Architecture autonome et dynamique d'adressage, de routage et de nommage pour réseaux recouvrants

Khaldoon Shami

Université de Haute Alsace
Laboratoire Modélisation, Intelligence, Processus, Systèmes
Groupe de recherche en Réseaux et Télécommunications de Colmar

N° d'ordre : 08MULH0917

THÈSE

Pour l'obtention d'un
Doctorat de l'Université de Haute Alsace
Spécialité Informatique

Présentée par

Khaldoon Shami

ARCHITECTURE AUTONOME ET DYNAMIQUE D'ADRESSAGE, DE ROUTAGE ET DE NOMMAGE POUR RESEAUX RECOUVRANTS

Soutenue publiquement le 15 octobre 2008 devant le jury composé de :

M. Hervé GUYENNET, rapporteur
Professeur à l'Université de Franche Comté
M. Pascal LORENZ, directeur de thèse
Professeur à l'Université de Haute Alsace
M. Damien MAGONI, co-encadrant de thèse
Professeur à l'Université de Bordeaux 1
M. Abdelhamid MELLOUK, rapporteur
Professeur à l'Université de Paris 12
M. Samir TOHME, examinateur
Professeur à l'Université de Versailles Saint-Quentin

Remerciements

Mes plus vifs remerciements s'adressent tout d'abord aux membres du jury, notamment à Monsieur Samir TOHME pour avoir bien voulu en prendre la présidence, ainsi qu'à Monsieur Hervé GUYENNET et à Monsieur Abdelhamid MELLOUK pour m'avoir fait l'honneur d'accepter d'être les rapporteurs de cette thèse. Je leur témoigne toute ma gratitude pour le temps consacré à l'examen de mon travail de thèse.

Je remercie respectueusement Monsieur Pascal LORENZ pour la confiance qu'il m'a accordée en acceptant de m'accueillir au sein de son équipe de recherche et pour avoir accepté de diriger cette thèse. Je le remercie aussi pour tous les efforts par lesquels il a contribué à l'avancement de mes travaux.

Mes sincères remerciements vont à Monsieur Damien MAGONI pour avoir été mon co-encadrant tout au long de mes travaux de thèse, pour ses encouragements, ses critiques constructives et surtout pour l'aide qu'il n'a cessé de m'accorder tout au long de ces années. Un grand merci pour m'avoir guidé dans mes recherches avec beaucoup de patience. Je vous en suis profondément reconnaissant.

Que ceux qui ont participé, de près ou de loin à l'aboutissement de ce travail, trouvent ici l'expression de mes sincères remerciements.

Je souhaite également remercier mes collègues et l'ensemble des membres de l'équipe du GRTC et tous ceux qui m'ont accompagné en pensée tout au long de cette thèse et qui m'ont soutenu surtout pendant les moments difficiles.

Je ne trouverai sans doute pas les mots pour remercier assez les personnes qui me sont les plus chères et sans lesquelles je n'aurai pas pu arriver là où je suis aujourd'hui : les membres de ma famille et notamment mes parents, pour leurs encouragements, leurs sacrifices, leur patience, leur bienveillance, leur soutien et tout ce qu'ils ont fait pour m'apporter le bonheur. Et bien évidemment, je n'oublierai pas ma très chère fille Aya. Sa présence à mes côtés m'a encouragé et m'a poussé à faire de mon mieux afin de mener cette thèse à bien. C'est à elle que je dédie ce travail. Je vous remercie tous de tout mon coeur.

Table des matières

Chapitre 1

Introduction

Actuellement, Internet offre un groupe de protocoles qui gèrent avec succès les communications point-à-point. Cependant, avec l'extension d'Internet ces dernières années, ces protocoles sont devenus difficiles à modifier ou à changer. Il est devenu par conséquent très difficile d'incorporer des nouveaux services au niveau de la couche réseau.

Pendant ces dernières années le concept des réseaux recouvrant (« overlay networks ») a émergé afin d'augmenter les fonctionnalités et la performance d'Internet. Ce concept a permis aussi d'expérimenter et de déployer des idées et des protocoles plus facilement en créant des topologies virtuelles au dessus du réseau Internet existant sans y faire aucun changement et sans prendre de risque au niveau de l'infrastructure. Les avantages de l'approche par recouvrement sont :

- offrir des nouveaux services, ce qui n'était pas possible à implémenter dans l'infrastructure actuelle sans déployer des réseaux recouvrants spécifiques,
- améliorer la performance des services existants.

Dès le début, Internet a adopté un modèle clair dans lequel des routeurs à l'intérieur du réseau sont responsables de la propagation des paquets d'une source à une destination et des applications tournant sur des stations connectées aux périphéries du réseau. Mais depuis quelques années de nouvelles applications prenant leurs propres décisions de propagation de l'information sont apparues. Les réseaux recouvrants ont été dès lors vus comme un mécanisme offrant des nouvelles fonctionnalités sur Internet.

On peut imaginer un réseau recouvrant comme un réseau logique implémenté au dessus d'un réseau physique. Mais en fait, les réseaux recouvrants peuvent être implémentés sur d'autres réseaux recouvrants, utilisant dans la plupart des cas Internet comme réseau sous-jacent.

Chaque noeud dans le réseau recouvrant existe dans le réseau inférieur. Ces noeuds sont reliés par des liens appelés tunnels à l'instar des VPN. Les noeuds de part et d'autre d'un tunnel sont chargés de router les paquets reçus vers leur destinataire (tout comme un routeur de niveau 3). Ces paquets contiennent entre autre l'identificateur des noeuds correspondant à la source et à la destination qui, précisons-le, n'est pas forcement une adresse IP. Chaque noeud est en mesure d'interpréter les deux en-têtes.

Les réseaux recouvrants sont un moyen d'introduire de nouvelles technologies indépendamment des processus de standardisation. Il n'existe en effet aucun standard que l'on pourrait utiliser comme exemple de base. Les chercheurs ont néanmoins proposé plusieurs idées de routage, d'adressage et de nommage. Les réseaux recouvrants sont idéaux pour expérimenter de nouveaux déploiements d'IP.

La conception d'un intergiciel pour l'adressage, le routage et le nommage au niveau application d'un réseau recouvrant sur Internet est un défi quand il n'y a aucune contrainte sur la topologie de ses membres. Il est très utile de fournir un tel intergiciel pour créer ces réseaux, par exemple, l'installation d'une topologie d'arbre est facile mais sa robustesse est très faible. C'est pourquoi il faut utiliser des mécanismes bien plus complexes pour éviter les boucles et pour recréer l'arbre en cas de panne. Les avantages d'avoir un réseau recouvrant avec une topologie libre et contrainte seulement par le réseau sur lequel elle est déployé (l'Internet dans notre travail) sont :

1. la facilité d'ajouter ou d'enlever des noeuds sans interrompre la connexion,
2. l'équilibrage de charge et la robustesse assurés par des liens redondants.

D'un autre côté, la topologie libre de ce type de réseau exige un système du routage approprié que notre architecture peut fournir. En outre, en utilisant notre intergiciel on sépare le routage et l'adressage des noeuds [SMLL06]. Donc, les applications peuvent :

- fonctionner sur des espaces d'adressages privés ou publiques (même sur des réseaux privés qui ont une adresse de recouvrement),
- contrôler la mobilité des noeuds sans le support de la couche réseau,
- assurer la mobilité des noeuds avec des connexions sécurisées,
- utiliser le multipoint dans la couche application,
- employer la sécurité basée sur le nom au lieu de celle basée sur l'adresse.

L'utilisation de notre architecture permet aux applications d'être déployées sur la topologie hétérogène actuelle d'Internet (routeurs NAT, tunnels IPSec, etc.) dans un vrai mode bout-en-bout (nécessaire pour des applications P2P par exemple) tout pouvant utiliser simultanément la mobilité, le multipoint et la sécurité et cela sans transition malgré la dynamique du réseau.

D'un point de vue théorique, beaucoup de travaux montrent qu'il est important de séparer l'adressage du nommage [FYT91, AWSBL99, Mag03b]. Par contre, plusieurs problèmes peuvent être résolus si cette propriété était assurée par le protocole IP. Plusieurs protocoles expérimentés ont proposé des solutions pour assurer un adressage indirect (INS [AWSBL99], INPL [FG01] et i3 [SAZ+02]) et parmi ces protocoles on trouve ceux qui ont traité la mobilité des hôtes (TCP-Migrate [SB00] et Tribe [VdaFR03]). Toutes ces solutions créent une certaine forme de réseaux recouvrants pas toujours au niveau de la couche application afin de résoudre des problèmes tels que l'adressage et la mobilité uniforme. D'un autre côté, les réseaux recouvrants au niveau application qui supportent le multipoint sont aussi construits et implémentés (Narada [hCRSZ01], GENTIL

[BKK+03b] et ROMA [KB04]) afin d'être déployés au-dessus des réseaux qui n'ont pas de protocoles multipoint. Ces réseaux recouvrants peuvent également être conçus pour fournir de nouveaux services tels que la connexion résiliente (RON [ABKM01]) et la recherche Pair-à-Pair (Chord [SMK+01], Pastry [CDG+02a], etc.) et ainsi offrir plus d'opportunité pour des applications.

Voici maintenant un bref résumé des chapitres de ce manuscript. Le deuxième chapitre représente un état de l'art des réseaux recouvrants. Nous commençons tout d'abord par définir le concept des réseaux recouvrants. Ensuite, nous poursuivrons par une étude détaillée sur les deux catégories principales de ces réseaux (structuré et non-structuré). Enfin, nous exposerons quelques fonctionnalités des réseaux recouvrants à traves lesquelles nous pourrions mieux comprendre leur structure et leurs caractéristiques principales et donc avoir une base sur laquelle notre travail évoluera.

Dans le troisième chapitre nous présenterons les réseaux pair-à-pair. Les différentes étapes de l'évolution de ces réseaux seront détaillées. Puis, nous décrivons quelques applications des réseaux P2P qui sont utilisées pour la recherche sur Internet. Nous finissons ce chapitre en présentant plusieurs applications des réseaux recouvrants tout en analysant les différentes architectures permettant la prise en charge de ces applications.

Après avoir présenté la partie théorique de notre travail nous allons présenter dans le quatrième chapitre notre principale contribution dans cette thèse. Nous décrivons dans ce chapitre notre architecture conçue pour réseaux recouvrants appelée DHARMA (Dynamic Hierarchical Addressing, Routing and naMing Architecture). Cette architecture logicielle est basée sur un modèle qui offre plusieurs avantages au niveau applicatif (mobilité, sécurité, multipoint, etc.) et qui seront détaillé dans la suite. Notre intergiciel extensible, résilient et autonome est basé sur une plateforme d'adressage, de routage et de nommage hiérarchique et distribuée. Afin d'évaluer la performance de notre architecture, nous terminons ce chapitre en présentant les différentes résultats des expérimentations que nous avons réalisé et qui montrent son efficacité et sa robustesse pour déployer et maintenir des réseaux recouvrants sur Internet.

Dans le cinquième chapitre, nous allons présenter notre deuxième contribution dans cette thèse. Cette contribution s'agit de proposer une application P2PTV qui emploie notre architecture DHARMA afin d'améliorer la performance des transmissions des flux multimédia en surmontant les obstacles rencontrés par le NAT. Nous commencerons ce chapitre par une étude théorique des systèmes P2PTV qui se basent sur la technique IPTV où les différents aspects théoriques et les problèmes liés à la capacité de téléversement et au NAT d'un système P2PTV seront présentés. Ensuite, nous allons aborder l'extensibilité de la redistribution dans les réseaux P2PTV en détaillant quelques caractéristiques des pairs dans ces réseaux sur lesquels notre travail est basé. Enfin, nous allons présenter les résultats des expérimentations que nous avons effectué en utilisant des données réelles à partir d'un système P2PTV commercial. A travers ces simulations nous prouvons qu'en utilisant notre architecture DHARMA nous pouvons surmonter le problème de la compatibilité du NAT en obtenant des meilleurs résultants et nous pouvons ainsi améliorer la performance du système et le rendre extensible.

