag}^{lma}$) *MAGs*, des messages pour qu'ils mettent à jour la liste des successeurs de leur *LMA* $[(\mathcal{N}_{nms}^{lma} + \mathcal{N}_{mag}^{lma})messages]$. Son nouveau successeur envoie un message *UPDATE_SUCCESSOR_LIST_MSG* avec un $TTL = r_{lma}$ aux prédécesseurs (r_{lma} messages). De plus, chacun des ($\mathcal{N}_{nms}^{lma}$) NMSs et des ($\mathcal{N}_{mag}^{lma}$) *MAGs* envoie un message *REGISTER* aux r_{lma} successeurs de leur *LMA* qui vont répondre $[2 \times (\mathcal{N}_{nms}^{lma} + \mathcal{N}_{mag}^{lma}) \times r_{lma}\ messages]$. Chacun des ($\mathcal{N}_{nms}^{lma}$) et ($\mathcal{N}_{mag}^{lma}$) envoie un message de confirmation au premier *LMA* à répondre $[(\mathcal{N}_{nms}^{lma} + \mathcal{N}_{mag}^{lma})\ messages]$. Donc, le nombre de messages générés pour la reconstruction du réseau lors de la panne d'un *LMA* est

$$\mathcal{N}_{LMA/Faulty}^{mess} = 2\times(r_{lma}-1)+\mathcal{N}_{nms}^{lma}+\mathcal{N}_{mag}^{lma}+r_{lma}+2\times(\mathcal{N}_{nms}^{lma}+\mathcal{N}_{mag}^{lma})\times r_{lma}+\mathcal{N}_{nms}^{lma}+\mathcal{N}_{mag}^{lma}+1$$

$$\mathcal{N}_{LMA/Faulty}^{mess} = 2 \times (r_{lma} + 1) \times (\mathcal{N}_{nms}^{lma} + \mathcal{N}_{mag}^{lma}) + 3r_{lma} - 1 \tag{5.6}$$

4.2 Panne d'un MAG

La panne d'un *MAG* est détectée d'une part, par son prédécesseur et son *LMA* lorsqu'ils ne reçoivent pas de réponse au message *HELLO*, d'autre part, par tous les NMS connectés à lui, lorsqu'ils ne reçoivent pas de message *HELLO* de la part de leur *MAG*. Lorsqu'un *LMA* détecte la panne d'un de ses *MAGs*, il le supprime tout simplement de la liste des noeuds connectés ; il n'y a pas d'envoi de messages. D'un autre côté, si un *MAG* constate la panne de son successeur, il envoie à tous les $(r_{mag} - 1)$ successeurs restants un message *LOOKINGFORNEWSUCCESSOR*. Ceux n'ayant pas de prédécesseurs (au plus $(r_{mag} - 1)$) répondent $[2 \times (r_{mag} - 1)]$ messages.
Le *MAG* qui reçoit la réponse de ses successeurs envoie une confirmation (1 message) au premier *MAG* à répondre. Il envoie à ses ($\mathcal{N}_{nms}^{mag}$) NMSs des messages pour qu'ils mettent à jour la liste des successeurs de leur *MAG* ($\mathcal{N}_{nms}^{mag}$) messages. Son nouveau successeur envoie un message *UPDATE_SUCCESSOR_LIST_MSG* avec un $TTL = r_{mag}$ aux prédécesseurs (r_{mag} messages). En outre, chacun des ($\mathcal{N}_{nms}^{mag}$) NMSs envoie un message *REGISTER* aux r_{mag} successeurs de leur *MAG* qui vont répondre ($2 \times \mathcal{N}_{nms}^{mag} \times r_{mag}$ messages). Chacun des ($\mathcal{N}_{nms}^{mag}$) envoie un message de confirmation au premier *MAG* à répondre ($\mathcal{N}_{nms}^{mag}$) messages. Donc, le nombre de messages générés pour la reconstruction du réseau lors de la panne d'un *MAG* est

$$\mathcal{N}_{MAG/Faulty}^{mess} = 2 \times (r_{mag} - 1) + \mathcal{N}_{nms}^{mag} + r_{mag} + 2 \times \mathcal{N}_{nms}^{mag} \times r_{mag} + \mathcal{N}_{nms}^{mag} + 1$$

$$\mathcal{N}_{MAG/Faulty}^{mess} = 2 \times \mathcal{N}_{nms}^{mag} \times (r_{mag} + 1) + 3r_{mag} - 1 \tag{5.7}$$

4.3 Panne d'un NMS

Lors de la panne d'un noeud mobile simple, il n'y a pas d'échange de messages. Son *LMA* et son *MAG*, constatant que leur NMS ne répond plus au messages *HELLO*, le suppriment de leur table.

5 Limites de $HPAMM^{IPV6}$

Nous avons précisé que, pour la stratégie de tolérances aux pannes, chaque *LMA* (respectivement chaque *MAG*) possède une table dans laquelle il indexe ses r_{lma} *LMAs* (respectivement r_{mag} *MAGs*) successeurs. C'est pourquoi lorsqu'un *LMA* (respectivement un *MAG*) est en panne, son prédécesseur envoie à tous les r_{lma} (respectivement r_{mag}) noeuds de la table des successeurs un message *LOOKINGFORNEWSUCCESSOR* afin de reconstituer le réseau et refermer les anneaux des *LMAs* et/ou des *MAGs*. Ceci reste possible tant qu'il ne s'agit pas de r_{lma} *LMAs* (respectivement r_{mag} *MAGs*) **successifs** qui sont en panne. En d'autres termes, lorsque r_{lma} *LMAs* ou r_{mag} *MAGs* successifs tombent en panne, cela empêche le fonctionnement du système car dans ce cas, le message *LOOKINGFORNEWSUCCESSOR* que le prédécesseur envoie sera destiné à des noeuds qui sont tous en panne. Donc les anneaux des *LMAs* ou des *MAGs* ne pourront pas être fermés.

La deuxième limite de notre système, est un défaut d'équilibrage de charge. En effet lorsqu'un noeud de niveau inférieur cherche à se connecter à un autre de niveau supérieur, tant que ce dernier n'a pas atteint le nombre limite des noeuds qu'il peut supporter, il accepte de le connecter alors que, peut être qu'à ce moment même son successeur n'a aucun noeud qui lui est rattaché. C'est un problème auquel nous nous attaquerons dans nos prochains travaux.

6 Conclusion

Le dimensionnement effectué dans ce chapitre, nous a permis de constater que le système est fortement extensible. Nous avons montré que, pour remplir efficacement leur rôle, les noeuds des niveaux supérieurs doivent disposer de capacités mémoires plus grandes que celles des autres des niveaux inférieurs, mais aussi d'une vitesse des processeurs assez grande. Grâce aux listes des successeurs que détient chaque noeud, le système se reconstitue rapidement en cas de panne d'un noeud.

Dans le chapitre suivant, nous allons étudier les performances analytiques et expérimentales de l'architecture proposée.

Chapitre 6

Performances analytiques et expérimentales de $HPAMM^{IPv6}$

1 Introduction

Dans ce chapitre, nous menons une analyse théorique et une validation expérimentale de $HPAMM^{IPv6}$. A ce titre, dans un premier temps, nous proposons une étude comparative de notre solution avec la solution *SARP* pour la gestion de la mobilité physique. Dans un second temps, nous présentons les performances de l'architecture $HPAMM^{IPv6}$ par des simulations, en mettant un accent particulier sur les délais de *handover*, les coûts de maintenance et de localisation, la robustesse et le passage à l'échelle.

2 Performances analytiques sur le délai de handover

Le délai de *handover* est défini suivant deux modes, selon que le noeud est en mouvement logique ou physique. Pour les mouvements logiques, le délai de *handover* est le temps nécessaire pour un noeud de se détacher de son point d'attachement, et de se rattacher à un autre. Tandis que pour les mouvements physiques, il concerne non seulement le temps nécessaire pour un noeud de quitter son réseau physique et de se connecter à un autre, mais aussi le temps nécessaire à l'enregistrement des changements d'adresse au niveau de leur nouveau point d'attachement dans le réseau de recouvrement. Pour déterminer le délai de *handover*, nous considérons l'approche utilisée dans *SARP* pour le *handover* physique.
Nous utilisons les notations suivantes :

- t_{X_Y} : délai moyen entre un noeud de type X et un autre de type Y ;
- D_X : délai de *handover* pour un noeud de type X ;
- $X, Y \in \{LMA, MAG, NMS\}$.

Nous supposons que $t_{X\_Y} = t_{Y\_X}$

2.1 Délais de handover logique

Le délai de *handover* logique consiste d'une part, au temps qu'il faut à un noeud pour constater que son point d'attachement ou son successeur a quitté le réseau. Il s'agit notamment du temps de t seconds que fait le noeud avant de remarquer

que : soit il n'a pas reçu un message $HELLO$, soit son message $HELLO$ n'a pas été répondu. D'autre part, il concerne le temps d'aller-et-retour d'un message envoyé/reçu par un noeud vers/de le successeur du noeud partant, lorsqu'il se rend compte du départ de son successeur ou de son point d'attachement. Dans ce qui suit, nous déterminons le délai de *handover* logique suivant le type de noeud qui quitte le réseau.

Lorsqu'un LMA quitte le réseau

Dans ce cas, le *handover* concerne uniquement ses $r_{lma} = log_2(N_{lma})$ prédécesseurs (qui doivent mettre à jour leur table de successeurs) et les noeuds de niveau inférieur ($MAGs$ et NMSs) qui lui sont rattachés.

Ayant constaté le départ de son successeur (après un temps de t secondes), un LMA envoie un message aux autres successeurs, qui répondent pour signifier son acceptation. Le LMA se connecte au premier ayant répondu $(2 \times t_{lma\_lma})$ et puis le LMA envoie aux autres $(r_{lma} - 1)$ prédécesseurs restants pour les informer de mettre à jour leur table de successeurs $[(r_{lma} - 1) \times t_{lma\_lma}]$. Ainsi,

$$D_{lma} = [log_2(N_{lma}) + 1] \times t_{lma\_lma} \ + \ t$$

De manière similaire, ayant remarqué le départ de son LMA, un MAG respectivement un NMS envoie un message aux successeurs de son LMA (t_{mag_lma}) respectivement (t_{nms_lma}) qui répondent (t_{lma_mag}) respectivement (t_{lma_nms}) pour notifier l'acceptation.
D'après notre hypothèse, $t_{mag\_lma} = t_{lma\_mag}$ et $t_{lma\_nms} = t_{nms\_lma}$. Nous obtenons alors :

$$D_{mag} = 2 \times t_{lma\_mag} + t$$

$$D_{smn} = 2 \times t_{lma\_smn} + t$$

Lorsqu'un MAG quitte le réseau

Dans ce cas également, le *handover* concerne uniquement ses $r_{mag} = log_2(N_{mag})$ prédécesseurs et les NMSs qui lui sont connectés.