Enfin, dans le dernier chapitre nous terminerons par donner les conclusions générales où nous récapitulons nos principales contributions et nous présentons quelques directions pour les futurs travaux qui permettraient d'étendre de manière substantielle la portée de nos travaux.

Chapitre 2

Etat de l'art des réseaux recouvrants

2.1 Introduction

Dans ce chapitre, nous parlons des principes généraux des réseaux recouvrants, en commençant par donner une définition des réseaux recouvrants, puis nous présentons les deux types principaux (structuré et non structuré) en comparant leurs caractéristiques principales. Nous détaillons ensuite les fonctionnalités des réseaux recouvrants ainsi que la construction d'un réseau recouvrant et le mécanisme qui explique comment les noeuds communiquent entre eux.

2.2 Organisation de l'Internet

Internet a connu une croissance exponentielle au cours des dernières années et il est constitué de nombreux réseaux autonomes. Il n'y a donc pas d'organisation centrale ou hiérarchique régissant Internet. Pourtant, il existe des organisations internationales chargées de faciliter la collaboration de ces réseaux. Internet désigne actuellement l'ensemble des réseaux et des machines accessibles par TCP/IP. Il n'est pas simple de définir une délimitation précise avec d'autres réseaux, utilisant en partie les liaisons Internet et en partie d'autres mécanismes, surtout du fait des passerelles (gateways). La plupart des services Internet nécessitent des liaisons TCP/IP, mais des services tels que la messagerie électronique ou le transfert des données sont également possibles à partir et vers d'autres réseaux.

Chaque machine dans Internet est identifiée par une adresse IP qui permet aux autres machines de lui envoyer des données. Le routage entre les machines est organisé par des domaines différents, chacun est un système autonome (AS). Chaque AS organise et fait fonctionner un réseau ou un groupe de réseaux (université, entreprises, fournisseurs d'accès Internet (ISP), etc.). Actuellement, les particuliers utilisent des modems (128 Kbits/s - 3 Mbits/s), des lignes DSL (128 Kbits/s - 2 Mbits/s), ou des connexions par câble (128 Kbits/s - 3 Mbits/s). Les entreprises, les universités, et les fournisseurs d'Internet peuvent se connecter à plusieurs ISPs et avec des vitesses plus élevées (T1 (1.5 Mbits/s) ou T3 (45 Mbits/s)). La figure 2.1 suivante montre un exemple de connexions où l'on peut voir des réseaux qui se connectent entre eux afin que les connexions entre les utilisateurs restent transparentes.

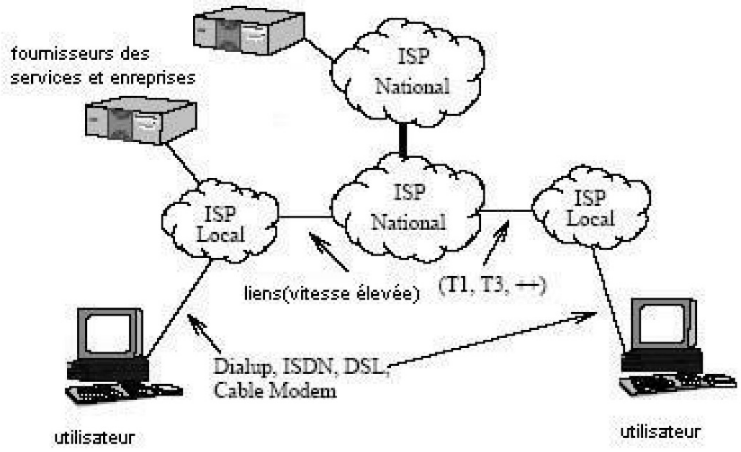

FIG. 2.1 – Exemple des connexions Internet entre plusieurs réseaux.

2.3 Architecture des réseaux recouvrants et Internet

Comme nous l'avons expliqué dans le paragraphe précédent, Internet est organisé comme un système indépendant, autonome et opérationnel composé d'AS. Dans cette architecture, les informations qui concernent le routage sont maintenues dans chaque AS et les réseaux les gèrent en utilisant un fournisseur de services. Les informations partagées avec les fournisseurs et d'autres AS sont organisées en utilisant le protocole BGP-4 qui fonctionne entre les AS [RL95].

Bien que les protocoles décentralisés aient été conçus pour passer à l'échelle, en ne reposant pas sur une connaissance globale de la topologie, ils sont souvent confrontés au problème du surcoût des communications élevées induit par leur processus de raffinement. En effet, dans de tels protocoles [hCRZ00, Hel02, LM03, TAC05], les noeuds maintiennent des positions relatives par rapport à la source de l'arbre de transmission. Périodiquement, chaque noeud essaie d'améliorer sa position en recherchant un meilleur parent, i.e., un noeud non descendant qui lui fournirait un meilleur délai à la source. Ce type de processus de raffinement, ne permet pas à la structure du réseau recouvrant de passer à l'échelle, d'autant plus si on considère une certaine dynamique au niveau des adhésions des noeuds au réseau ou au niveau des conditions du réseau sous-jacent. Un coût supplémentaire des messages de contrôle sera induit lors des opérations périodiques de maintenance et de réparation des structures. D'un autre côté, une fréquence plus élevée d'émission des messages de contrôle sera nécessaire pour transposer des variations des caractéristiques de la topologie physique vers la topologie virtuelle.

L'extensibilité du routage à grande échelle peut être accomplie avec un coût élevé et avec une fiabilité moins importante pour les communications de bout-en-bout entre les hôtes. Ce coût peut être encore plus élevé car BGP est extrêmement limité par le nombre des liens Internet. Les mécanismes employés afin de faire face à ce problème sont parfois assez lents (i.e., quelques minutes) [BKK+03a]. De plus, en cas de panne, les communications peuvent être fortement perturbées [CDGN01, Pax97b, Pax97a]. Le résultat est qu'Internet est très vulnérable concernant les pannes de liens, les problèmes de routage et les erreurs de configuration, ce qui peut affecter la connectivité fournie par les fournisseurs d'accès Internet (ISP) [NAN99].

Donc, avec un réseau recouvrant (overlay) qui est un réseau *virtuel* créé au dessus d'un autre réseau, les noeuds se comportent comme des routeurs pour envoyer les données, et les connexions entre ces noeuds sont établies sur le réseau de base qui est Internet dans notre cas. Ce genre de réseau est capable d'améliorer la fiabilité perçue par les applications. Si nous prenons l'exemple du réseau recouvrant résilient RON [ABKM01] que nous pouvons employer de plusieurs manières comme dans le cas d'un programme de visioconférence où il y a un lien direct avec une bibliothèque RON d'une manière transparente, nous pouvons remarquer que ce type de réseau recouvrant est créé entre tous les participants dans la conférence en utilisant les taux de perte ou les débits comme matrices afin d'établir des liens entre eux. Les utilisateurs assistant à une vidéo conférence ou à une diffusion d'événement s'attendent à une qualité acceptable dès lors qu'ils rejoignent la session multipoint. Puisqu'un arbre de transmission recouvrant multipoint est étudié dans le but de minimiser le délai moyen induit observé par les récepteurs, nous considérerons que cet arbre est efficace si son délai moyen induit est inférieur à une valeur seuil. Nous nous attendons de ce fait à ce que les approches qui se basent sur des raffinements incrémentaux nécessitent un long délai avant que l'arbre de transmission ne converge vers une structure efficace. De plus, un administrateur peut utiliser une application basée sur un routeur RON pour former un réseau recouvrant entre plusieurs LANs comme un réseau recouvrant VPN. Cette idée peut être aussi employée pour développer un ISP recouvrant qui est formé en liant plusieurs ISP traditionnels après achat de bande passante à ces ISP. En utilisant la machinerie de routage RON, un ISP recouvrant peut fournir aux clients un service Internet plus fiable et plus résistant aux pannes.

2.4 Contexte des réseaux recouvrants

2.4.1 Définition

Le réseau recouvrant est un ancien concept dans le domaine des réseaux. L'Internet, lui-même, était déployé au début comme réseau recouvrant au-dessus du réseau téléphonique en utilisant les câbles téléphoniques pour connecter les hôtes et les routeurs. Actuellement, un grand nombre des connexions Internet fonctionnent encore ainsi. Les réseaux recouvrants modernes fonctionnent de la même manière mais ils utilisent les chemins entre les hôtes comme "*liens*" sur lesquels ils transférent les données, ils construisent donc ainsi un réseau sur un autre réseau. Ce type de réseau permet aux concepteurs d'utiliser leurs propres algorithmes de routage et de gestion de données sur Internet. D'un

autre côté, au lieu de passer des années à développer et à modifier les routeurs Internet, les réseaux recouvrants peuvent être utilisés pour déployer de nouvelles fonctionnalités quasiment immédiatement. Ils permettent aussi aux développeurs de profiter d'une plate-forme flexible et efficace pour créer des services.

Nous pouvons définir les réseaux recouvrants comme des topologies virtuelles construites au niveau utilisateur au-dessus d'une infrastructure déjà existante et qui fonctionnent comme un intergiciel entre les applications au niveau utilisateur et les services réseaux basiques. Ces topologies sont construites en choisissant un sous-ensemble de noeuds dans l'infrastructure et en les connectant avec des liens virtuels. Le fait qu'ils soient virtuels permet aux réseaux recouvrants de créer des réseaux dynamiques, extensibles et décentralisés. Cela permet aux réseaux virtuels de coexister [PST04] sans faire aucun investissement dans la couche physique de l'infrastructure [TT04]. Les raisons principales pour utiliser un tel intermédiaire sont les fonctionnalités requises par les nouveaux services et la performance améliorée qui est offerte par des protocoles spécifiques qui peuvent être déployés dans un réseau recouvrant et cela sans besoin de faire des changements dans l'infrastructure.

Plusieurs réseaux recouvrants peuvent fonctionner indépendamment sur la même infrastructure de base où chacun a une topologie virtuelle spécifique. Donc, un noeud dans le réseau de base peut faire partie de plusieurs réseaux recouvrants.

Comme nous l'avons vu, un réseau recouvrant est un réseau construit entre des hôtes sur Internet qui effectue son propre routage et transfère des données en utilisant Internet comme moyen de transport. Il peut également être utilisé pour détecter et masquer rapidement les pannes et pour faire face aux attaques qui menacent la sécurité du réseau. Les hôtes d'un réseau recouvrant participent au protocole de routage et coopèrent pour transférer les données jusqu'à la destination, ce qui permet de créer des applications capables de résoudre le problème de la recherche distribuée [BKK+03a] sans l'assistance du réseau ou celle des systèmes centralisés. La figure 2.2 montre un exemple simple d'un réseau recouvrant.

2.4.2 Fonctionnement des réseaux recouvrants

Comme nous avons vu précédemment, les réseaux recouvrants sont déployés entre des hôtes qui coopèrent entre eux sans besoin de changer l'infrastructure d'Internet. Par conséquent, ils peuvent être déployés rapidement et d'une manière incrémentale. Par contre, il est difficile de changer l'infrastructure du routage IP à cause de la grande quantité des équipements IP installés et que l'on doit changer, ce qui explique le fait que le déploiement d'IPv6 est encore limité 10 ans après son apparition [Hin96].

Avec les réseaux recouvrants il est possible d'effectuer un routage global en rajoutant une moindre quantité d'information dans chaque hôte et en prenant seulement des décisions locales. Comme le fonctionnement pourvu traditionnellement par les serveurs est distribué entre les noeuds du réseau, il est important de voir chaque noeud comme un petit serveur ou une base de données. Ces noeuds peuvent s'organiser comme un réseau ad hoc pour former un réseau recouvrant logique au lieu d'avoir besoin de l'infrastructure de base.

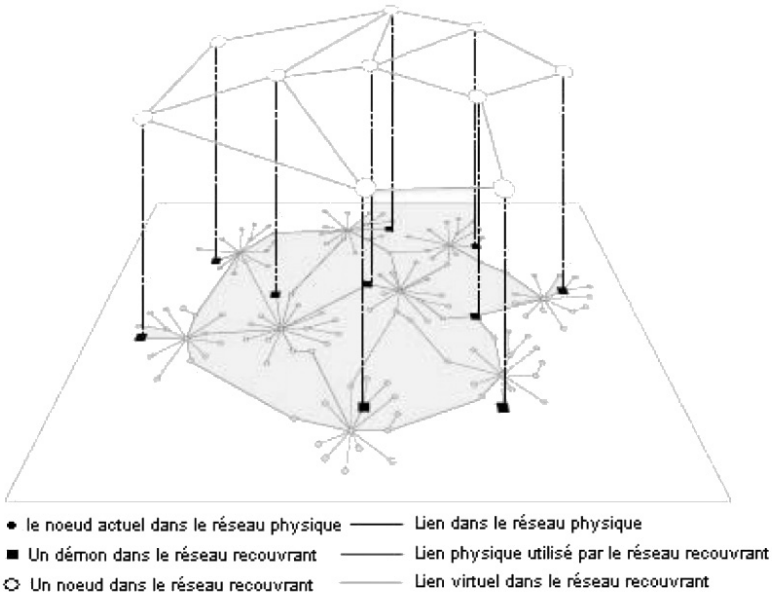

FIG. 2.2 – Exemple d'un réseau recouvrant.

En général, les réseaux recouvrants partagent les caractéristiques suivantes :
– Recouvrement des données garanti.
– Equilibrage de charge automatique.
– Auto organisation.

Comme les réseaux recouvrants peuvent identifier leurs voisins par les données qu'ils contiennent, ils peuvent changer le mode de recherche utilisé (e.g., l'algorithme qui utilise la méthode du graphe traversé traditionnel [MS90]) par le processus de localisation itérative [WDY05]. Dans ce processus, chaque saut fait avancer la requête vers la destination en se basant sur des fonctions mathématiques. Cela réduit la charge du réseau et rend le processus de la requête déterministe. Autrement dit, un réseau recouvrant fonctionne comme une table de hachage distribuée avec l'insertion de la clé, les requêtes et le renvoi des données. Ces paramètres peuvent être obtenus à travers le contenu des noeuds en utilisant des algorithmes de hachage tel que l'algorithme de hachage sécurisé SHA-1 [KAHC05].

La méthode de la connectivité utilisée dans les réseaux recouvrants est différente de celle obtenue en utilisant l'algorithme TTL [MKG+03], car la première est purement structurée et symétrique. Sa structure est basée sur plusieurs fonctions mathématiques qui déterminent comment les noeuds se connectent. La structure du réseau détermine aussi le nombre de tentatives fixé pour faire la recherche dans le réseau. En cas d'échec, les réseaux recouvrants utilisent des mécanismes qui permettent de récupérer et de recréer ou maintenir la structure appropriée du réseau.

2.4.3 Exemple d'un réseau rcouvrant

Parmi les premiers réseaux recouvrants utilisés, citons le réseau expérimental basé sur le modèle OSI (EON) [SAR99] où on a proposé un réseau recouvrant sur le réseau IP qui permet de faire des expérimentations avec la couche réseau dans le système OSI. Le schéma était uniquement expérimental et il n'a pas mené à un déploiement pratique ou à des nouveaux services ou de protocoles. D'un autre côté, le service IP multipoint [AMS+03, APM+04] a été proposé il y a plus de dix ans pour ajouter un autre modèle que celui du point-à-point. Malgré les avantages de ce modèle dont la simplicité et l'efficacité, plusieurs problèmes cruciaux [ASS04, AJY00] ont empêché le déploiement global du multipoint IP. Notons que plusieurs réseaux recouvrants sont extrêmement génériques tel que le RON [ABKM01], le Dynabone [HDA05], et le Yoid [CDK+03]. Ces systèmes existent uniquement pour fournir un réseau recouvrant avec des propriétés meilleures que celles du réseau sous-jacent, comme le délai et la forte résistance contre les attaques DoS.

Prenons l'exemple d'un réseau recouvrant où les noeuds veulent diffuser l'identificateur du stockage comme dans un système de base de données distribuées. Dans ce cas, le noeud dont le contenu est exposé, est un identificateur où on le définit comme un nombre entier positif et on passe ainsi l'étape du hachage. Si les identificateurs sont communs et uniques, des règles simples permettant de créer une topologie d'un réseau recouvrant :

- Chaque noeud dans le réseau recouvrant a deux voisins : le noeud dont la valeur est le prochain nombre entier valable (supérieur) et le noeud dont la valeur est le précédent nombre entier valable (inférieur).
- Si le noeud actuel est l'identificateur supérieur ou inférieur du réseau, un de ces voisins aura la valeur inverse existant dans la fourchette valable du noeud (la valeur la plus haute ou la plus basse).
- Pour joindre le réseau, un noeud doit lancer une requête "*hors bande*" pour trouver un autre noeud du réseau. Ensuite, le nouveau noeud peut utiliser la fonction de recherche pour trouver l'endroit où il va s'installer.

Le processus de recherche est simple : le noeud qui lance la requête détermine la relation entre sa valeur et la valeur de destination. Si la valeur de la destination est supérieure à celle du noeud, le noeud passe la requête à son voisin qui a la valeur la plus élevée. Dans le cas contraire, il le passe au noeud qui a la valeur inférieure. Le processus continue jusqu'à arriver au noeud destinataire qui répond à son tour directement à la source en envoyant son adresse physique.

Le figure 2.3 montre un réseau recouvrant hypothétique construit en utilisant l'algorithme expliqué ci-dessus. Comme les noeuds dans le réseau recouvrant sont connectés en utilisant leurs contenus, les requêtes peuvent être routées efficacement vers la destination.

Cet exemple est théorique et non réel car le temps de recherche est borné et non linéaire (le nombre maximal des sauts est N/2), ce qui provoque un nombre inacceptable de temps de recherche. De plus, il ne réussit pas à traiter les différentes boucles crées par les noeuds manquants dans la topologie. Cependant, cet exemple montre comment les noeuds

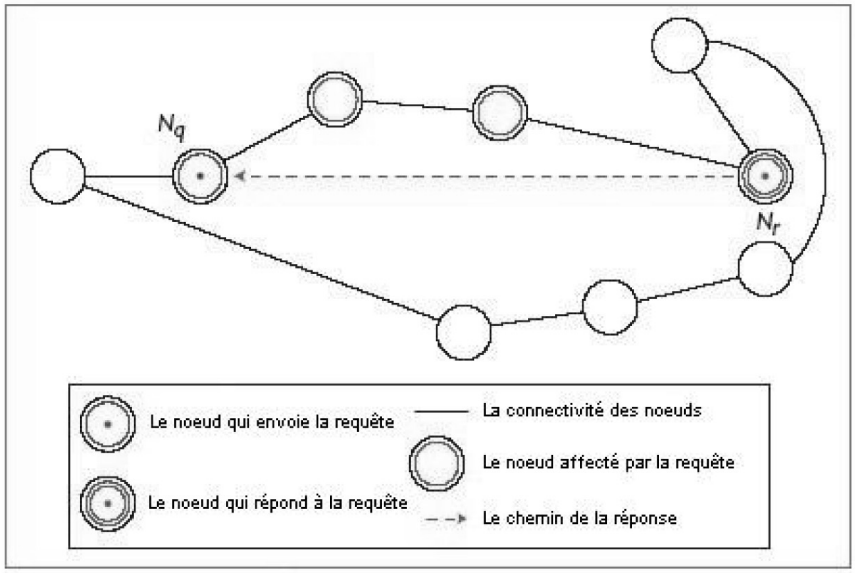

FIG. 2.3 – Exemple simple d'une requête dans un réseau recouvrant.

peuvent utiliser leurs contenus pour s'organiser en suivant quelques règles simples. Par contre, dans le monde réel, les réseaux recouvrants utilisent des règles bien plus complexes pour organiser les noeuds.

D'un autre côté, l'algorithme Chord [SMK+01] montre un exemple plus pertinent où chaque noeud dans le réseau a un seul successeur, donc on y définit une topologie globale sous une forme d'un cercle (comme nous l'avons expliqué dans notre exemple mais *unidirectionnel*). Cet algorithme implique une seule connexion par noeud ce qui le rend résilient au changement (nouveau noeud, panne, etc.). [SMK+01] remplace le noeud basique avec un groupe des noeuds plus distants et cela selon la règle mathématique suivante : pour une valeur n, les voisins seront ceux qui correspondent à $n + 2^0$, $n + 2^1$, $n + 2^2$, ...,$n + 2^m$ où m est le nombre des octets dans l'identificateur (2^m détermine le nombre maximal des noeuds). Pratiquement, les noeuds dans ce type du réseau sont définis par chaque successeur dans la boucle $(n + 2^{k-1})$ mod 2^m où 1<k<m.

Les réseaux recouvrants actuels sont utiles pour les applications qui exigent la fiabilité, l'extensibilité et l'auto organisation comme les bases des données distribuées et les applications dont la méthode de recherche est déterministe. La plupart de ces systèmes sont libres (open source) où les développeurs peuvent les utiliser directement pour construire des applications distribuées.

2.4.4 Avantages et inconvénients des réseaux recouvrants

Récemment, les réseaux recouvrants ont attiré l'attention dans le domaine du commerce et dans la communauté de la recherche grâce à leur capacité d'offrir des services supplémentaires autres que ceux qui sont offerts par le réseau sous-jacent Internet. De plus, ils offrent plus de flexibilité, d'organisation et forment un réseau logique au niveau de la couche application dont le but est de fournir des services particuliers aux utilisateurs. Parmi ces services citons la distribution du contenu [AWtTW02], le partage des fichiers [PF03], le multipoint [KB04], la qualité de service [SSBK04], la transmission des flux multimédia [PWCS02], la tolérance de panne [ABKM01], le routage amélioré [SSK99].

2.4.4.1 Avantages

Les réseaux recouvrants permettent de :

- Fournir de nouveaux services, ce qui n'est pas possible à implémenter dans l'infrastructure actuelle du réseau sans utiliser des applications spécifiques.
- Avoir plus de performance pour les services existants en utilisant des algorithmes qui offrent une flexibilité avec plus d'information sur les ressources du réseau.

De plus, parmi les avantages des réseaux recouvrants, ces réseaux offrent un déploiement plus rapide et plus facile au lieu de penser à modifier les protocoles et les routeurs actuels d'Internet. Le fait d'essayer de changer des millions de machines qui utilisent l'infrastructure IP représente un vrai challenge. Donc, au lieu de faire un changement dans la couche IP, plusieurs recherches étudient actuellement des protocoles qui fonctionnent dans un réseau recouvrant.

Un autre avantage des réseaux recouvrants est qu'ils peuvent éviter de charger le réseau sous-jacent avec des fonctions qui peuvent être plus performantes dans les couches les plus hautes. De plus, il se trouve que dans ces réseaux les bogues sont remarquablement moins graves que dans l'infrastructure de base. De plus, des systèmes comme MONET [ABKR05] peuvent choisir un chemin basé sur un serveur WEB sans que les routeurs d'Internet comprennent quoi que ce soit sur les protocoles des niveaux supérieurs.

Enfin, les réseaux recouvrants peuvent traiter plusieurs connexions vers plusieurs destinations où il y a pour chaque connexion un nombre limité de sauts intermédiaires dans le réseau. D'un autre côté, ils peuvent offrir des routes différentes pour une seule destination tout en dépendant de la performance (délai, débit, taux de perte, etc.) que chaque application réclame [ABKM01]. Ils peuvent également fournir une meilleure performance de bout-au-bout en faisant le routage à travers des noeuds recouvrants intermédiaires, ce qui permet de transférer les flux dans des chemins bout-au-bout ce qui n'était pas permis par les ISPs [SCH+99].

2.4.4.2 Inconvénients

Les deux principaux inconvénients des réseaux recouvrants sont :

– Tout d'abord, les réseaux recouvrants représentent un niveau intermédiaire entre l'application et l'infrastructure du réseau car dans chaque noeud intermédiaire, les communications doivent être gérées afin de contrôler les données et de les transférer plus tard vers les autres noeuds. Autrement dit, les routeurs dans un réseau recouvrant subissent un "overhead" à cause de la gestion du réseau et du traitement de chaque paquet parce qu'il faut d'abord livrer les paquets au niveau applicatif, les traiter puis les transférer au routeur prochain.