Lorsqu'un MAG quitte le réseau, son LMA qui constate le départ, n'a pas besoin d'entrer en phase de *handover*. Il se contente de supprimer le MAG de sa table de noeuds fils. Ceci donne

$$D_{lma} = 0$$

Par ailleurs, lorsqu'un MAG détecte que son successeur ne répond plus au message $HELLO$, après les t secondes, il envoie un message aux autres successeurs qui répondent. Il se connecte au premier MAG à répondre $(2 \times t_{mag\_mag})$ et ensuite envoie aux $(r_{mag} - 1)$ prédécesseurs restants, un message de mise à jour $((r_{mag} - 1) \times t_{mag\_mag})$. Ce qui donne :

$$D_{mag} = [log_2(N_{mag}) + 1] \times t_{mag\_mag} + t$$

Lorsqu'il s'agit d'un NMS qui se rend compte du départ de son MAG (parce que étant resté pendant plus de t secondes sans recevoir le message $HELLO$ du MAG), il envoie un message aux successeurs de son MAG (t_{nms_mag}) qui lui envoient leur

réponse et il se connecte au premier à répondre (t_{mag_nms}).
Or, $t_{mag\_nms} = t_{nms\_mag}$. Donc :

$$D_{nms} = 2 \times t_{mag\_nms} + t$$

Lorsqu'un NMS quitte le réseau

Dans ce cas, aucun noeud n'a besoin d'entrer en phase de *handover*, c'est-à-dire

$$D_{lma} = D_{mag} = D_{nms} = 0$$

2.2 Délais de handover physique

Le délai de *handover* physique consiste d'une part, au temps que met un noeud avant de remarquer qu'il se trouve dans un autre réseau physique. Sans nuire à la généralité, nous considérons ce temps négligeable. D'autre part, il concerne le temps nécessaire à un noeud, pour informer son point d'attachement, ses prédécesseurs et les noeuds de niveau inférieur connectés à lui ; lorsqu'il constate qu'il est dans un autre réseau. Dans ce cas, aucune réponse n'est attendue. Les noeuds recevant le message, mettent à jour leur table avec la *care-of-address* du noeud.

Déplacement d'un LMA d'un réseau physique à un autre

Les mises à jour dues au *handover* physique se font essentiellement au niveau de des $r_{lma} = log_2(N_{lma})$ prédécesseurs et des noeuds de niveau inférieur rattachés au *LMA* en mouvement. Ceci donne :

$$D_{lma} = log_2(N_{lma}) \times t_{lma\_lma} + t_{lma\_mag} + t_{lma\_nms} \tag{6.1}$$

Déplacement d'un MAG d'un réseau physique à un autre

Cette fois les mises à jour dues au *handover* physique concernent ses $r_{mag} = log_2(N_{mag})$ prédécesseurs, son *LMA* et les NMS qui lui sont rattachés. Le délai est donc :

$$D_{mag} = log_2(N_{mag}) \times t_{mag\_mag} + t_{mag\_lma} + t_{mag\_nms} \tag{6.2}$$

Déplacement d'un NMS d'un réseau physique à un autre

Lorsqu'un NMS se déplace, les mises à jour sont effectuées au niveau de son *LMA* et de son *MAG* qui doivent prendre en compte sa nouvelle adresse. Le délai s'obtient par :

$$D_{nms} = t_{nms\_lma} + t_{nms\_mag} \tag{6.3}$$

3 Comparaison analytique de $HPAMM^{IPv6}$ avec SARP

A notre connaissance, *SARP* reste la seule solution qui traite de la mobilité sur IPv6 avec un réseau H-P2PSIP structuré. Une étude plus récente de 2015 notamment [Song2015] a considéré cet aspect mais en utilisant un réseau non structuré . Toutefois, la solution *SARP* a pris en compte uniquement le *handover* physique

d'un NMS, elle n'a pas considéré le *handover* ni d'un *LMA* ni d'un *MAG*. Ce qui limite dans une certaine mesure leur solution, car dans les réseaux peer to peer, les arrivées et les départs de noeuds se passent à tout moment de manière incontrôlée. C'est dans ce cadre que, dans $HPAMM^{IPv6}$, tout noeud peut entrer en phase de *handover*.
Notre étude comparative ne se fera donc uniquement que sur le seul aspect qu'ils ont considéré dans leur étude, à savoir le *handover* d'un NMS. Dans leurs travaux ils ont calculé le délai de *handover* de la manière suivante :

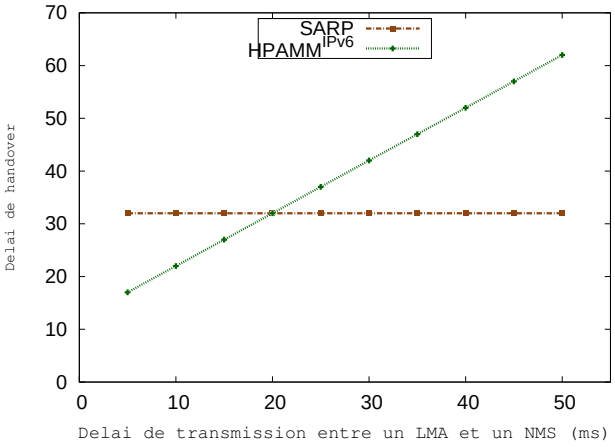
$$D_{nms}(SARP) = 2 \times t_{lma\_mag} + t_{mag\_nms}$$

Dans $HPAMM^{IPv6}$, le délai de *handover* physique d'un NMS est donné par l'équation 6.3, c'est-à-dire

$$D_{nms}(HPAMM^{IPv6}) = t_{nms\_lma} + t_{nms\_mag}$$

Pour comparer les délais de *handover* entre $HPAMM^{IPv6}$ et SARP, nous prenons les mêmes valeurs que celles utilisées dans SARP.

Premièrement : nous fixons les délais de transmission entre d'une part, un *LMA* et un *MAG* ($t_{lma\_mag} = 10ms$) et d'autre part, entre un *MAG* et un NMS ($t_{mag\_nms} = 12ms$). Nous faisons varier le délai entre un *LMA* et un NMS (t_{nms_lma}). La figure 6.1 montre les résultats.



$t_{lma\_mag} = 10ms$ et $t_{mag\_nms} = 12ms$

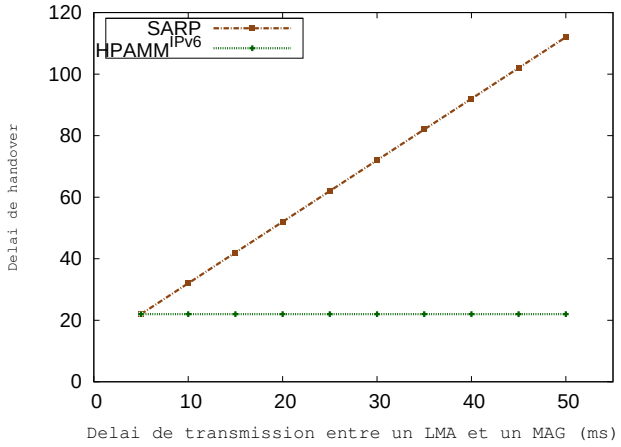
FIGURE 6.1 – Comparaison entre SARP et $HPAMM^{IPv6}$ en fonction du délai de transmission entre un LMA et un NMS

Dans SARP, comme nous l'avons expliqué au chapitre 2, lors d'un *handover* physique d'un noeud mobile simple, tous les messages échangés se passent entre les *MAGs* et les *LMAs* (envoi des messages de mise à jour), mais aussi entre les *MAGs* et les noeuds mobiles simples (le routage des messages de confirmation de mise à jour d'un *MAG* vers un NMS). Dans ce cas, les NMS et les *LMAs* ne communiquent pas directement. C'est ce qui explique d'ailleurs que dans la figure 6.1, le délai de *handover* dans SARP conserve une valeur constante. A l'opposé, dans $HPAMM^{IPv6}$, un NMS communique directement avec son *LMA*. C'est pourquoi, lorsque le délai entre

les *LMAs* et les NMSs croît, SARP offre de meilleures performances. $HPAMM^{IPv6}$ est meilleure dans le cas où le délai de transmission entre les *LMAs* et les NMSs est faible.

Cependant, le fait que dans SARP les *LMAs* et les NMSs ne communiquent pas directement entre eux durant le *handover* d'un NMS est un problème car, si le *MAG* de ce NMS tombe en panne, alors le NMS est déconnecté. C'est ce que nous avons résolu en permettant aux *LMAs* d'une part, de pouvoir faire toutes les tâches que font les *MAGs* et d'autre part, de communiquer directement avec les NMSs.

Deuxièment : nous fixons le délai de transmission entre les *LMAs* et les NMSs ($t_{lma\_nms} = 10ms$) et entre les *MAGs* et les NMSs ($t_{mag\_nms} = 12ms$). Alors nous faisons varier celui entre les *LMAs* et les *MAGs* (t_{lma_mag}). Les résultats obtenus sont illustrés à la figure 6.2.



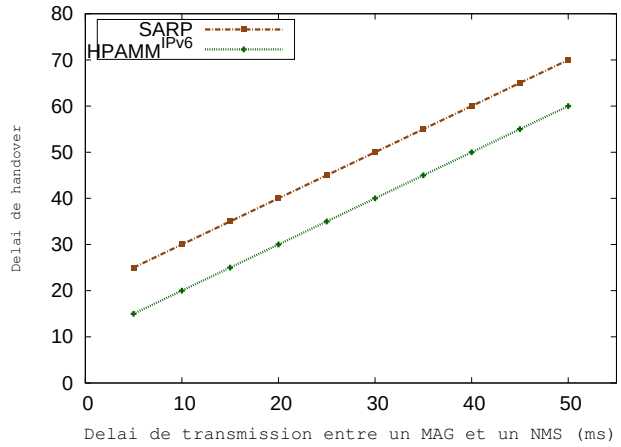
$t_{lma\_nms} = 10ms$ et $t_{mag\_nms} = 12ms$

FIGURE 6.2 – Comparaison entre SARP et $HPAMM^{IPv6}$ en fonction du délai de transmission entre un LMA et un MAG

Dans la figure 6.2, nous observons que $HPAMM^{IPv6}$ est meilleure que SARP lorsque le délai de transmission entre les *MAGs* et les *LMAs* varie. En effet, comme nous l'avons rappelé plus haut, dans SARP les messages sont échangés entre les *MAGs* et les *LMAs*. C'est pourquoi, lorsque le délai entre ces derniers croît, le temps de latence devient plus grand. Alors que, dans $HPAMM^{IPv6}$, durant un *handover* physique d'un NMS, les *MAGs* et les *LMAs* n'ont pas besoin de communiquer entre eux. Les communications se déroulent directement entre le NMS et son *LMA* et aussi entre le NMS et son *MAG*. En d'autres termes, ce qui se passe entre les *LMAs* et les *MAGs* n'affecte pas le *handover* physique du NMS.