– Deuxièmement, le déploiement des noeuds des réseaux recouvrants dans la topologie physique n'est pas toujours optimal, ce qui ne permet pas d'accéder aux noeuds intermédiaires car le développeur du réseau recouvrant a rarement le contrôle sur le réseau physique ou bien il n'a même pas de connaissances sur sa topologie actuelle. Par contre, les réseaux recouvrants peuvent parfois trouver des chemins alternatifs ce qui améliore la latence des connexions de bout-en-bout. Cependant, dans la plupart des cas, les chemins trouvés par les réseaux recouvrants avec des sauts multiples ont une latence plus élevée que dans le cas des connexions de type point-à-point sur Internet.

D'ailleurs, il se trouve que le routage sur Internet peut engendrer une perturbation et une dégradation de la performance [ABKM01]. L'unique chemin entre la source et la destination n'est pas forcement toujours valable tandis que les règles du routage et du trafic imposent parfois une pénalisation relativement lourde sur la performance des connexions de bout-en-bout [AMS+03].

Cependant, les réseaux recouvrants sont relativement petits par rapport au réseau sous-jacent global (Internet), donc les protocoles qui les gèrent peuvent non seulement surmonter ces inconvénients, mais ils peuvent aussi offrir une meilleure performance aux applications, car les réseaux recouvrants offrent un contrôle complet aux protocoles qui fonctionnent entre les noeuds participants.

2.5 Types de réseaux recouvrants

2.5.1 Non structuré

Les systèmes non structurés sont simples à implémenter et à maintenir comme le prouve le succès et la popularité des implémentations fonctionnant sur Internet. Les réseaux recouvrants non structurés comme Gnutella [PF05], Kazaa [Pap01] et FreeNet [CS+01], malgré leur efficacité et leur large exploitation, ne sont pas vraiment optimaux ce qui conduit les chercheurs à essayer d'arriver à avoir des systèmes optimaux tout en maintenant les caractéristiques qui garantissent leur succès.

Dans ce type de réseaux, les noeuds sauvegardent leur contenu indépendamment sans être reliés aux autres ressources et ils se connectent avec le minimum de contraintes.

Par contre, il existe une relation proportionnelle entre la probabilité de la localisation du contenu dans le système et le facteur de duplication (la zone dans laquelle une partie du contenu est copiée).

Gnutella et les autres systèmes implémentent deux niveaux hiérarchiques de feuilles et de "SuperPeer" [SR02] (les noeuds qui ont une grande bande passante et une latence de faible connectivité). Par contre, les feuilles ont généralement une faible connexion et sont souvent derrière des pare-feux. Ils utilisent un mécanisme de démarrage pour se connecter de manière aléatoire aux SuperPeers. Chaque SuperPeer maintient un maximum de feuilles qui va servir pour essayer de se connecter à plusieurs SuperPeers. Par conséquent, la feuille établit un groupe de deux ou quatre connexions dans un réseau SuperPeer. La figure 2.4 montre un exemple d'un réseau Gnutella. Nous avons remarqué que quand une feuille se connecte, elle envoie une partie du fichier de noms et chaque SuperPeer maintient alors une partie du fichier des noms enregistrée dans ses feuilles.

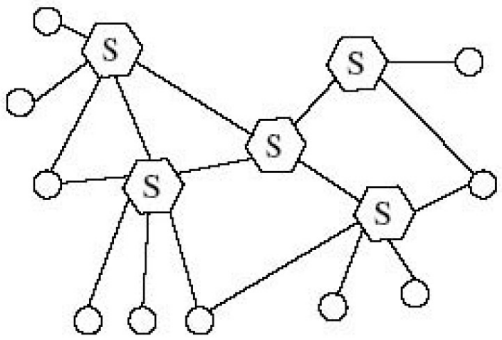

FIG. 2.4 – Exemple du réseau gnutella.

Les réseaux recouvrants non structurés n'ont pas vraiment pour but d'avoir une topologie fixe mais ils ont tendance à avoir une structure aléatoire. Ces topologies sont souvent inefficaces pour trouver les éléments rares où les mécanismes de recherche embarqués utilisés sont coûteux [LRS02]. Tandis que dans les scénarios où la distribution des requêtes est non uniforme, les réseaux recouvrants structurés peuvent être plus efficaces.

2.5.2 Structuré

Malgré la popularité des réseaux recouvrants non structurés, ces derniers manquent de garanties fiables et sont souvent reliés au nombre de requêtes. Pour surmonter ces inconvénients, on emploie le terme de réseaux recouvrant structurés [BKK+03a] . Prenons l'exemple des réseaux Chord [SMK+01], Tapestry [ZKJ01, GBL+03, RFaKS00, RD01] et Pastry [ZHS+04] les plus connus comme réseaux recouvrants structurés qui sont généralement construits en utilisant une table de hachage distribuée (DHT). Contrairement aux réseaux recouvrants non structurés, ces réseaux contrôlent les noeuds et les identificateurs de contenu qui déterminent d'ailleurs leur placement dans la topologie et cela en utilisant la notion de table de hachage distribuée. D'un autre côté, les réseaux recouvrants structurés imposent une forte homogénéité, les noeuds maintiennant un nombre constant de connexions et les quantités de contenu correspondantes. Dans la plupart des implémentations, le succès des requêtes est garanti après O(log n) sauts dans le réseau recouvrant si le contenu existe.

Dans les réseaux recouvrants structurés, le contenu est copié par un certain nombre de noeuds où les données sont souvent mutables (modifiables). Comme les réseaux recouvrants structurés utilisent la notion de localisation selon le contenu, les noeuds n'enregistrent pas forcement leur contenu et les concepteurs de ces réseaux se basent sur le fait qu'un noeud enregistre le contenu des autres noeuds. Les noeuds sont donc disposés à enregistrer le contenu des autres utilisateurs dans le système car ces derniers dépendent également des autres noeuds pour enregistrer leur contenu. Ngan *et al.* proposent une approche où les noeuds publient leur consommation de ressources et leur contribution au système [NWD03]. Dans cette approche, les noeuds sont motivés à publier leur ressource d'une manière équitable. Cependant, ce n'est qu'un problème parmi plusieurs autres problèmes d'équilibrage dans les réseaux recouvrants structurés. Comme le contenu est attribué aux noeuds en fonction d'un mécanisme de hachage cohérent, le noeud peut être sollicité aléatoirement pour enregistrer une partie du contenu afin de faire partie du système P2P. Les méthodes actuelles utilisent des mécanismes d'équilibrage de charge afin d'éviter la saturation des noeuds. Ce problème a commencé à être étudié plus sérieusement surtout dans les services DHT proposés comme dans OpenHash [KRRS04].

2.5.3 Comparaison des réseaux recouvrants structurés et non structurés

Malgré la capacité des réseaux recouvrants structurés, Chawathe *et al.* [BKK+03a] identifient succinctement plusieurs désavantages de l'application la plus populaire : le partage des fichiers.

Le premier inconvénient concerne le placement strict du contenu quand on prend en compte les changements des caractéristiques d'un système de partage des fichiers utilisé sur Internet [GSG02, RFI02]. L'allocation du contenu d'une manière uniforme dans le système exige une fonction consistante de hachage pour chaque partie du contenu afin de déterminer les noeuds à enregistrer (e.g., dans les noeuds de Chord, le contenu est enregistré en fonction d'un identificateur "ring". Pour cela, les noeuds doivent enregistrer le contenu des autres noeuds pour maintenir les invariants de la table dynamique de hachage

(DHT) qui garantit une recherche efficace avec une complexité minimale. D'un autre côté, à cause du mécanisme de duplication des noeuds, le système subit le coût des communications et des traitements afin de re-localiser le contenu des noeuds. Le deuxième inconvénient est que les recherches des clés sont plus significatives dans les réseaux de partage des fichiers par rapport à la conformité exacte des requêtes sur laquelle se basent les réseaux recouvrants structurés.

Contrairement aux réseaux recouvrants structurés, les réseaux recouvrants non structurés sont simples à construire et à maintenir. Ils utilisent essentiellement des routes aléatoires et emploient une construction de recouvrement et un algorithme de maintenance plus ou moins trivial. Pour palier à ces défauts, les réseaux recouvrants structurés sont conçus sur une structure particulière qui permet d'atteindre efficacement un objet, mais en contrepartie demande une construction et une maintenance nettement plus complexe. Dans ce cas, le système pourrait répliquer chaque fichier sur plusieurs noeuds afin d'en améliorer la disponibilité. Mais il ne faut pas oublier que même si les noeuds sont voisins sur l'espace de l'identification, ils sont réellement distribués physiquement sur Internet. Ainsi, si une panne d'alimentation de toute une ville se produit, et qu'un fichier partagé n'est plus disponible, une ou plusieurs réplications de ce fichier survivent ailleurs sur le réseau recouvrant.

D'ailleurs, d'autres services que le partage de fichiers peuvent aussi être développés sur la base des tables de hachage distribuées. Si l'on considère les applications multipoint, au lieu de construire un arbre multipoint pour une maille, la construction peut se faire à partir des bords du réseau recouvrant structuré, et permet alors d'amortir les coûts de la construction et d'amortissement à travers plusieurs applications et groupes multipoints.

2.6 Fonctionnalités des réseaux recouvrants

2.6.1 Structure des réseaux recouvrants (système des couches)

Les architectures des réseaux recouvrants n'ont pas beaucoup changé depuis l'apparition de leurs origines commençant par USNET [Spa99] et FidoNet [Bus93]. Plusieurs difficultés se sont résorbées dans ces réseaux comme le routage, le stockage et la performance de la recherche tout en suivant la rapidité et la qualité du réseau sous-jacent (Internet).

Afin de mieux comprendre la structure des réseaux recouvrants concernant le système des couches, nous allons prendre l'exemple du protocole du routage Hypercube [LB99]. Afin de séparer la fonctionnalité de la couche du routage existante des avantages de ce protocole, il est nécessaire de les diviser en deux couches comme le montre la figure 2.5.

La couche du réseau sous-jacent est supposée avoir un routage efficace ce qui veut dire qu'elle est capable de délivrer n'importe quel message de la source vers la destination et d'une manière efficace ce qui est le cas de la couche IP (la diffusion à travers Internet est un environnement optimal pour le protocole Hypercube). Comme nous pouvons le voir dans la figure, la couche du protocole de routage Hypercube est située directement au-dessus de la couche du réseau sous-jacent et l'efficacité de cette couche est mesurée

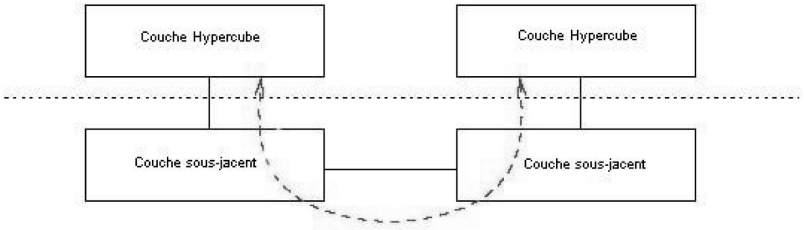

FIG. 2.5 – La division des couches.

par la capacité de trouver la meilleure adresse de destination où la requête sera transférée. Comme nous employons la notion de réseau recouvrant, cette décision doit être prise en utilisant seulement des informations locales de routage.

Le réseau recouvrant fonctionne comme un groupe abstrait où les hôtes jouent un rôle actif dans le processus de localisation des données. Comme nous pouvons le remarquer dans la figure 2.6, les hôtes dans le réseau recouvrant sont un sous-ensemble du nombre total des hôtes.

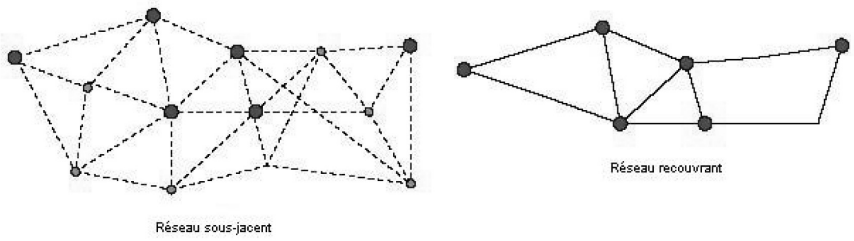

FIG. 2.6 – Le réseau recouvrant et le réseau sous-jacent.

Les chemins entre deux hôtes peuvent être différents dans les deux couches. Un seul bord dans le graphe du réseau recouvrant peut inclure plusieurs bords dans le réseau sous-jacent correspondant. Cette relation entre les deux couches rend le processus de routage plus difficile : le chemin qui peut être optimal dans le réseau recouvrant peut être non optimal dans l'autre réseau. Ce problème est parmi les problèmes les plus importants dans les implémentations actuelles des DHTs. D'un autre côté, la plupart des décisions sont prises à chaque saut en se basant sur la relation de voisinage dans le réseau recouvrant tout en dépendant de la manière dont le réseau est construit, ce qui peut mener parfois à des résultats *catastrophiques* car le voisin d'un noeud du réseau recouvrant peut être situé à l'autre bout du globe. D'un autre côté, le routage entre deux hôtes qui sont proches physiquement l'un de l'autre peut contenir des sauts vers d'autres hôtes bien plus loin. Par conséquent, la performance du protocole peut se dégrader concernant la longueur du chemin du routage et la latence totale de bout-en-bout.

Enfin, l'interaction entre les deux couches peut être importante. Car la couche du routage dans le réseau recouvrant est construite en utilisant l'information prise sur la

topologie locale de la couche du réseau sous-jacent. Autrement dit, chaque fois qu'un noeud veut joindre le réseau, des informations doivent être accessibles localement afin de déterminer le meilleur point dans la structure auquel le nouveau noeud peut être raccordé.

2.6.2 Sélection du voisinage

Les mécanismes utilisés dans les réseaux recouvrants comme Tapestry [ZKJ01] et Pastry [RD01] pour construire les tables de routage prennent en compte la proximité du réseau. Dans ce type de réseau, le facteur principal pour la gestion de la table du routage est la distance entre les noeuds enregistrés dans la matrice de proximité. Ici, on essaie de minimiser cette distance en fonction du préfixe dans l'identificateur du noeud en prenant l'exemple du réseau Pastry qui assure l'invariant pour chaque table du routage.

L'invariant de la proximité est défini par chaque entrée dans la table du routage et fait référence, selon la matrice de la proximité, au noeud le plus proche parmi tous les noeuds existants et cela avec le préfixe de l'identificateur du noeud approprié.

Par conséquent, le message est normalement transféré à chaque étape du routage, en fonction de la matrice de proximité, au noeud le plus proche parmi les noeuds dont l'identificateur partage le préfixe le plus long avec la clé. De plus, la distance parcourue dans chaque étape du routage augmente exponentiellement parce que la quantité des noeuds diminue exponentiellement avec la longueur du préfixe. Donc, nous pouvons définir deux propriétés du réseau recouvrant tout en tenant compte de la localité du réseau de la manière suivante :

- La distance totale parcourue : la distance prévue pour la dernière étape du routage détermine la distance totale parcourue par le message. Par conséquent, la distance totale moyenne parcourue par le message dépasse la distance entre le noeud source et le noeud destinataire d'une petite valeur constante.
- La convergence locale du chemin : les chemins de deux messages envoyés par des noeuds proches avec des clés identiques peuvent converger dans un noeud proche de la source dans l'espace de la proximité. Donc, la probabilité de convergence du chemin augmente dans chaque étape du routage. Par conséquent, cela nous permet de rendre les applications qui utilisent ce réseau transparentes.

Les algorithmes de routage dans les réseaux recouvrants exigent une sélection du voisinage efficace car ils peuvent choisir les éléments proches dans la table de routage parmi un grand groupe de noeuds.

Une autre forme est proposée par CAN [RFH+01] pour la sélection du voisinage où plusieurs noeuds sont groupés dans une zone dans un espace de D dimension. Chaque noeud reçoit régulièrement une liste de noeuds dans la zone voisine et il mesure le RTT chaque fois. Puis, le noeud qui a une valeur RTT minimale sera choisi comme un voisin pour cette zone. Cette technique est moins efficace que celle des réseaux Tapestry et Pastry car ici chaque valeur choisie dans la table du routage (en fonction de la valeur minimale de RTT) représente un petit groupe des noeuds de la zone voisine.

2.6.3 Placement du voisin

Deux approches sont possibles pour déterminer la relation de voisinage entre les noeuds. Pour mieux expliquer ces approches, nous allons considérer un espace à deux dimensions ID. Les deux cas sont montrés dans les figures 2.7 (a, b) :

- Le cas (a) est la relation de voisinage la plus simple : le noeud est utilisé comme une origine d'un système Cartésien de dimension d, et les voisins sont placés tout au long des axes dans le système. Le nombre des voisins dans ce cas est 2d.
- Dans le cas (b), les voisins sont placés en divisant l'espace en 2d quadrants, donc le nombre des voisins est 2d.

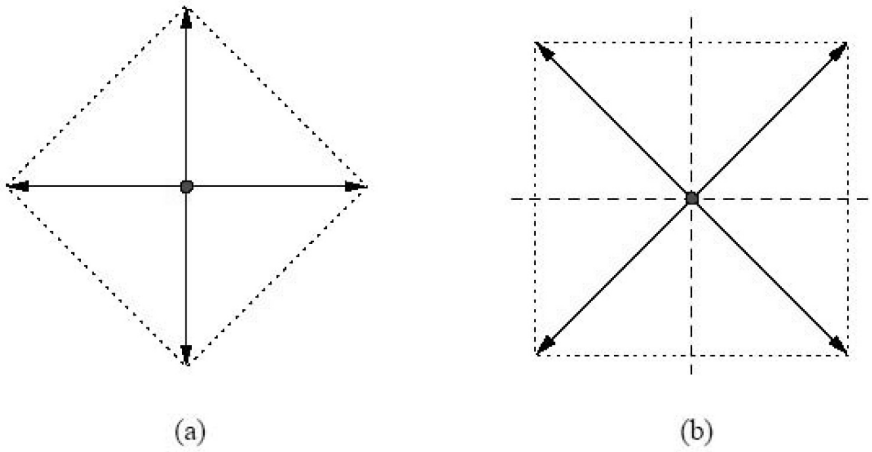

FIG. 2.7 – Les deux cas possibles pour le placement des voisins.

Les figures 2.8 (a, b) montrent une généralisation pour un espace à 3 dimensions :

- Dans le cas (a), les voisins d'un noeud sont placés dans le sommet d'une forme d'octaèdre où le noeud se situe dans son centre.
- Dans le cas (b), les voisins d'un noeud sont placés dans le sommet d'un cube dont le centre est le noeud lui-même. Le nombre des voisins dans ce cas est 2d.

Nous avons remarqué que la deuxième approche est la plus utilisée car elle offre plus de flexibilité concernant la position des noeuds où les voisins ne sont pas obligés d'être placés sur l'axe et peuvent être n'importe où dans le quadrant.

2.6.4 Construction d'un réseau recouvrant

Pour construire un réseau recouvrant, la première tâche à accomplir est de construire le graphe qui représente les caractéristiques de la topologie en question (les noeuds et

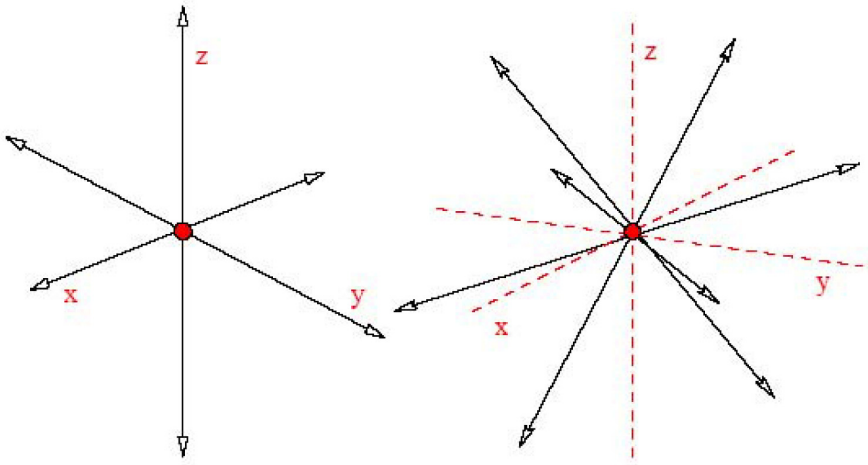

FIG. 2.8 – Les relations du voisinage possible dans un espace de 3 dimensions.

la connectivité entre eux). Donc afin de garantir la meilleure solution, les développeurs doivent prendre en compte les services de base et les opérations qui peuvent être développées dans ce graphe (le routage des messages et la maintenance de la structure du graphe).

Prenons l'exemple de la topologie de l'étoile qui représente la relation entre les services de base et la structure du graphe. Dans cette topologie, le noeud central représente le point sensible dans l'architecture. Pour cela, en construisant le réseau recouvrant il faut prendre en compte les caractéristiques statiques et dynamiques du graphe. De plus, un autre point important qu'on doit prendre en compte pour la construction d'un réseau recouvrant, il s'agit du déploiement des services et ainsi de bien s'adapter aux conditions imposées afin d'avoir une exploitation optimale de ces services [ITU94].

Pour mieux comprendre le processus de construction des réseaux recouvrant, nous allons prendre l'exemple du réseau Gnutella [PF03] qui est un réseau pair-à-pair. C'est un logiciel de partage de fichiers qui créé un réseau recouvrant formant un système de partage de contenu entre les utilisateurs. La caractéristique la plus importante de ce réseau est que tous les noeuds peuvent fonctionner comme clients autant que serveurs et ils peuvent recevoir et envoyer des données. Les noeuds dans ce réseau cherchent le contenu auquel ils s'intéressent et ils le téléchargent auprès des noeuds qui l'ont déjà. Dans ces systèmes, n'importe quel utilisateur peut participer, ce qui rend le système très divers en terme de capacité des ressources des noeuds.

Ce système est très dynamique car les noeuds rejoignent et quittent le réseau fréquemment. De plus, leur topologie peut être soit structurée et les noeuds maintiennent alors une structure particulière en joignant le réseau, ou soit non structurée.

L'étape la plus importante pour construire un réseau recouvrant s'appelle *le processus de démarrage* "bootstrap". Chaque noeud doit l'exécuter avant l'utilisation du système. Dans cette étape, le noeud qui veut joindre le réseau cherche les adresses des noeuds (en ligne) avec lesquels il va établir la connexion et puis il se connecte avec eux. Après cette

étape, le noeud sera prêt à participer dans l'activité de la recherche dans le système.

Le processus de démarrage joue un rôle très important car le temps de ce processus doit être minimal et le résultat (les noeuds voisins) doit montrer une bonne performance dans les activités de la recherche et du téléchargement.

2.6.5 Communications entre les noeuds dans un réseau recouvrant

Pour établir les connexions entre les noeuds dans un réseau recouvrant il existe plusieurs algorithmes, mais quelque soit l'algorithme il faut fournir certaines informations soit à un noeud central qui gère l'opération, soit aux autres noeuds dans le réseau recouvrant.

Dans l'algorithme centralisé [BSR07], tous les noeuds dans le réseau envoient toutes les informations à un seul noeud désigné *noeud de contrôle*. Ce noeud calcule le nombre de connexions demandées pour chaque saut dans le réseau en utilisant cet algorithme, et il envoie la réponse à tous les noeuds. Cet algorithme est considéré efficace seulement pour les petits réseaux recouvrants.

Dans l'algorithme "upstream and downstream tree knowledge" [Kar05], tous les noeuds qui existent tout autour du réseau recouvrant envoient les informations collectées à leurs voisins qui à leur tour les envoient à leurs propres voisins et ainsi de suite jusqu'à ce que chaque noeud reçoive toutes les informations sur l'arbre construit. De la même manière, les noeuds qui existent à l'intérieur du réseau recouvrant envoient les informations nécessaires mais dans l'autre sens cette fois. Dans ce cas, chaque noeud peut calculer les connexions TCP pour effectuer les sauts dans le réseau recouvrant.