Troisièment : Le délai de transmission entre les *LMAs* et les *MAGs* est fixé ($t_{lma\_mag} = 10ms$) et celui entre les *LMAs* et les NMSs ($t_{lma\_nms} = 10ms$). Nous faisons alors varier celui entre les *MAGs* et les NMSs (t_{mag_nms}). Nous obtenons le graphe de la figure 6.3.

Durant le *handover* physique d'un NMS, SARP génère plus de messages que $HPAMM^{IPv6}$. Il s'agit essentiellement d'une part, des aller-et-retour des messages de mise à jour entre les *MAGs* et les *LMAs* et d'autre part, des messages entre le



$t_{lma\_mag} = 10ms$ et $t_{lma\_nms} = 10ms$

FIGURE 6.3 – Comparaison entre SARP et $HPAMM^{IPv6}$ en fonction du délai de transmission entre un MAG et un NMS

MAG et le NMS. Alors que dans $HPAMM^{IPv6}$, les messages entre les *MAGs* et les *LMAs* ne sont pas générés puisque les échanges sont directement menés entre le NMS et son *LMA* et entre NMS et son *MAG*. C'est ce qui explique que $HPAMM^{IPv6}$ offre de meilleures performances.

En résumé, en plus de gérer des aspects de la mobilité non pris en compte par SARP, $HPAMM^{IPv6}$ présente de meilleures performances comparée à SARP sur le délai de *handover* d'un noeud mobile simple.

Après une étude des performances analytiques de $HPAMM^{IPv6}$, nous allons dans la section suivante passer à une évaluation de ses performances par simulation.

4 Evaluation par simulation des performances de $HPAMM^{IPv6}$

Dans cette section, nous évaluons par des simulations les performances de l'architecture $HPAMM^{IPv6}$. Nous présentons l'environnement de simulation, les métriques considérées, les paramètres utilisés et les résultats issus des simulations.

4.1 Environnement de simulation

Afin de mesurer les performances de $HPAMM^{IPv6}$, nous avons mené toutes nos expériences avec le simulateur OMNeT++ [1] (*Objective Modular Network Test-bet in C++*). Il est un simulateur à événements discrets orienté objet. En raison de ses nombreux cas d'utilisation, il est devenu un des simulateurs les plus utilisés. En effet, il peut servir à la modélisation des réseaux de communication filaires et sans fil, à la modélisation des protocoles, de réseaux de files d'attente, des multiprocesseurs et autres systèmes matériels distribués, à la validation d'architectures matérielles, etc.

1. https ://omnetpp.org/doc/omnetpp/SimulationManual.pdf

Nous avons utilisé le logiciel R[2] et la librairie igraph[3] pour générer nos topologies de réseaux physiques sur lesquels repose notre architecture pair-à-pair *SIP* hiérarchique.

4.2 Métriques de mesures

Puisque nous étudions les communications mobiles, nous nous intéressons à mesurer les métriques qui pourraient avoir des influences sur la qualité de la communication. Il s'agit entre autres des coûts de localisation, des coûts de *handover*, des coûts de maintenance, etc.

Coûts de localisation : ils sont définis en termes de délai et en termes de nombre de messages échangés.

Délai de localisation : c'est le temps nécessaire pour un noeud, ayant envoyé un message *INVITE*, de trouver son correspondant dans le réseau.

Nombre de messages de localisation : ils se répartissent en nombre de sauts (depuis la source jusqu'au destinataire) et en nombre de messages global générés depuis l'envoi de l'invitation jusqu'à l'établissement de la session.

Coûts de handover logique : lorsqu'un noeud constate la panne de son voisin, il engage le processus de *handover*. Les coûts de *handover* logique sont composés du nombre de messages échangés lors du *handover* et du délai de*handover*.

Nombre de messages de *handover* : représentent le nombre de messages échangés, depuis qu'un noeud constate la panne de son voisin, jusqu'à ce que ce dernier se reconnecte à nouveau à un autre noeud.

Délai de *handover* : c'est le temps qui sépare l'instant de constatation de la panne et l'instant de connexion à un autre noeud. Ce temps doit être faible pour une bonne qualité de service.

Coûts de maintenance : ils sont composés des coûts engendrés par l'arrivée des noeuds, des coûts de vérification de survie et des coûts dûs à la tolérance aux pannes.

Coûts d'enregistrement : ils concernent le nombre de messages échangés depuis qu'un noeud arrivant envoie un message *REGISTER* jusqu'à ce qu'il réussisse à intégrer le réseau.

Coûts de vérification de survie : ils concernent l'ensemble des messages *HELLO* que les noeuds s'envoient mutuellement pour vérifier si leur voisin est toujours actif.

2. https ://www.r-project.org/
3. http ://igraph.org/r/

Coûts de la tolérance aux pannes : lorsqu'un noeud quitte le réseau (surtout les *LMAs* et les *MAGs*), plusieurs messages sont échangés pour reconstituer les anneaux, mais aussi mettre à jour les listes de successeurs, et permettre au système de continuer à fonctionner. L'ensemble de ces messages échangés depuis la panne d'un noeud jusqu'au rétablissement du réseau constitue les coûts de la tolérance aux pannes. Ils diffèrent selon la nature du noeud en panne.

Nous étudions ces métriques dans deux situations différentes. Dans un premier temps, nous les étudions dans un environnement sans panne. Dans un second temps, nous provoquons des pannes de noeuds et les étudions à nouveau pour observer l'impact des pannes sur ces métriques. Pour ce faire plusieurs paramétrages ont été nécessaires.

4.3 Paramètres de simulation

Dans toutes les simulations effectuées, nous fixons certaines valeurs et faisons varier d'autres. Pour mieux étudier le passage à l'échelle de $HPAMM^{IPv6}$, nous faisons varier la taille du réseau de 100 à 1000. Nous considérons des topologies de réseaux générées suivant le modèle de Erdoï et Réyni [Erd1959]. Pour chaque taille du réseau, nous générons 100 configurations différentes afin de mieux prendre en compte la diversité des scénarios. Nous utilisons également un taux élevé de noeuds en communication (90%) pour étudier l'impact de la surcharge du réseau sur les services de localisation et de *handover*, avec des messages d'une taille maximale de 2048 bits. De même, pour prendre en compte les sessions avec plusieurs participants, nous considérons que chaque noeud peut entretenir jusqu'à 10 communications simultanées.

Pour étudier les coûts de maintenance et leur impact sur les services de localisation et de *handover*, nous prenons un intervalle de temps régulier de $5ms$ au bout duquel les *LMAs* et les *MAGs* envoient à leur successeur et à leurs noeuds fils un message *HELLO*. Pour mieux appréhender les délais de localisation et de *handover*, nous fixons (comme il est fait dans *SARP*) les délais de transmission entre les différents types de noeuds et fixons également à $1us$ (respectivement à $5us$) le temps qu'un *LMA* (respectivement un *MAG*) prend pour traiter une requête.

Pour les capacités des noeuds, nous prenons des valeurs aléatoires pour les capacités de stockage, les capacités mémoires, les vitesses de traitement, le pourcentage des autres capacités réunies et puis calculons le pourcentage global en fonction des valeurs références définies à la section 3.2 du chapitre 4 .

Nous étudions les métriques principalement dans deux scénarios. D'abord, nous considrons un environnement sans fautes. Ensuite, pour étudier la robustesse du système et l'impact des pannes sur les coûts de localisation, les coûts de *handover* et les coûts de maintenance, nous introduisons des pannes de noeuds avec des taux de 15%, 30%, 45%, 60%, 75% et 90% de noeuds en panne.

Nous étudions la précision sur les valeurs recueillies en choisissant un degré de confiance de 99%. Nous traitons les résultats obtenus avec le logiciel R et déterminons les intervalles de confiance [4] pour les différentes métriques.

4. https ://fr.wikihow.com/calculer-un-intervalle-de-confiance

intervalle de confiance = $Moyenne \pm marge\_derreur$ avec

- $Moyenne$: la moyenne des valeurs de la métrique étudiée
- $marge\_derreur = T_z \times \frac{\delta}{\sqrt[2]{n}}$
- T_z : est la valeur sur la table de Z (table des probabilités) correspondant au degré de confiance choisi. Pour le degré de confiance de 99% que nous avons choisi, $T_z = 2,58$
- δ : est l'écart type de la métrique étudiée
- n : est la taille (nombre de noeuds du réseau)

L'ensemble des paramètres utilisés dans la simulation sont représentés dans les tableaux 6.1 et 6.2.

Paramètres	**Valeur**	**Correspondance**
Capacité de stockage	$2To$	100%
Capacité mémoire	$16Go$	100%
Vitesse de traitement	$4Ghz$	100%

TABLE 6.1 – Tableau de correspondance des capacités réelles d'un noeud

Paramètres	**Valeurs**
Nombre de noeuds	[100 - 1000]
Degré de confiance utilisé	99%
Intervalle d'envoi du message HELLO	$5ms$
Temps de simulation	$600s$
Taille maximale de messages	2048 bits
Nombre maximum de communications simultanées que peut entretenir un noeud	10
Délai de transmission entre un LMA et un autre	$0,5ms$
Délai de transmission entre un LMA et un MAG	$1ms$
Délai de transmission entre un MAG et un autre	$1ms$
Délai de transmission entre un LMA et un NMS	$2ms$
Délai de transmission entre un MAG et un NMS	$3ms$
Temps nécessaire à un LMA pour traiter une requête	$1us$
Temps nécessaire à un MAG pour traiter une requête	$5us$
Nombre de noeuds en communication	90% des noeuds
Nombre de noeuds en pannes	15%, 30%, 45%, 60%, 75%, 90% des noeuds
Somme des capacités réelles d'un NMS	$\leq 50\%$
Somme des capacités réelles d'un MAG	]50%, 80%]
Somme des capacités réelles d'un LMA	$> 80\%$

TABLE 6.2 – Paramètres de simulation

4.4 Résultats de simulation

Dans cette section, nous présentons les résultats obtenus lors des différentes simulations.

4.4.1 Environnement sans pannes :

Dans cette partie, nous montrons les résultats des coûts de maintenance. Il faut noter que dans ce cas, il n'y a pas de coûts de tolérance aux pannes.

Coûts de maintenance : Les coûts de maintenance se composent des coûts d'enregistrement (arrivée de noeuds) et des coûts de vérification de survie. Ces derniers représentent l'ensemble des messages *HELLO* échangés par les différents noeuds pour vérifier que leurs voisins sont toujours fonctionnels.

Premier cas : coûts d'enregistrement : Lorsqu'un noeud cherche à intégrer le réseau, il envoie un message *REGISTER*. L'ensemble des messages échangés, depuis l'envoi du message *REGISTER* jusqu'à ce que le noeud intègre le réseau, constitue les coûts d'enregistrement. Ces coûts sont représentés à la figure 6.4.

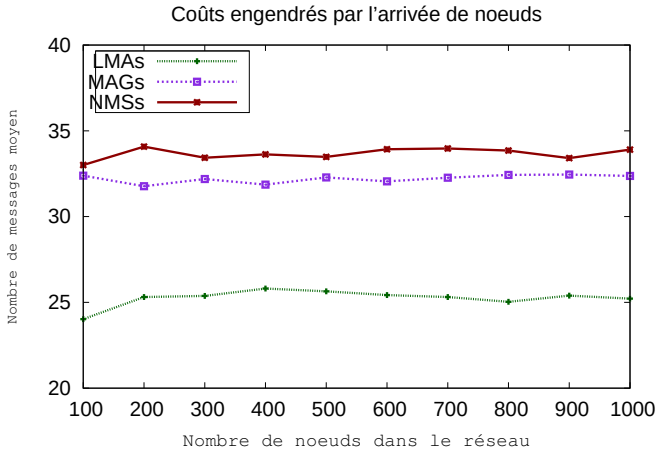


FIGURE 6.4 – Coûts d'enregistrement

Sur la figure 6.4, nous observons que, plus un noeud est de niveau bas, plus ses coûts d'enregistrement deviennent élevés. Ceci est dû au fait que, chaque noeud de bas niveau (NMS, *MAG*) doit nécessairement se connecter à un noeud de niveau plus haut (*MAG*, *LMA*). Or, lorsqu'un noeud arrive, le premier noeud de haut niveau qui est contacté peut atteindre la limite des noeuds fils qu'il est en mesure de supporter. Dans ce cas, le message sera acheminé à son successeur. Il est donc possible que ce message fasse le tour des noeuds de ce niveau haut avant de trouver un, qui soit en mesure de le connecter. Donc pour un NMS, le message peut faire le tour des *LMAs* et le tour des *MAGs*. Pour un *MAG*, le message peut faire le tour des *LMAs*. Tandis que pour un *LMA*, il sera juste connecté entre le premier *LMA* à répondre et son successeur.

De plus, les résultats montrent que l'augmentation du nombre de noeuds dans le réseau n'entraîne pas nécessairement l'accroissement des coûts. Cela témoigne le

passage à l'échelle de la solution.

Deuxième cas : coûts de vérification de survie : A intervalle de temps régulier de $5ms$, les *LMAs* et les *MAGs* envoient, à leur successeurs et à leurs noeuds fils, des messages *HELLO* pour vérifier leur survie. Ces derniers répondent par un message *HELLO*. L'ensemble des messages échangés constitue les coûts de vérification de survie. Ils sont représentés à la figure 6.5.

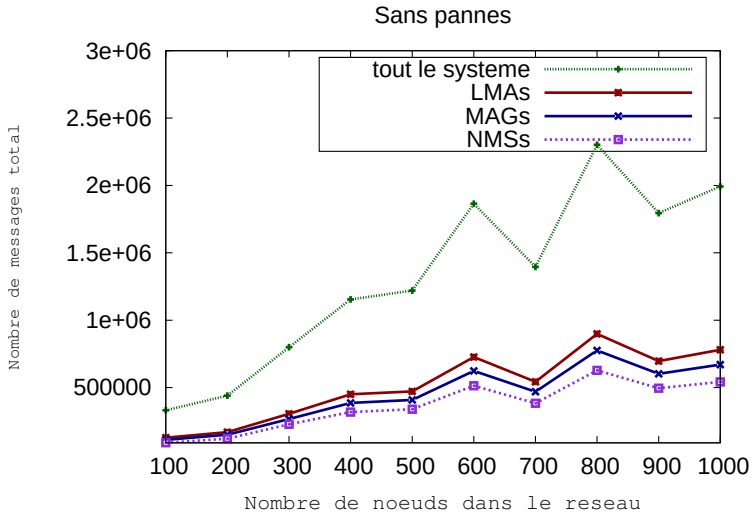


FIGURE 6.5 – Coûts de vérification de survie

Pour les coûts de vérification, les résultats de la figure 6.5 montrent que plus le noeud est de niveau haut, plus ses coûts sont élevés. Cela est dû au fait que pour les NMSs par exemple, ils ne font que répondre au messages *HELLO* qui leur sont envoyés. Ils n'initient pas de messages *HELLO*.
Pour les *MAGs*, en plus de répondre aux messages *HELLO* de leur *LMA*, ils doivent en envoyer à tous leurs NMSs et à leur premier successeur. Ce qui permet d'obtenir des coûts plus élevés que les NMSs.
Quant aux *LMAs*, ils doivent envoyer aussi bien à leur premier successeur, à leurs NMSs qu'à leurs *MAGs*. C'est pourquoi ils sont encore plus élevés que pour les autres types de noeuds, surtout lorsque le *LMA* possède beaucoup de noeuds fils.

Les résultats montrent également que l'augmentation des coûts de vérification n'est pas nettement dépendante de celle du nombre de noeuds dans le réseau. Car pour 800 noeuds par exemple, nous constatons qu'il y a plus de coûts que pour 900 noeuds. Ce qui prouve encore le passage à l'échelle. En réalité, ces coûts globaux dépendent de la composition réelle du réseau, c'est-à-dire, le pourcentage de *LMAs*, de *MAGs* et de NMSs. Or, de tels pourcentages ne sont pas déterminés à l'avance, puisque les noeuds arrivent dans le réseau de manière aléatoire, chacun venant avec ces capacités réelles qui le placent dans une catégorie de noeuds donnée. Donc un système peut avoir globalement plus de noeuds qu'un autre, mais que le dernier possèdent plus de *LMAs* et de *MAGs* que le premier. Dans ce cas, le deuxième système va générer plus de coûts de vérification que le premier.

4.4.2 Environnement avec pannes :

Dans cette partie, nous provoquons de manière aléatoire des pannes de noeuds en augmentant non seulement le pourcentage du nombre de noeuds en pannes mais aussi le nombre total de noeuds dans le réseau. Dans un second temps, nous fixons le pourcentage de noeuds en pannes à 60% et faisons varier le nombre de noeuds du système de 100 à 1000 avec un pas de 100. L'objectif est d'abord de vérifier, si après un nombre important de noeuds en pannes, le système peut toujours continuer à fonctionner, ensuite de voir l'impact des pannes sur les différentes métriques.

Coûts de localisation : dans cette sous partie, les métriques auxquelles nous nous intéressons sont le délai de localisation, le nombre de sauts pour localiser un noeud existant et le nombre total de messages générés depuis l'initiation de l'appel jusqu'à l'établissement de la session. Les valeurs recueillies après l'expérimentation son représentées au niveau des figures 6.6 (a), 6.6 (b) et 6.6 (c).

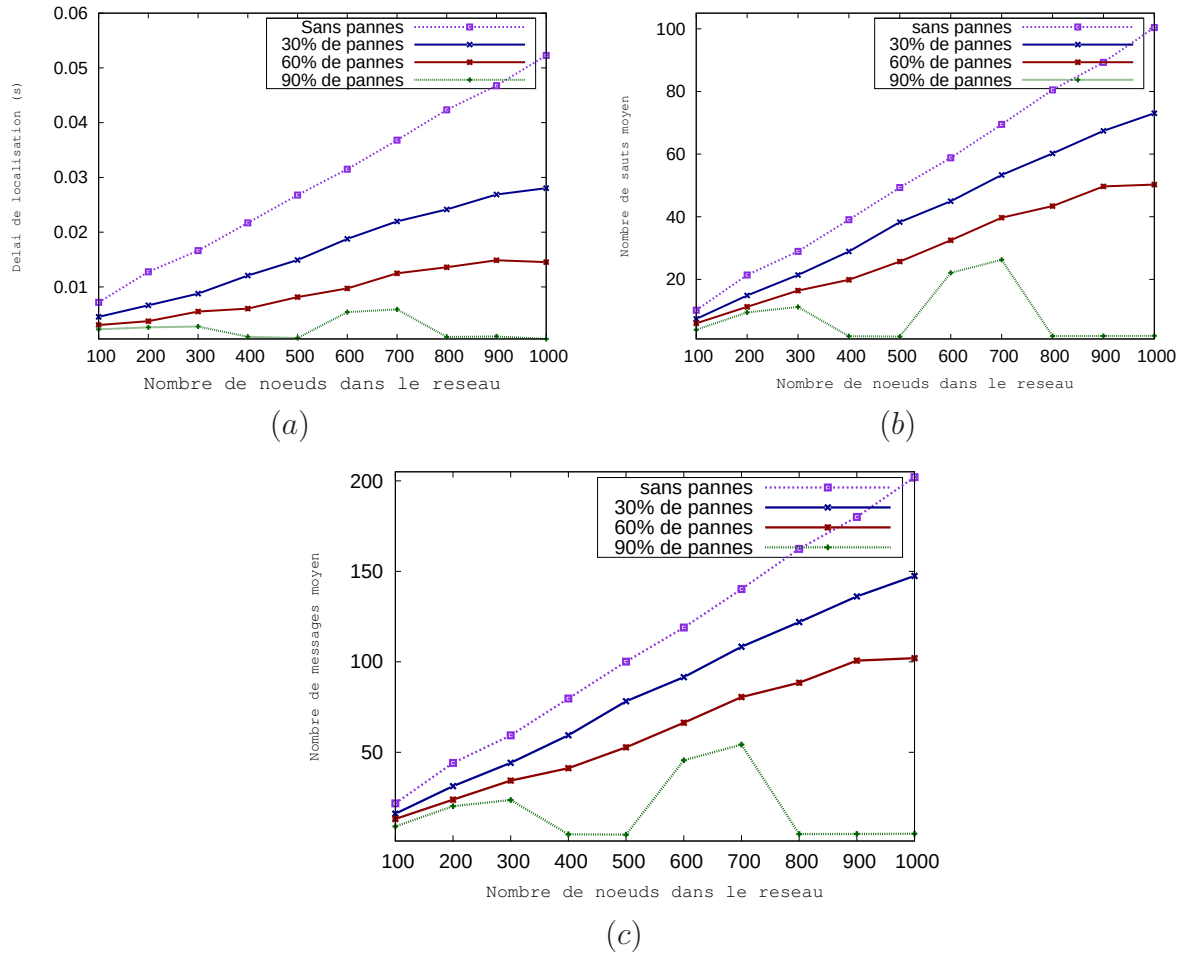


FIGURE 6.6 – Coûts de localisation en situation de pannes

Les trois figures (a), (b) et (c) montrent que lorsqu'il n'y a pas de pannes dans le système, la localisation d'un noeud prend un délai plus important qu'en cas de pannes (figure (a)). Il en est de même pour les nombres de sauts (figure (b)) et de messages total (figure (c)) de localisation. Ceci est dû au fait que dans l'architecture