D'autres algorithmes n'exigent aucune communication entre les noeuds dans le réseau recouvrant comme l'algorithme "zero knowledge" [Kar05].

2.6.6 Localisation des réseaux recouvrants

Des recherches ont prouvé qu'en connectant les noeuds qui ont les mêmes intérêts, on peut contrôler et minimiser l'inondation des requêtes et optimiser ainsi les systèmes recouvrants. Dans ce contexte, Kleheer *et al.* [KBS02] argumentent la maintenance et la localisation des systèmes P2P distribués.

Les réseaux recouvrants sémantiques (SON) proposés dans [CGM02] développent une classification hiérarchique où un noeud peut être un membre d'un ou plusieurs réseaux recouvrants indépendants. Comme les requêtes sont étiquetées, alors elles sont envoyées vers le réseau recouvrant approprié. La seule difficulté avec cette classification est que sa performance dépend des critères déjà définis pour classifier les données et elle exige que les utilisateurs et les noeuds classifient leurs données aussi.

Le travail de Sripanidkulchai *et al.* [SMZ03] se base sur la localisation pour créer des *raccourcis* où chaque noeud crée une liste de raccourcis pour les noeuds qui ont répondu aux requêtes précédentes. Pour trouver un contenu (un fichier), le noeud regarde d'abord cette liste avant d'envoyer les requêtes à tous les noeuds dans le réseau.

Les réseaux recouvrants fournissent des limites probabilistes quant au nombre de sauts requis pour localiser un objet cherché (log16 N où N représente le nombre des noeuds dans le cas de l'application nommée Pastry [RD01]), chaque saut contribue à un délai supplémentaire. Cela est dû à chaque intermédiaire qui peut se trouver n'importe où sur Internet ; dans le pire des cas, chaque noeud se trouve sur un continent différent. En fait, dans un réseau recouvrant, le délai prévu de chaque saut, est la moyenne des délais de chaque paire de noeuds. Heureusement, en pratique les résultats sont meilleurs et donc l'idée est de choisir chaque valeur dans la table du routage qui représente le noeud le plus proche sur le réseau physique.

2.7 Conclusion

Dans ce chapitre nous avons présenté plusieurs aspects des réseaux recouvrants qui conduisent à avoir une plus ample vue sur leurs fonctionnements. Nous avons parlé des différentes fonctionnalités des réseaux recouvrants tout en étudiant les deux types de réseaux recouvrants, à savoir : les réseaux structurés et non structurés.

Les réseaux recouvrants qui sont des réseaux logiques construits sur un autre réseau physique permettent d'offrir des services qui ne sont pas vraiment exploités dans la couche réseau et qui permettent d'améliorer d'autres services déjà existants. Plusieurs travaux ont été réalisés afin d'essayer de résoudre ces problèmes et à travers ces derniers, nous avons proposé notre architecture qui sera détaillée dans les chapitres suivants.

Les travaux de recherche effectués afin de séparer l'espace d'adressage et de nommage sont situés au niveau de la couche réseau. Mais ce que nous proposons dans ce travail c'est de fournir une architecture d'adressage, de routage et de nommage au niveau application d'un réseau recouvrant sur Internet. Cette architecture sera étudiée en détail dans les chapitres suivants.

Chapitre 3

Applications des réseaux recouvrants

3.1 Introduction

Dans ce chapitre nous allons tout d'abord définir et présenter les réseaux P2P, puis nous allons détailler l'évolution de ces réseaux. Ensuite, nous allons présenter quelques applications des réseaux P2P modernes (Guntella, Chord, pastry, Can) et enfin, nous allons parler brièvement de quelques exemples des réseaux recouvrants (MBone, 6-Bone, ALMI, RON, etc)

3.2 Réseaux pairs-à-pairs

Les réseaux pair à pair (« Peer-to-Peer » ou P2P) [Ora01] sont un domaine de recherche récent et en pleine expansion comme en témoignent les récentes publications sur le sujet [Loo03, BPTU03, Dru02] et leur comportement à grande échelle (plusieurs millions de machines) est encore mal connu [NHB04]. La technologie des réseaux « Peer-to-Peer » peut être utilisée à de nombreuses fins (travail collaboratif, réseaux de proximité ou de secours). Mais leur succès auprès du grand public vient de la possibilité d'échanger différents types de fichiers (musicaux, audiovisuels ou des logiciels en dehors des circuits de distribution traditionnels).

Un réseau P2P est un réseau de systèmes distribués composés de noeuds interconnectés qui sont capables de s'organiser afin de se partager des données et des ressources. Ces noeuds sont capables de s'adapter afin de maintenir la connectivité et permettent d'avoir de bonnes performances.

L'idée générale du terme P2P est simple : il s'agit de s'affranchir du noeud central dans le réseau (architecture décentralisée) afin d'obtenir un système complètement distribué. Le terme de pair se justifie par l'égalité entre les différents noeuds du réseau. Les noeuds n'ont pas de rôle prédéfini (figure 3.1). Ils peuvent prendre tour à tour les rôles de client et de serveur et même être routeur dans le réseau. Etant donné que toutes les machines sont similaires du point de vue des fonctionnalités, elles sont appelées pairs. Cette simple différence autorise des transferts de données depuis et vers n'importe quel pair dans le réseau. Les réseaux de pairs sont des réseaux logiques et non physiques. Ainsi, l'organisation des pairs dans le réseau logique ne reflète pas du tout la topologie du

réseau physique, autrement dit, deux pairs connectés directement peuvent être physiquement éloignés alors que deux physiquement proches ne se sont pas mis en relation.

FIG. 3.1 – Mode P2P.

Dans ces réseaux, il n'y a pas de configuration initiale à effectuer. En effet, un pair qui veut se connecter recherche un autre pair dans son voisinage et accède au réseau par cette voie.

Le principal intérêt de ces réseaux provient de leur architecture décentralisée qui permet, contrairement au mode client/serveur, de récupérer des fichiers directement chez leurs détenteurs. De plus, les fichiers partagés peuvent être fragmentés en blocs, ce qui permet à un utilisateur de récupérer simultanément différentes parties d'un fichier auprès de différentes sources.

Le bon fonctionnement des réseaux P2P est basé sur la participation active de chacun des pairs. Si un pair réclame des ressources, il doit en proposer en contrepartie. Cependant, il est difficile d'éviter les utilisateurs qui ne participent que dans un sens. Une solution consiste à récompenser les bons participants et à punir les mauvais [BLV05].

Ces réseaux doivent fonctionner en présence d'un contrôle centralisé minimal pour assurer l'équité entre les utilisateurs et être le plus résistant possible aux attaques. La difficulté de la conception d'un système P2P provient (1) de l'extrême volatilité des utilisateurs qui se connectent et se déconnectent à volonté (Napster, ancêtre des réseaux P2P), et (2) d'une volonté d'anonymat des utilisateurs qui ne souhaitent pas pouvoir être identifiés lors de l'utilisation du réseau.

De plus, ce type de système ne serait pas viable si la majorité des utilisateurs prenait des ressources sans rien donner. Ce type d'utilisateurs (surnommés « leecher » ce qui signifie sangsue) participe grandement à la diminution de l'efficacité d'un tel système.

Ce système ne peut pas assurer la confidentialité des données échangées et les vitesses de transfert sont aléatoires. Les utilisateurs qui téléchargent en même temps le même fichier doivent partager leur bande passante entre leur activité en tant que clients et serveurs à la fois et limiter la vitesse en chargement sur leur machine. Pour un utilisateur possédant une bande passante de type ADSL, les temps de téléchargement peuvent être relativement longs. De plus, les ressources sur tel système ont généralement une durée de vie parfois limitée. Autrement dit, un fichier accessible sur un réseau P2P peut très bien ne plus l'être dans l'heure qui suit. D'ailleurs, les utilisateurs d'un réseau P2P effectuent de nombreuses

requêtes de recherche et de nombreux transferts de fichiers. Les réponses aux requêtes et les transferts doivent s'effectuer aussi rapidement que possible.

D'une manière générale, on peut distinguer quatre différents types d'architectures P2P possibles :

- Le P2P pur où tous les noeuds ont le même rôle et ils peuvent être serveur ou client à tout moment. Ce réseau nécessite un programme qui doit être installé sur tous les noeuds pour établir la connexion.
- Le P2P *hybride* ou P2P avec serveur d'information qui se charge de lier tous les utilisateurs connectés.
- Le P2P avec serveur d'information contenant l'utilisateur et le contenu où le serveur est au courant de l'ensemble des données partagées par chaque pair.
- Le P2P avec serveur d'information utilisateur et contenu où les données partagées sont sur le serveur lui-même.

Enfin, on peut résumer les différentes caractéristiques du modèle P2P de la manière suivante : la décentralisation confère aux applications P2P et, comparativement au modèle client-serveur, un bon équilibre de charge, une meilleure extensibilité car l'utilisateur a juste besoin de se connecter à un noeud du réseau pour devenir à son tour un membre du réseau, une répartition des coûts de mise en oeuvre et de maintenance du service offert et une bonne tolérance aux fautes. Les autres caractéristiques de ce modèle sont :

1. l'auto-organisation qui permet aux communautés de pairs de délivrer leur service de manière autonome.
2. la dynamicité et la disponibilité des données et des ressources.
3. l'utilisation d'un réseau recouvrant qui abstrait les caractéristiques physiques des éléments.
4. l'anonymat qui permet aux pairs d'agir librement sans révéler leur identité.
5. la simplicité et la rapidité de l'installation.

3.3 L'évolution des réseaux P2P

De nombreux systèmes pair à pair ont récemment été proposés [CS+01, ABC+02, KBC+00, MMGC02, FFD+01, YCMM02], mais peu de prototypes ont été développés à ce jour, et les mesures de performance manquent pour valider et comparer les différentes architectures proposées.

Les réseaux P2P sont actuellement considérés comme les ancêtres de l'Internet moderne [Sun01]. A l'origine, USENET [Spa99] a été développé afin que les utilisateurs se connectent à un serveur central pour lire et envoyer des messages. Ce système était similaire à celui de Napster qui est apparu après presque vingt ans plus tard. Mais le problème avec USENET était lié à la présence d'un seul serveur (e.g., pas de service quand il tombe en panne). FidoNet [Bus93] est apparu pour résoudre ce problème. Il est basé sur USNET qui a été créé dans un environnement décentralisé imitant la même transition qui a eu lieu 16 ans plus tard lorsque Gnutella a été développé en se basant sur Napster en utilisant

la technologie Peer-to-Peer centralisée. Dès la première semaine, 15 000 personnes ont téléchargé le logiciel, puis 23 millions en Juillet 2000. Au fur et à mesure, USENET et FidoNet ont été améliorés en résolvant quelques problèmes de sécurité et d'extensibilité. Mais malheureusement, les développeurs n'ont pas profité des travaux de leurs prédécesseurs. Par conséquent, durant les deux dernières années, on trouve quelques réseaux et d'applications P2P qui sont apparus et qui ont les mêmes problèmes fondamentaux (e.g., la sécurité). Parmi les systèmes les plus récents citons Freenet [CS+01] , Napster et Gnutella. Le principe des réseaux recouvrants a été employé dans Freenet dont le but est de garantir l'anonymat et de fournir un service se rapprochant d'un Internet anonyme et résistant à la censure. Mais son développement a été stoppé à cause du succès des applications de partage des fichiers.

L'application la plus populaire était Napster qui a été développé à l'origine pour partager des morceaux de musique. Plus tard, son développement a mené à la création de Gnutella qui était destiné pour tout type de fichiers. Ces deux systèmes sont devenus rapidement les applications les plus populaires jusqu'en 2001 où ils ont été bloqués à cause de problèmes juridiques (i.e., droits d'auteurs). Et depuis, Napster est réapparu pour être utilisé comme un service d'abonnement de la musique mais Gnutella a été remplacé par d'autres applications de partage de fichiers.

Les logiciels autrefois attaqués en justice se reconvertissent dans la vente légale de musique sur Internet (Napster, KaZaA). Mais les internautes sont parvenus à pirater ces versions afin de pouvoir continuer à télécharger, sur des réseaux à priori plus légitimes.