$HPAMM^{IPv6}$, tous les messages de localisation sont envoyés au niveau des *LMAs*. Chaque fois qu'il reçoit un message *INVITE* un *LMA* doit parcourir la liste de ses NMSs et de ses *MAGS* avant de s'assurer que le destinataire n'est pas rattaché à lui, et auquel cas il l'achemine à son successeur. Or, lorsqu'il y a des pannes, s'il s'agit de NMSs ou de *MAGs* qui tombent en panne, ils sont tous supprimés de la liste de ses noeuds fils. Ainsi, le temps nécessaire pour parcourir cette liste devient plus petit. De plus, lorsqu'il s'agit de ses homologues *LMAs* qui sont en pannes, il envoie directement le message aux autres *LMAs* de la liste de ses r_{lma} successeurs. Ce qui lui permet d'atteindre plus rapidement beaucoup de noeuds à la fois (donc il gagne en nombre sauts et en délai), contrairement à la situation de non pannes dans laquelle il faudra faire le tour des *LMAs*.

En résumé, plus il y a des pannes, plus la taille des listes des noeuds fils des *LMAs* est réduite et plus les messages sont envoyés vers plusieurs noeuds à la fois (les successeurs) et donc plus il y a un gain en termes de délai, en nombre de sauts et par conséquent en nombre de messages total.

Coûts de handover logique : Pour ce cas, nous nous intéressons au délai qui sépare l'instant qu'un noeud détecte la panne de son voisin et l'instant qu'il se reconnecte à un autre (délai de *handover* logique). Dans un premier cas, nous fixons le nombre de noeuds dans le réseau à mille (1000) et faisons varier le pourcentage du nombre de noeuds en pannes. Dans le deuxième scénario, nous fixons le taux de noeuds en pannes à 60% et faisons varier le nombre de noeuds dans le réseau. Enfin, dans la troisième phase, le nombre de noeuds dans le réseau et le pourcentage de noeuds en pannes évoluent.

Premier cas : le nombre de noeuds est fixé

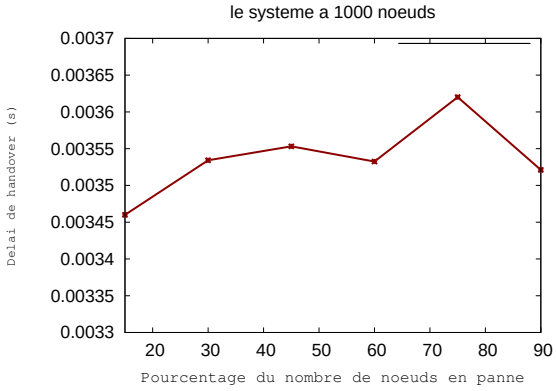


FIGURE 6.7 – Délai de handover avec un nombre de noeuds fixe

Avec la figure 6.7, nous constatons que le délai de *handover* n'est pas directement lié au taux de noeuds en pannes. Sur cette figure, nous voyons par exemple qu'avec un taux de pannes de 90%, le délai de *handover* est plus faible qu'avec un taux de 75% de pannes. Dans cette même lancée, le délai obtenu avec un taux de pannes de 15% est plus faible que celui de 30%. En plus, les résultats montrent que, même

avec un très fort taux de pannes (90%), le système fonctionne. Cela démontre de la robustesse de $HPAMM^{IPv6}$. En effet, ceci est dû au fait que dans $HPAMM^{IPv6}$, chaque noeud possède plusieurs "routes de secours".
Un NMS possède deux listes : une pour les successeurs de son *LMA* et une autre pour les sucesseurs de son *MAG*. Un *MAG* quant à lui, possède une liste des successeurs de son *LMA* et une autre de ses propres successeurs. Enfin, un *LMA* possède une liste de ses successeurs. Ainsi, lorsqu'un noeud détecte la panne de son voisin, il contacte tous les noeuds se trouvant dans ses différentes listes. Cet envoi multiple permet au noeud de gagner en délai et donc de pouvoir se reconnecter dans un délai raisonnable.

Deuxième cas : le taux de noeuds en panne est fixé

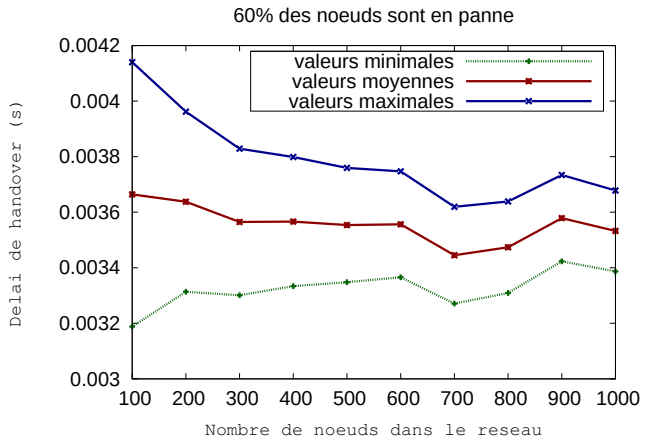


FIGURE 6.8 – Délai de handover avec un taux de pannes fixe

La figure 6.8 montre que le délai de *handover* ne croît pas en fonction du nombre de noeuds. Pour 700 noeuds par exemple, le délai obtenu est plus faible que pour 200. Ceci témoigne du passage à l'échelle de l'architecture. De plus, nous constatons que l'écart entre les valeurs maximales et minimales (qui détermine l'intervalle de confiance) se réduit au fur et à mesure que le nombre de noeuds croît. En d'autres termes, plus le nombre de noeuds augmente, plus la mesure devient précise ; ce qui confirme le passage à l'échelle. En effet, puisque le taux de noeuds en pannes est fixé, alors, plus le nombres de noeuds devient grand, plus il y a de noeuds pouvant être contactés par un autre à la recherche d'une reconnexion.

Troisième cas : le nombre de noeuds dans le système et le taux de pannes évoluent

La figure 6.9 est le condensé des figures 6.7 et 6.8. Elle confirme les deux résultats précédents, c'est-à-dire que le délai de *handover* n'est pas linéairement dépendant du taux de pannes ni du nombre de noeuds dans le réseau. En prenant par exemple le taux de pannes de 30%, nous observons que le délai qui y est obtenu est tantôt

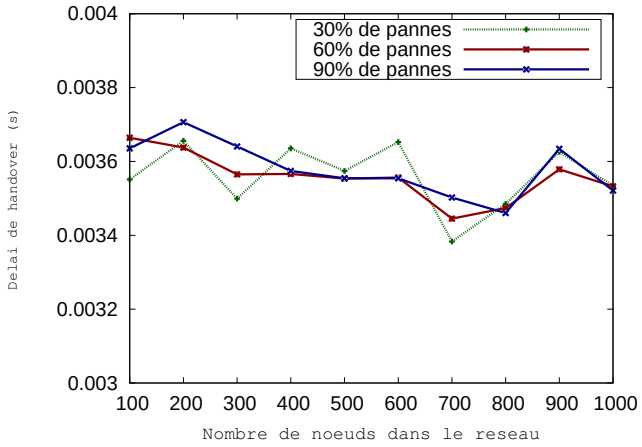


FIGURE 6.9 – Délai de handover avec un taux de pannes et un nombre de noeuds qui évoluent

supérieur aux autres (de 60% et de 90%), tantôt il en est inférieur. De même, le délai obtenu avec 700 noeuds par exemple est plus petit que ceux obtenus avec 600 ou 900 noeuds. En définitive, ces résultat confirment la robustesse et le passage à l'échelle de $HPAMM^{IPv6}$.

Coûts de tolérance aux pannes : Lorsqu'un noeud détecte la panne de son voisin, il cherche à se reconnecter à un autre noeud. Les différents messages échangés, depuis que le noeud constate la panne de son voisin jusqu'à ce qu'il se reconnecte à un autre noeud, constituent les coûts de tolérance aux pannes. Pour les évaluer, nous faisons les simulations dans trois scénarios différents. Un premier scénario où nous fixons le nombre de noeuds dans le réseau et faisons varier le taux de noeuds en panne. Dans le deuxième cas, nous fixons le taux de noeuds en panne à 60% et faisons varier le nombre de noeuds du réseau. Enfin, dans la troisième étape, le nombre de noeuds du réseau et le taux de pannes évoluent.

Premier cas : le nombre de noeuds est fixé

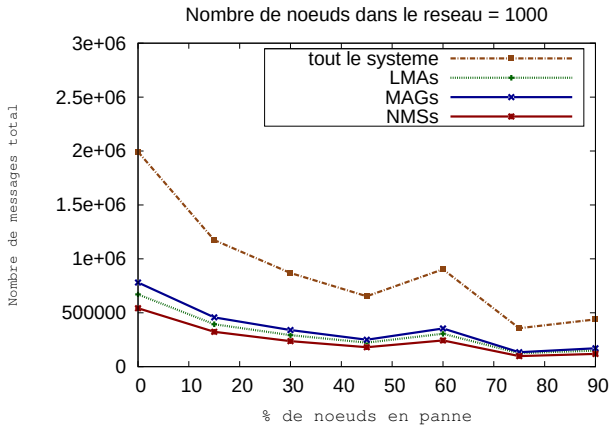


FIGURE 6.10 – Coûts de maintenance avec un nombre de noeuds fixe

Les résultats de la figure 6.10 montrent que, plus il y a de noeuds en panne, plus les coûts sont réduits. C'est parce que, lorsqu'un noeud détecte la panne de son voisin, il arrête de lui envoyer le message *HELLO*. Ainsi, plus le nombre de noeuds en panne augmente, moins il y a des noeuds vers lesquels il faut envoyer le message *HELLO*. Le nombre de messages qu'un noeud envoyait au début diminue car les éventuels destinataires n'existent plus.

De plus, nous constatons que les coûts des *MAGs* sont les plus élevés. En effet, lorsqu'un noeud de niveau haut détecte la panne d'un autre de niveau bas, il n'y a pas d'échange de messages. Le noeud de niveau bas est juste supprimé de la liste. Ainsi, les coûts générés par les *LMAs* n'existent qu'en cas de panne d'autres *LMAs*. Par contre pour les *MAGs*, les coûts sont générés, non seulement en cas de panne d'autres *MAGs*, mais aussi en cas de panne de *LMAs*. Dans ce dernier cas, tous les *MAGs*, dont c'est le *LMA* qui est en panne, vont créer des messages pour essayer de se connecter à un autre *LMA*. De même, lors de la panne de *MAGs*, en plus des messages que les autres *MAGs* envoient aux NMS pour qu'ils mettent à jour la liste des successeurs de leur *MAG*, il y a les nombreux messages pour reconstruire l'anneau. Ceux-ci sont souvent très nombreux car les prédécesseurs doivent opérer des mises à jour.
Pour les NMS, une panne d'un noeud de niveau haut les affecte. Cependant, les coûts qu'ils génèrent relèvent uniquement des messages qu'ils envoient aux différents successeurs de leur *MAG* ou de leur *LMA*.

Deuxième cas : le taux de noeuds en panne est fixé

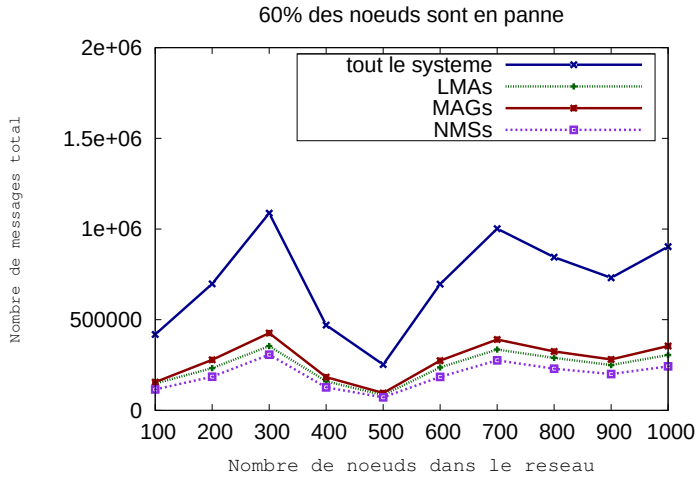


FIGURE 6.11 – Coûts de maintenance avec un taux de pannes fixe

Au niveau de la figure 6.11, les mêmes explications pour les coûts des *MAGs*, *LMAs* et NMS reviennent. En outre, la figure montre que les coûts n'ont pas une dépendance linéaire avec le nombre de noeuds du réseau. Pour 300 noeuds par exemple, les coûts sont plus élevés que pour 900 noeuds. Les coûts dépendent de la nature des noeuds en panne, c'est-à-dire du taux de *LMAs*, de *MAGs* et de NMSs en panne. Ceci confirme encore le passage à l'échelle, mais aussi et surtout la robustesse du

système, car même avec un fort taux de pannes, il fonctionne.

Troisième cas : le nombre de noeuds dans le système et le taux de pannes évoluent

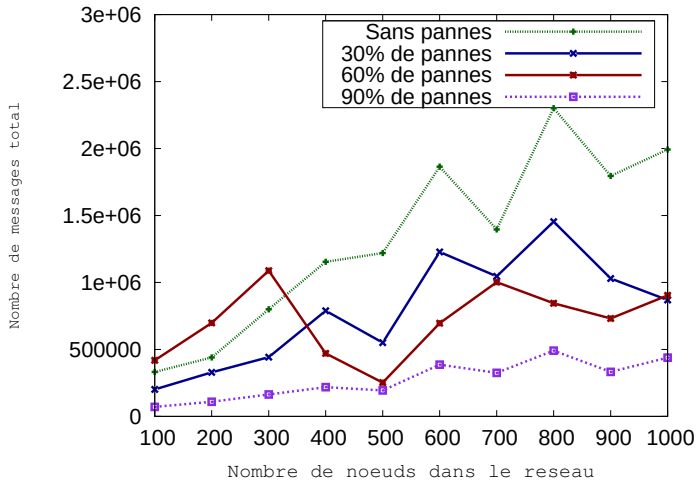


FIGURE 6.12 – Coûts de maintenance avec un taux de pannes et un nombre de noeuds qui évoluent

Avec la figure 6.12, nous observons toujours qu'il n'y a pas de corrélation directe entre le nombre de noeuds du réseau, le taux de noeuds en panne et les coûts de maintenance. Les mêmes explications tiennent toujours, à savoir que les coûts dépendent de la nature même des noeuds en panne et surtout de leur pourcentage (taux de *LMAs*, de *MAGs* et de NMSs en panne). Par ailleurs, nous constatons que les coûts sans pannes restent plus importants, surtout lorsque la taille du réseau augmente. C'est parce que dans ce cas, plus le nombre de noeuds croît, plus il y a des noeuds vers lesquels, il faut envoyer les messages *HELLO*.

De plus, les résultats confirment la robustesse car même avec 90% de noeuds en panne, le système fonctionne.

5 Conclusion

Dans ce chapitre, nous avons d'abord étudié les performances analytiques de notre solution et avons montré ses atouts par rapport à la solution SARP qui est la seule, à notre connaissance, à traiter du même cas. Ensuite, nous avons étudié de manière expérimentale ses performances. Sur les différentes métriques mesurées (coûts de maintenance, délai de localisation, délai de *handover*, nombre de sauts de localisation, nombre de messages global générés, etc.), les résultats nous ont montré que la solution est fortement robuste et passe à l'échelle.

De plus, les résultats nous montrent que, même avec un taux élevé de noeuds en panne, les délais de *handover* restent toujours faibles. Dans la même lancée, les différents résultats montrent que ces métriques dépendent, beaucoup plus de la structuration du réseau que de sa taille.

Conclusion générale et perspectives

Conclusion

Dans cette thèse, nous avons eu comme objectif principal, de proposer une solution permettant de gérer les communications mobiles dans un réseau pair-à-pair SIP hiérarchique, en nous basant sur le protocole IPv6. Au vu de l'objectif, nous avons concentré nos études sur, dans un premier temps, la mobilité dans IPv6, et dans un second temps, la mobilité dans un système pair-à-pair SIP. Nous avons par la suite proposé une architecture permettant de gérer la mobilité dans un système P2PSIP hiérarchique, gérant la mobilité lors des phases de localisation et de communication et intégrant la gestion du *handover* logique et physique.

Ainsi, pour mener à bien nos travaux, nous avons d'abord parcouru les multiples solutions de la gestion de la mobilité dans le protocole IPv6. Cette investigation nous a permis de découvrir les modes de gestion du *handover*, existants dans IPv6. Nous avons donc mené une étude comparative de ces derniers afin de trouver les solutions les plus optimisées. Nous avons ensuite effectué un état de l'art et une étude comparative sur les différents types d'architectures pair-à-pair SIP hiérarchiques. Suite à ces travaux, nous avons retenu le mode prédictif avec le *FPMIPv6* (permettant d'avoir de faibles délais de *handover*), combiné à une architecture P2PSIP hiérarchique à plusieurs niveaux avec un mode de routage hybride (récursive/semi-récursive). La méthode récursive, malgré qu'elle génère plus de messages que la méthode semi-récursive, nous a permis d'assurer la cohérence de la table *CLN*.

Partant de ces études, nous avons ensuite proposé une architecture pair-à-pair SIP hiérarchique ($HPAMM^{IPv6}$) pour gérer la mobilité des noeuds en nous basant sur le protocole IPv6. Etant basée sur l'une des meilleures combinaisons proposées, $HPAMM^{IPv6}$ permet de gérer tous les différents types de mobilité qui existent (mobilité physique et mobilité logique). Elle prend en charge la gestion de la mobilité lors des phases de localisation et de la communication. Puisque $HPAMM^{IPv6}$ est composée de trois niveaux hiérarchiques, nous avons montré par son dimensionnement, qu'elle est très extensible. Cette extensibilité démontrée théoriquement, a été également prouvée par les résultats expérimentaux.

Enfin, nous avons étudié les performances de l'architecture $HPAMM^{IPv6}$ par des validations analytiques et expérimentales. Les multiples simulations effectuées ont montré que la solution proposée présente des performances optimales (temps de *handover* réduits, coûts de mainance réduits, scalabilité, robustesse). En d'autres termes, les résultats des simulations ont montré que, même avec une taille du réseau

qui augmente et un fort taux de noeuds en panne, les délais de localisation et de *handover* restent encore faibles. De même, avec toujours un pourcentage élevé de noeuds en panne et un réseau qui croît, les coûts de maintenance du réseau restent raisonnables (car ils n'augmentent pas en fonction de la taille du réseau). Ce qui montre que $HPAMM^{IPv6}$ est une architecture tolérante aux pannes et peut fonctionner à l'échelle.

Perspectives

Les perspectives qui ressortent de cette thèse viennent essentiellement de deux aspects. D'abord, elles sont tirées des limites de l'architecture proposée, que nous avons soulignées plus haut (cf. chapitre 5, section 5). Ensuite, elles proviennent des nombreux défis issus de la solution proposée.

Vis-à-vis des limites

Nous avons précisé dans le chapitre 5 que la solution présente un défaut d'équilibrage de charge, mais également que lorsqu'il y a un nombre de $LMAs$ égal à $log_2(N_{lma})$ (respectivement un nombre de $MAGs$ égal à $log_2(N_{mag})$) **successifs** qui tombent en panne à la fois, le système arrête de fonctionner. N_{lma} et N_{mag} représentent respectivement le nombre de $LMAs$ et le nombre de $MAGs$ dans le système. Ainsi, dans nos travaux futurs, nous comptons mettre en place une stratégie d'équilibrage de la charge, mais aussi étudier la stabilité du système vis-à-vis des défis de $HPAMM^{IPv6}$.