Les systèmes pair à pair ont évolué depuis le simple système de partage de fichiers distribués comme Napster et Gnutella jusqu'aux systèmes évolués de gestion de données distribuées comme Edutella [NWQ+02] ou Piazza [HIMT03], qui offrent un niveau sémantique de description des données permettant l'expression de requêtes complexes. Dans ces systèmes, la complexité du problème de la réponse à une requête est directement liée à l'expressivité du formalisme utilisé pour mettre en relation les schémas des différents pairs via des correspondances sémantique (« mappings ») [AGDI03].

En effet, les réseaux Pair-à-Pair sont constitués de nombreux noeuds, munis de mécanismes de réplication qui permettent une haute disponibilité et, ces noeuds sont des PCs standards. Ces systèmes sont d'abord connus pour leurs applications de partage de fichiers (Gnutella [Sol00], Kazaa [LKR04], eDonkey [HBMS04]), mais ils permettent également de mettre en oeuvre des systèmes de fichiers distribués (Past [DR01]), des applications de type publish/subscribe (Meghdoot [GSAA04]), de la multidiffusion (SplitStream [CDK+03]) ou encore de la voix sur IP (P2PSIP [BLJ05]).

Donc après Gnutella, et dès 2001 les utilisateurs méfiants se replièrent sur KaZaA de Sharman Networks [LKR04], qui est basé sur une architecture Peer-à-Peer décentralisée où les internautes sont reliés directement entre eux avec la possibilité de reprendre un téléchargement interrompu, et le fait de pouvoir télécharger depuis plusieurs sources le même fichier afin d'augmenter la vitesse de téléchargement permet de pallier les inconvénients de la disponibilité temporelle.

Et puis en 2003 eDonkey2000 surpasse Kazaa. Grâce à la technique du fractionnement des fichiers (à peine un téléchargement commencé, la partie récupérée est déjà disponible

à l'envoi), les utilisateurs passent de consommateurs à acteurs afin d'alimenter le réseau. C'est donc avec de meilleurs moyens, un plus grand choix de média et une rapidité de téléchargement accrue que les internautes échangent des fichiers protégés par droit d'auteur.

Ensuite, une nouvelle génération de logiciels technologiquement plus avancée apparaît, représentée par BitTorrent [Coh03], Overnet et Grabit, qui optimisent au maximum la bande passante en envoi et réception. Ils ne misent pas sur la disponibilité temporelle mais sur un débit maximal en flux continu.

Enfin, depuis quelques années, nous avons connu le « Peer-to-Peer » crypté (Freenet, GNUnet). Ces logiciels sont basés sur des systèmes de chiffrement variés et garantissent à leurs utilisateurs une confidentialité parfaite dans leurs échanges. Le chiffrement se fait sur un système de clé publique et privée. À noter cependant que l'open-source dans ce cas est une nécessité afin que les programmeurs puissent vérifier l'efficacité du chiffrement.

3.4 Applications modernes des réseaux P2P

Les réseaux pair-à-pair [Sch01, SKL+02] sont largement développés sur Internet. Mais en général nous pouvons définir deux grandes familles de réseaux pair-à-pair : les réseaux non-structurés (e.g., Gnutella [RFI02]) et les réseaux structurés (e.g., Pastry [RD01], CAN [RFH+01], Chord [SMK+01], Tapestry [ZKJ01]). Les réseaux non-structurés sont plutôt adaptés à des recherches complexes (portant sur le contenu du fichier) ou à des applications de diffusion, alors que les réseaux structurés sont limités à des recherches simples. Par contre, les réseaux structurés sont beaucoup plus performants en terme de coût de communication [CCR04].

Au delà du partage de fichiers, les réseaux P2P permettent de mettre en oeuvre des systèmes de fichiers distribués (Past), des applications de type publish/subscribe (Meghdoot [GSAA04]), de la multidiffusion au niveau applicatif (SplitStream [CDK+03]) ou encore de la voix sur IP (P2PSIP [BLJ05]).

3.4.1 Gnutella

Le réseau Gnutella est un réseau composé des noeuds qui utilisent le protocole Gnutella pour communiquer entre eux. C'est l'un des premiers réseaux P2P qui a fait la distinction entre l'échange de musique et l'échange de fichiers. Il se base, dans sa version originale, sur une inondation totale (une transmission à tous les voisins). Mais ce qui rend ce genre de réseau intéressant parce qu'il est l'un des tout premier système qui est entièrement décentralisé, aucun noeud n'ayant un rôle plus important que les autres. Il n'existe aucun point central et par conséquent aucune autorité ponctuelle. Donc, plus de serveurs au sens commun, tout le monde étant à la fois client et serveur. Toutefois, une distinction est faite en fonction de la bande passante supportée par chaque pair ; ainsi les serveurs offrant les meilleures bandes passantes sont mis en avant et sont très sollicités. Lorsqu'un utilisateur sur un noeud donné cherche un objet, Gnutella envoie un message Requête et il spécifie par exemple le nom du fichier recherché à tous ses voisins dans le graphe et cela afin de connaître les noeuds actifs dans le réseau qui seront les seuls à répondre à ce message. Donc, si un des voisins possède l'objet en question, il répond au noeud en

envoyant un message Réponse à une requête qui spécifie où l'objet peut être téléchargé (adresse IP, numéro de port TCP). Par contre, si le noeud qui a reçu la requête ne peut pas la satisfaire, il va renvoyer la requête à tous ses voisins (sauf à l'initiateur du mouvement). Autrement dit, Gnutella inonde le réseau afin de trouver l'objet désiré. D'un autre côté et pour de ne pas noyer le réseau indéfiniment, Gnutella utilise un « Time To Live » (TTL) de la même manière que IP (ce qui doit assurer une inondation totale avec une grande probabilité, mais sans garantie). Le message Requête contient un identificateur (QID pour « Query Identifier ») qui ne contient pas l'identité de la source originelle du message et chaque noeud tient un historique de toutes les requêtes qu'il a reçues récemment. Donc, si un noeud reçoit plusieurs fois un message contenant le même QID, il stoppera sa propagation. Chaque fois qu'un noeud reçoit une réponse à une requête d'un voisin descendant (dans l'arbre), il sait vers quel voisin en amont il doit le rediriger. La figure 3.2 montre le système de recherche de ressources de Gnutella dans sa version initiale.

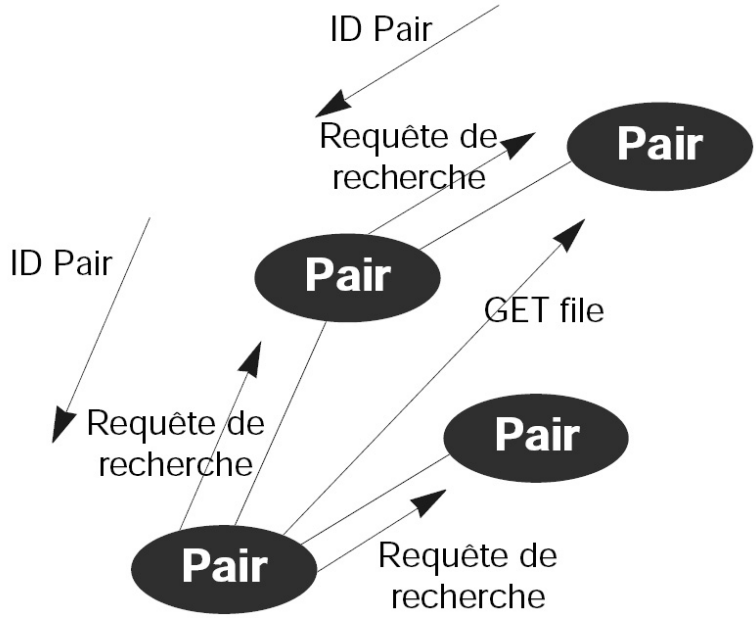

FIG. 3.2 – Le système de recherche de ressources de Gnutella.

L'évolution du graphe de Gnutella se fait de la manière suivante : un noeud qui veut joindre le réseau doit connaître au moins un autre noeud déjà connecté auquel le nouveau noeud sera attaché. Ensuite, il apprendra à connaître les autres noeuds du réseau en recevant des requêtes ou des réponses de requêtes de leur part. Une fois la connaissance effectuée, il peut faire de lui un voisin direct en lui envoyant un message PING auquel il devra répondre par un PONG s'il accepte son nouveau voisin. En d'autres termes, dans le réseau Gnutella, les pairs ne connaissent que les pairs auxquels ils sont connectés. Régulièrement chaque noeud recherche de nouveaux voisins pour maintenir la quantité de noeuds auxquels il est connecté. En fait, Gnutella n'est pas un protocole spécialement intelligent et il est limité par les débits des noeuds. L'inondation assure de trouver un objet en un minimum de sauts mais il ne s'étend pas bien. Il est donc possible de diffuser des requêtes plus ou moins aléatoirement en tenant compte de la probabilité de succès basée

sur les résultats passés. Une autre possibilité est de dupliquer les objets (surtout les plus recherchés) et de cette manière, il est plus facile d'en trouver une copie rapidement.

Les avantages de Gnutella sont :

- La taille du réseau est théoriquement infinie, contrairement aux autres architectures où le nombre de clients dépend du nombre et de la puissance des serveurs
- L'utilisation du réseau est anonyme
- Le réseau est très tolérant aux fautes
- Adaptation dynamique au réseau.
- Les connexions peuvent se faire entre des plateformes distinctes (Windows, Linux, Mac) grâce à la multitude de clients développés pour ces plateformes.

Les inconvénients de Gnutella sont :

- Gros consommateur de bande passante et un nombre très important d'informations circule sur ce type de réseau
- Pas de garantie de succès, ni d'estimation de la durée des requêtes
- Pas de sécurité, ni de réputation (pas de notion de qualité des pairs, ni de données fournies).

3.4.2 Chord & Pastry

3.4.2.1 Chord

Chord [SMK+01] est un protocole de nouvelle génération totalement distribué et fortement extensible. Il dispose uniquement de deux fonctionnalités simples : la création et la recherche des clés pour publier ou récupérer des données. Il fonctionne comme une sous-couche au niveau application.

Chord est un protocole simple et de qualité car il dispose de toutes les caractéristiques nécessaires à la gestion d'un réseau pair-à-pair où chaque noeud du réseau contribue équitablement en utilisant une fonction de hachage consistante qui évite les problèmes de surcharge. De plus, Chord est un protocole totalement décentralisé où chaque noeud fonctionne d'une manière autonome ce qui rend le réseau extrêmement robuste aux pannes. D'un autre côté, Chord est un protocole extensible et très réactif grâce à son algorithme performant de maintenance des tables de routages qui offre une forte disponibilité.

Chord repose sur une structure en anneau où chaque noeud a la connaissance de son prédécesseur et de son suivant mais cette connaissance n'est pas suffisante pour garantir une bonne performance de l'anneau, notamment en termes de nombre de sauts par requête. Ce système fournit le routage de messages uniquement vers le noeud qui possède la ressource. Dans Chord, chaque ressource est assignée à un noeud de la manière suivante : une ressource d'identifiant id (générée à partir de son adresse IP en utilisant une fonction de hachage régulière) est assignée au noeud possédant le premier identifiant supérieur à id (figure 3.3). Donc, chaque pair est placé dans l'anneau de manière à ordon-

ner les identifiants par ordre croissant. Etant donnée la nature volatile des réseaux P2P, un noeud entrant peut donc prendre le contrôle d'une ressource existante.

Afin d'effectuer le routage des messages, chaque noeud connaît seulement son successeur et les noeuds forment ainsi une liste chaînée. Donc, les messages sont routés le long de l'anneau tant que l'identifiant du noeud actuel est inférieur à l'identifiant recherché. Puis, une requête sera envoyée au noeud pour une ressource d'identifiant inférieur car il sait qu'il est responsable de cette ressource.