Vis-à-vis des ouvertures

Nous avons montré que la solution gère beaucoup d'aspects qui n'étaient pas pris en compte par les solutions existantes. Ainsi, nous envisageons poursuivre l'expérimentation avec le *handover* physique et ensuite, nous projetons de développer une plate forme complète de $HPAMM^{IPv6}$, et la déployer concrètement dans un réseau, afin de faire bénéficier aux utilisateurs des apports réels de la solution. En outre, lorsqu'il s'est agi d'implémenter notre solution, nous avons remarqué qu'il n'y avait pas de couche pair-à-pair (parmi les modèles standards) permettant de porter notre modèle d'architecture hiérarchique. C'est pourquoi, nous envisageons également continuer à améliorer $HPAMM^{IPv6}$ pour en faire un modèle standard de simulation d'architectures pair-à-pair hiérarchiques (comme le sont Chord, pastry, etc. pour des architectures pair-à-pair plates).

Bibliographie

[Abd2016] R. M. Abdullah, Z. A. Zukarnain, and R. Iqbal. 2016. *"Improved fast handover method for multiple node by using mobile nodes guide"*. In Telecommun Systems, Springer.

[Abe2016] Abel Diatta, Ibrahima Niang and Mandicou Ba, *"An efficient fault tolerant scheme for mobility management in wireless networks"*, in Int'l Conf. Par. and Dist. Proc. Tech. and Appl. | PDPTA'16 |, July 2016, pp. 50 – 56.

[Abe2017] A. Diatta, I. Niang, B. Gueye and M. Ba. *"PMIPv6-Based Mobility for Realtime Communications on Hierarchical P2PSIP Systems"*. in IEEE International Symposium on Advances in Communications and Computing for Internet-of-things (CCIoT-2017), june 2017, Exeter, Devon, United Kingdom

[Aish2013] Aisha-Hassan A.Hashim, Wan H. Hassan, Shayla Islam, R.A. Saeed, M.K. Hasan, Jamal I. Daoud and Othman O. Khalifa, *"An Enhanced Macro Mobility Management Scheme in NEMO Environment to Achieve Seamless Handoff"*, World Applied Sciences Journal 21 (Mathematical Applications in Engineering) : 35-39, 2013, ISSN 1818-4952, © IDOSI Publications, 2013, DOI :10.5829/idosi.wasj.2013.21.mae.99910

[Ana2015] Ananthi Jebaseeli Samuelraj and Sundararajan Jayapal. 2015. *"Efficient Mobility Management Signalling in Network Mobility Supported PMIPV6"*. The Scientific World Journal 2015 (2015), 14.

[Ant2001] Antony Rowstron and Peter Druschel, *"Pastry : Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems"*, R. Guerraoui (Ed.) : Middleware 2001, LNCS 2218, pp. 329–350, 2001. Springer-Verlag Berlin Heidelberg 2001

[arcep2017] Autorité de régulation des communications électroniques et des postes (ARCEP). *"LES SERVICES DE COMMUNICATIONS ELECTRONIQUES EN FRANCE, 3^E TRIMESTRE 2016, OBSERVATOIRE DES MARCHES DES COMMUNICATIONS ELECTRONIQUES 05 JANVIER 2017"*, ISSN n^o 2258-3106

[Art2007] M. Artigas, P. Lopez, and A. Skarmeta, *"A comparative study of hierarchical dht systems"*, in Local Computer Networks, 2007. LCN 2007. 32nd IEEE Conference on, Oct 2007, pp. 325–333.

[Bar1999] R.Albert and A-L. Barabási, *"Emergence of Scaling in Random Networks"*, Science, vol. 286, pp. 509-512, 1999

[Bau2012] I. Baumgart and B. Heep, *"Fast but economical : A simulative comparison of structured peer-to-peer systems"*, in Next Generation Internet (NGI), 2012 8th EURO-NGI Conference on. IEEE, 2012, pp. 87–94

[Ben2004] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John D. Kubiatowicz, *"Tapestry : A Resilient Global-Scale Overlay for Service Deployment"*, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 22, NO. 1, JANUARY 2004.

[Bra2012] O. Bravo, A. Costa, and M. Nicolau, *"Design and implementation of a hierarchical sip-based peer-to-peer network"*, in Software, Telecommunications and Computer Networks (SoftCOM), 2012 20th International Conference on, Sept 2012, pp. 1–9.

[Bry2008] D. Bryan, B. Lowekamp, and M. Zangrilli, *"The design of a versatile, secure p2psip communications architecture for the public internet"*, in Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on, April 2008, pp. 1–8.

[Bud2003] Nathalie BUDAN, Benoit TEDESCHI, Stéphane VAUBOURG *"Nouvelles Technologies Réseau : Les réseaux peer-to-peer, Fonctionnement, exemples, limites"*, 2003.

[Chu2011] M. C. Chuang and J. F. Lee. 2011. *"FH-PMIPv6 : A fast handoff scheme in Proxy Mobile IPv6 networks"*. In Consumer Electronics, Communications and Networks (CECNet). 1297–1300.

[Cos2010] Daniel G. Costa and Sergio V. Fialho. 2010. *"A P2P Architecture to Support Mobile Real-Time Multimedia Communications"*. JOURNAL OF MULTIMEDIA 5, 5 (OCTOBER 2010), 514–521.

[credoc2015] Lucie Brice, Patricia Croutte, Pauline Jauneau-Cottet, Sophie Lautié. *"Baromètre du NUMÉRIQUE - Édition 2015"*. Autorité de régulation des communications électroniques et des postes (ARCEP)

[Dian2010] I. Diane, I. Niang, and B. Gueye, *"A hierarchical dht for fault tolerant management in p2p-sip networks"*, in Proceedings of the 2010 IEEE 16th, ser. ICPADS '10, 2010, pp. 788–793.

[Diat2015] A. Diatta, I. Niang, and M. Ba, *"An overview and comparison of hierarchical p2p-sip networks"*, (IJCSIS) International Journal of Computer Science and Information Security, vol. 13, no. 3, pp. 47–58, March 2015.

[Diat2016] A. Diatta, I. Niang, and M. Ba, *"Cost analysis in fmipv6 and fpmipv6-based on hp2p-sip networks"*, in 2016 IEEE 7th Annual (UEMCON), Oct 2016, pp. 1–7.

[Diat2017] Abel DIATTA, Ibrahima NIANG, Mandicou BA, and Bassirou GUEYE. 2017. *"A Survey of IPv6-Based Mobility Support for P2P-SIP Networks"*. In Proceedings of Second International Conference on Advanced Wireless Information, Data, and Communication Technologies (AWICT 2017), Universite de Paris-Saclay, Paris, France, Nov 2017 (AWICT 2017), 8 pages. https ://doi.org/10.1145/nnnnnnn.nnnnnnn

[Elys2012] HOUNGUE YENUKUNME PELAGIE ELYSE, E. Damiani, R. Glitho *"A MIDDLEWARE-INDEPENDENT AND SECURE PEER-TO-PEER SIP ARCHITECTURE (MISE-P2PSIP)"*, 2012 Mar 06.((23. ciclo, Anno Accademico 2010.

[Erd1959] P. Erdös and A. Rényi, *"On Random Graphs I"*, Publ. Math. Debrecen, vol. 6, pp. 290-297, 1959

[Eun2014] Eunsam Kim, Sangjin Kim, and Choonhwa Lee. *"Supporting Seamless Mobility for P2P Live Streaming"*. The Scientific World Journal 2014, 8. https ://doi.org/10.1155/2014/134391

[Fab2004] Fabien Viger and Matthieu Latapy, *"Génération de graphes connexes aléatoires avec séquence de degrés donnée"*, novembre 2004

[Far2006] R. Farha, K. Khavari, N. Abji, and A. Leon-Garcia. 2006. *"Peer-to-Peer Mobility Management for all-IP Networks"*. In 2006 IEEE International Conference on Communications, Vol. 5. 1946–1952.

[Fessa2006] Fabrice Le Fessant, *"Le peer-to-peer : Comprendre et utiliser"*,*N°* 11731, 2006, 168 pages, Editions EYROLLES

[Fu2014] H. L. Fu, P. Lin, H. Yue, G. M. Huang, and C. P. Lee. *"Group Mobility Management for Large-Scale Machine-to-Machine Mobile Networking"*. IEEE Transactions on Vehicular Technology 63, 3, 2014, 1296–1305.

[Furn2013] J. Furness, M. Kolberg, and M. Fayed, *"An evaluation of chord and pastry models in oversim"*, in Modelling Symposium (EMS), 2013 European, Nov 2013, pp. 509–513.

[Gar2003] L. Garcece-Erice, E. Biersack, P. Felber, K. Ross, and G. Urvoy-Keller, *"Hierarchical peer-to-peer systems"*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, vol. 2790, pp. 1230–1239.

[Georg2011] Georges Rodriguez, *"Introduction aux réseaux cellulaires Techniques d'accès et de partage de la ressource radio"*, Systèmes de Télécommunication Cycle d'harmonisation 2A AST TEL-COM202, Septembre 2011.

[Gua2016] Jianfeng Guan, Ilsun You, Changqiao Xu, and Hongke Zhang. 2016. *"The PMIPv6-Based Group Binding Update for IoT Devices"*. Mobile Information Systems 2016 (2016), 8.

[ILR2012] Institut Luxembourgeois de Régulation (ILR). *"Rapport statistique des télécommunications du Luxembourg de l'année 2011"*. Juin 2012

[Irfan2009] Irfan Ali, Alessio Casati, Kuntal Chowdhury, Katsutoshi Nishida, Eric Parsons, Stefan Schmid, Rahul Vaidya,*"Network-Based Mobility Management in the Evolved 3GPP Core Network"*, IEEE Communications Magazine - February 2009, LTE — 3GPP RELEASE 8

[Isla2011] M. S. Islam, S. A. Rahman, R. Ahmen, and M. H. Raju, *"A hierarchical overlay design for peer to peer and sip integration"*, International Journal of Computer Science and Information Security, vol. 9, pp. 94–99, 2011.

[John2004] D. Johnson, C. Perkins, and J. Arkko, *"Mobility support in ipv6"*, in Internet Soc., Reston, VA, IETF RFC 3775, June 2004.

[Kha2014] R. A. Khan and A. H. Mir. 2014. *"Performance analysis of host based and network based IP mobility management schemes over IPv6 network"*. In Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on. 1798–1803.

[Kim2016] M. S. Kim and S. Lee. 2016. *"Enhanced Network Mobility Management for Vehicular Networks"*. IEEE Transactions on Intelligent Transportation Systems 17, 5 (May 2016), 1329–1340.

[Kris2016] I. L. Krishnan and S. P. Davidson. 2016. *"A Novel Approach to Reduce Handover Latency in Proxy Mobile IPv6 Based on Multi-Homing"*. In Circuits and Systems, Vol. 7. 2530 – 2541.

[Lap1988] J. C. Laprie, *"Surete de fonctionnement et tolerance aux fautes : concepts de base"*. Rapport technique LAAS-88287, Laboratoire d'automatique et d'analyse des systèmes (Toulouse), 1988.

[Leb2016] Lebajoa A. Mphatsi. 2016. *"A Scalable Fast Handovers Proxy Mobile IPv6 Scheme for 4G Wireless Networks"*. IJCSNS International Journal of Computer Science and Network Security 16, 6 (2016), 52 – 61.

[Lee2010] M. S. Kim, S. Lee, D. Cypher, and N. Golmie. 2010. *"Fast Handover Latency Analysis in Proxy Mobile IPv6"*. In Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE. 1–5.

[Lei2016] Lei Zhang and Yu-Chu Tian. 2016. *"An enhanced fast handover triggering mechanism for Fast Proxy Mobile IPv6"*. In Wireless Network, Springer.

[Lekuo2007] L. Le and G.-S. Kuo, *"Hierarchical and breathing peer-to-peer sip system"*, in Proceedings of the IEEE International Conference on Communications,, ser. ICC '07, 2007, pp. 1887–1892.

[Leslie1978] Leslie Lamport, *"The Implementation of Reliable Distributed Multiprocess Systems"*, © North-Holland Publishing Company, Computer Networks 2 (1978)95-114

[Leu2008] Ed. K. Leung S. Gundavelli, V. Devarapalli, K. Chowdhury, and B. Patil. 2008. *"Proxy Mobile IPv6"*. Network Working Group RFC 5213.

[Liu2012] W. Liu, J. Song, and J. Yu, *"An overlapping structured P2P for REIK overlay network"*, Physics Procedia, vol. 33, no. 0, pp. 1022– 1028, 2012 International Conference on Medical Physics and Biomedical Engineering (ICMPBE2012).

[Luo2011] H. Luo, H. Zhang, Y. Qin, and V. C. M. Leung. 2011. *"An Approach for Building Scalable Proxy Mobile IPv6 Domains"*. IEEE Transactions on Network and Service Management 8, 3 (September 2011), 176–189.

[Man2014] Mandicou Ba *"Vers une structuration auto stabilisante des réseaux ad hoc : cas des réseaux de capteurs sans fil"*, 2014, 164 pages

[Mar2003] Martin Drapeau, *"Typologie des réseaux Peer-to-peer et application au réseau GNUtella"*, 11000712, Décembre 2003

[Mat2008] M. Matuszewski and E. Kokkonen. 2008. *"Mobile P2PSIP - Peer-to-Peer SIP Communication in Mobile Communities"*. In 2008 5th IEEE Consumer Communications and Networking Conference. 1159–1165.

[Mol1998] M. Molloy and B. Reed, *"The size of the giant component of a random graph with a given degree sequence"*, Combin. Probab. Comput., pp. 295-305, 1998

[Mor2003] M. Torrent-Moreno, X. Perez-Costa, and S. Sallent-Ribes. 2003. *"A performance study of fast handovers for mobile IPv6"*. In Local Computer Networks, 2003. LCN'03. Proceedings. 28th Annual IEEE International Conference on. 89–98.

[MPhil2014] N.Gomathy MPhil, Dr.N.Radha M.Sc, MPhil, Phd, *"A Survey on Mobility Management Protocols for Improving Handover Performance"*, International Journal of Computer Trends and Technology (IJCTT) – volume 10 number 1 – Apr 2014

[Mus2012] M. M. Muslam, H. A. Chan, L. A. Magagula, and N. Ventura. 2012. *"Network-based mobility and Host Identity Protocol"*. In IEEE Wireless Communications and Networking Conference (WCNC). 2395–2400.

[Myu2014] Myungseok Song and Jongpil Jeong. 2014. *"Performance Analysis of A Novel Inter-Networking Architecture for Cost-Effective Mobility Management Support"*. KSII TRANSACTIONS ON INTERNET AND INFORMATION SYSTEMS 8, 4 (Apr 2014), 1344–1367.

[Par2012] J. T. Park, S. M. Lee, and S. M. Chun. 2012. *"P2P-based mobility management protocol for global seamless handover in heterogeneous wireless networks"*. In Software, Telecommunications and Computer Networks (SoftCOM). 1–5.

[Plan2010] equipe Planete, *"Mobile IPv6 (MIPv6)"*, INRIA Rhône-Alpes, 18 janvier 2010.

[Rat2001] Ratnasamy, Sylvia and Francis, Paul and Handley, Mark and Karp, Richard and Shenker, Scott, *"A Scalable Content-addressable Network"*, SIGCOMM Comput. Commun. Rev., 2001, vol. 31, no 4, pp. 161-172.

[RFC5128] P. Srisuresh, B. Ford, D. Kegel, *"State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs)"*, Request for Comments : 5128, March 2008

[Shi2007] J. Shi, Y. Wang, L. Gu, L. Li, W. Lin, Y. Li, Y. Ji, and P. Zhang, *"A hierarchical peer-to-peer sip system for heterogeneous overlays interworking"* in Proceedings of the IEEE Global Telecommunications Conference, ser. GLOBECOM '07, 2007, pp. 93–97.

[Sil2006] Silvia Hagen, *"IPv6 Essentials"*, Second Edition, ISBN :978-0-596-10058-2, O'REILLY, 2006

[Sim2005] Simon ZNATY, Jean-Louis DAUPHIN, *"SIP : Session Initiation Protocol"*, EFORT 2005, http ://www.efort.com

[Sing2005] K. Singh and H. Schulzrinne, *"Peer-to-peer internet telephony using sip"*, in Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video, ser. NOSSDAV '05. New York, NY, USA : ACM, 2005, pp. 63–68.

[Song2015] B. H. Song, J. Shin, S. Kim, and J. Jeong. 2015. *"On PMIPv6-Based Mobility Support for Hierarchical P2P-SIP Architecture in Intelligent Transportation System"*. In System Sciences (HICSS), 2015 48th Hawaii International Conference on. 5446-5452.https ://doi.org/10.1109/HICSS.2015.637

[Sor2012] P. Sornlertlamvanich, S. Kamolphiwong, R. Elz, and P. Pongpaibool. *"NEMO-Based Distributed Mobility Management"*. In Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on. 645–650.

[Soto2010] Ignacio Soto, Carlos J. Bernardos, MarÃŋa CalderÃşn, and Telemaco Melia. *"PMIPv6 : A Network-Based Localized Mobility Management Solution"*. The Internet Protocol Journal, septembre 2010, volume 13, no 3. https ://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-49/133-pmipv6.html

[Stoi2001] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan,*"Chord : A scalable peer-to-peer lookup service for internet applications"*, ACM SIGCOMM Computer Communication Review, vol. 31, no. 4, pp. 149–160, 2001.

[Tanen2007] Andrew S. Tanenbaum, Maarten Van Steen, *"DISTRIBUTED SYSTEMS : Principles and Paradigms"*, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

[Thie2002] Thierry Lucidarme. *"Principes de radiocommunication de troisième génération"*, Vuibert, Paris, 2002, ISBN 2-7117-8693-5

[Tho2017] Thomas Coëffé. *"Les statistiques du digital en 2017 : mobile, publicité, gaming, Chine, GAFAM. . . "*, le 1 juin 2017

[Waq2013] Waqas A. Imtiaz. 2013. *"Two-Tier CHORD for Decentralized Location Management"*. International Journal of Computer Applications (0975 âĂŞ 8887) 69, 4 (May 2013), 1 – 5.

[Waro2012] Warodom WERAPUN, *"Architectures de réseaux pour la délivrance de services à domicile"*, Septembre 2012, 175 pages.

[Wei2015] Wei Siang Hoh, Sashikumar Muthut, Bi-Lynn Ong, Mohamed Elshaikh, Mohd Nazri Mohd Warip, and R. Badlishad Ahmad. 2015. *"A SURVEY OF MOBILITY MANAGEMENT PROTOCOLS"*, In ARPN Journal of Engineering and Applied Sciences, Vol. 10. 9015 – 9019.

[Yah2012] S. Yahiaoui, Y. Belhoul, N. Nouali-Taboudjemat, and H. Kheddouci, *"Adsip : Decentralized sip for mobile ad hoc networks"*, in Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on, March 2012, pp. 490–495.

[Yel2009] Isaias Martinez-Yelmo, Alex Bikfalvi, and Carmen Guerrero. 2009. *"Benefits on using H-P2PSIP in mobile environments"*. 2009, Asociación de Ingeniería Telemática, VIII Jornadas de Ingeniería Telemática (JITEL '09), http ://hdl.handle.net/10016/7938.

[Yelmo2009] Isaias Martinez-Yelmo, Carmen Guerrero, Ruben Cuevas, Andreas Mauthe. *"A Hierarchical P2PSIP Architecture to support Skype-like services"*. Parallel, Distributed and Network-based Processing, 2009 17^{th} Euromicro International Conference on. pp 316 - 322

[Yizhen2014] Yizhen Wu, Ke Chen, Kaiping Xue, Dan Ni, *"NEMO-based Mobility Management in LISP Network"*, 2014 Sixth International Conference on Wireless Communications and Signal Processing (WCSP)

[Yon2008] H. Yongfeng, S. Tang, and Y. Yip, *"A new security architecture for sip based p2p computer networks"*, Journal of Computer Science, Informatics and Electrical Engineering, vol. 2, no. 1, 2008.

[Youn2009] Youn-Hen Han *"MIPv4 & MIPv6 - Overview of IP Mobility Protocols"*, Korea Univ. Of Technology, February 2009

[Zheng2010] X. Zheng and V. Oleshchuk, *"A Survey on Peer-to-Peer SIP Based Communication Systems"*, Peer-to-Peer Networking and Applications, vol. 3, no. 4, pp. 257–264, 2010.

[Zho2013] J. Zhou, R. Chai, and Q. Chen, *"Network-based mobility management for lisp network"*, in 2013 22nd Wireless and Optical Communication Conference, May 2013, pp. 360–365.

[Zoels2006] S. Zoels, Z. Despotovic, and W. Kellerer, *"Cost-based analysis of hierarchical dht design"*, in Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing, ser. P2P '06, 2006, pp. 233-239.

[Zoh2014] F. T. Zohra, S. Azam, and Md. M. Rahman. 2014. *"Overview of IPv6 Mobility Management Protocols and their Handover Performances"*. International Journal of Computer Science and Engineering (IJCSE) 2, 3 (March 2014), 121–129.