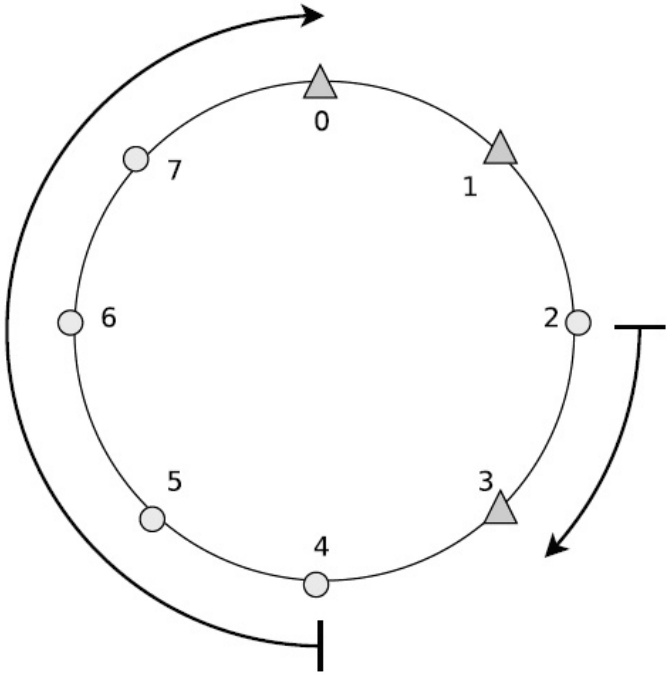

FIG. 3.3 – Assignation des ressources aux noeuds dans Chord. Ici, la ressource 2 est assignée au noeud 3 et les ressources 4 à 7 au noeud 0.

Dans Chord, chaque noeud a un pointeur vers son prédécesseur afin de simplifier l'insertion et le départ des noeuds. Pour simplifier ces tâches il y a trois étapes à effectuer qui sont : l'initialisation de la table du nouveau noeud, la mise à jour des tables des noeuds qui doivent contenir le nouveau noeud et enfin, la signalisation d'insertion à la couche logicielle supérieure du transfert des ressources dont elle est responsable.

Ce système garantit le maintien de la cohérence de l'anneau et des tables de routage des pairs concernés par l'arrivée du nouveau noeud en exécutant régulièrement un processus de maintenance. Dans ce processus, chaque pair vérifie la validité des informations détenues dans sa table de routage à savoir : son suivant et les *fingers* (les noeuds séparés de 2^i, où i varie de 0 à m, m est le nombre d'entrées dans la table de routage, dans le sens inverse des aiguilles d'une montre, d'où l'utilisation des prédécesseurs). Afin de limiter les échecs de requêtes survenants dans l'intervalle du temps situé entre la disparition d'un noeud et l'exécution de ce processus, chaque noeud maintient une liste de suivants qui peuvent être utilisés en cas de défaillance.

Chord est capable de supporter un réseau à large échelle et peut assurer l'exécution de toutes les requêtes. Son interface simple peut fournir deux fonctions (recherche et obtention de clés) permettant une implémentation simple avec des systèmes déjà existants comme par exemple un système de fichier réseau. Il peut être implémenté pour gérer des index centralisés comme dans Napster ce qui augmente la robustesse du système.

En fait, Chord est un protocole de recherche P2P qui est déjà déployé dans plusieurs applications : tout d'abord, dans CFS [FFD+01] qui est un système de fichiers distribué à l'échelle de l'Internet, ensuite dans ConChord [SECHS02] qui utilise CFS afin de fournir une infrastructure distribuée pour la délivrance de certificats de sécurité SDSI (Simple Distributed Security Infrastructure) et enfin dans DDNS [CMM02] qui propose une implantation P2P du DNS.

Enfin, il est important à retenir que Chord ne fournit qu'un service de routage et laisse notamment le transfert et la réplication des ressources à la charge de la couche supérieure, car c'est simplement un protocole de répartition de clés de hachage au sein d'un réseau en anneau.

3.4.2.2 Pastry

Pastry [AP01, ZKJ01, RD01], comme Chord, est un réseau auto-organisant basé sur une structure en anneau comme le montre la figure 3.4. Il utilise un protocole de routage et de localisation P2P où chaque noeud membre du réseau possède un identificateur unique de 128 bits choisi aléatoirement et représentant sa localisation dans l'anneau. Il est le résultat d'une fonction de hachage appliquée sur l'adresse IP ou sur la clé publique du noeud, alors que les identifiants des ressources sont codés sur 160 bits. Ces identificateurs sont virtuellement disposés sur un cercle et l'espace d'adressage ainsi constitué est considéré comme circulaire. Cette architecture repose sur un modèle P2P décentralisé et utilise une approche hybride pour le routage. Il suggère des contraintes pour un système de transfert et de réplication efficace tout en assurant la pérennité des données dans un environnement coopératif.

Pour router un message dans le réseau, les applications qui utilisent Pastry lui associent une clé dont l'espace des valeurs est le même que celui des identifiants du noeud. Pastry utilise un algorithme de routage dérivé de celui de Plaxton [PRR97] qui consiste à propager les messages vers le noeud qui partage un plus grand préfixe commun avec la ressource, ou à défaut il sera routé vers le noeud partageant un préfixe de même taille et plus proche numériquement de la ressource ciblée. Ainsi, pour un réseau comprenant N noeuds, le routage nécessite au plus log2bN sauts où 2b est la base de numérisation retenue pour les identifiants.

Le routage dans Pastry est effectué de la manière suivante : quand un message, dont la clé est K, arrive dans un pair n, ce dernier vérifie si cette clé correspond à un pair d'identifiant voisin virtuellement. Si c'est le cas, le message sera routé directement vers le pair correspondant, sinon, le pair n détermine m qui est le nombre des chiffres communs entre son identifiant et K. Ensuite, il consulte sa table de routage afin de trouver un pair qui a m+1 chiffres en commun et lui fait suivre le message. Par contre, si aucun pair ne correspond, le pair n va chercher un autre pair ayant m chiffres en commun et dont l'identifiant

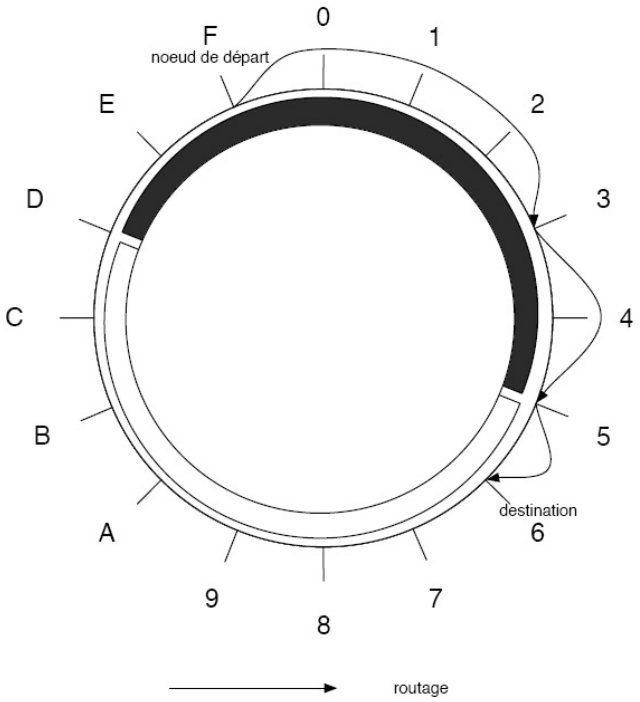

FIG. 3.4 – Disposition des noeuds de Pastry dans l'espace circulaire.

est le plus proche de K. Enfin, le message arrive à destination lorsqu'il n'y a aucune entrée dans la table de routage qui permet de se rapprocher plus près de la clé K.

La table de routage est mise à jour de manière passive tout en complétant les trous des noeuds défaillants et en remplaçant les entrées par les noeuds les plus proches. En autres terme, on peut avoir le cas où une des valeurs de la table de routage soit vide (à cause du départ ou de la défaillance d'un noeud), et comme chaque noeud connaît ses plus proches voisins, le message sera transmit au noeud qui possède l'identifiant le plus proche dans sa liste de voisins.

Donc, nous pouvons remarquer que le routage dans Pastry est naturellement résistant à la défaillance des noeuds et la progression d'un message vers la destination est assurée. Mais il paraît également qu'une trop grande liberté des entrées dans les tables de routage pose parfois des problèmes de sécurité, où un adversaire peut occuper une partie considérable de la table de routage même si les identifiants des noeuds sont sécurisés.

Outre les avantages de son algorithme de routage dont l'efficacité, la tolérance aux fautes et le passage à l'échelle, la caractéristique distinctive de Pastry est qu'il permet la prise en compte d'un critère de distance entre les noeuds dans l'espace physique. Cette localité géographique est prise en compte lors de la mise à jour de la table de routage. D'un autre côté, lors de la construction de cette table de routage, Pastry renseigne systématiquement les entrées de la table avec les noeuds les plus proches du noeud local, grâce à un mécanisme de localité, le routage d'un message au travers de Pastry n'est que 30% à 40% plus long que le plus court chemin existant.

Actuellement, Pastry est devenu un standard parmi les systèmes P2P structurés. Il est utilisé dans deux applications P2P : PAST [DR01] qui est une application distribuée de stockage de fichiers et Scribe [RKCD01] qui est une application P2P de diffusion d'événements.

3.4.3 CAN

Le réseau CAN (« Content-Addressable Networks ») [RFH+01, CDG+02b, RFH+01] est une infrastructure distribuée qui peut fournir des services de routage, de réplication, de répartition de charge et de hachage à grande échelle (i.e., Internet) en utilisant un routage par progression dans un espace cartésien multidimensionnel. Dans ce réseau, chaque noeud possède une localisation dans un espace cartésien multidimensionnel qui peut être également l'espace de nommage des clés. Donc, CAN définit un espace virtuel de d dimensions qui n'a aucun rapport avec l'espace physique. Cet espace est partitionné et partagé entre tous les pairs dans le système où chaque pairs est responsable d'un sous-ensemble unique dans cet espace. Autrement dit, un noeud (hôte par exemple) connecté à une application peer-to-peer sera donc responsable d'une région de l'espace.

Concernant le routage dans CAN, chaque noeud possède une table de routage qui contient des informations sur chacun de ses voisins immédiats dans l'espace virtuel, ainsi que sur les zones qu'ils couvrent. Donc, quand un pair veut router un message, il l'envoie vers le pair qui peut le rapprocher de la destination. Pratiquement, on peut trouver plusieurs noeuds qui peuvent remplir cette tâche (plusieurs routages possibles). Pour optimiser le routage, le message peut être envoyé vers un noeud proche géographiquement (qui a une latence faible)et ce mécanisme permet à CAN d'être naturellement résistant aux défaillances. La figure 3.5 montre un exemple du routage dans CAN où le noeud N0 essaie de contacter la clé C. Il transmet le message vers le noeud responsable de l'espace le plus proche de la destination. Ainsi le message sera routé de cette manière à travers le réseau jusqu'au noeud N1 qui est responsable de l'espace contenant l'identifiant (xc,yc) de la clé C. Dans ce réseau, l'augmentation du nombre de pairs n'influence pas sur le nombre d'informations concernant les voisins d'un pair, par contre, la longueur moyenne d'un chemin évolue en termes de $O(n^{1/d})$. L'insertion d'un noeud ne nécessite donc de communiquer qu'avec un nombre de noeuds très faible.

Dans CAN, quand un noeud quitte le réseau ou subit une défaillance, sa zone doit être contrôlée par un de ses voisins après un certain délai (proportionnellement à l'espace qu'il gère déjà). Le noeud qui prend le contrôle possède initialement un petit espace, il avertit les voisins et met à jour sa table de voisinage.

Pour optimiser le routage dans CAN et le rendre plus rapide, un certain nombre d'optimisations peuvent être effectuées afin d'exploiter la localisation physique sans provoquer de longs routages tout en assurant la disponibilité des ressources. Un premier mécanisme consiste à ajouter des dimensions ce qui permet d'avoir un plus grand choix de voisins à chaque saut. Une autre amélioration peut être obtenue par le mécanisme de la réplication et de la surcharge des zones où plusieurs noeuds peuvent posséder la même zone. Dans ce cas, chaque noeud choisit le voisin le plus proche de lui. Afin de limiter la charge au niveau des ressources très demandées, un noeud maintient un cache des données pour éviter

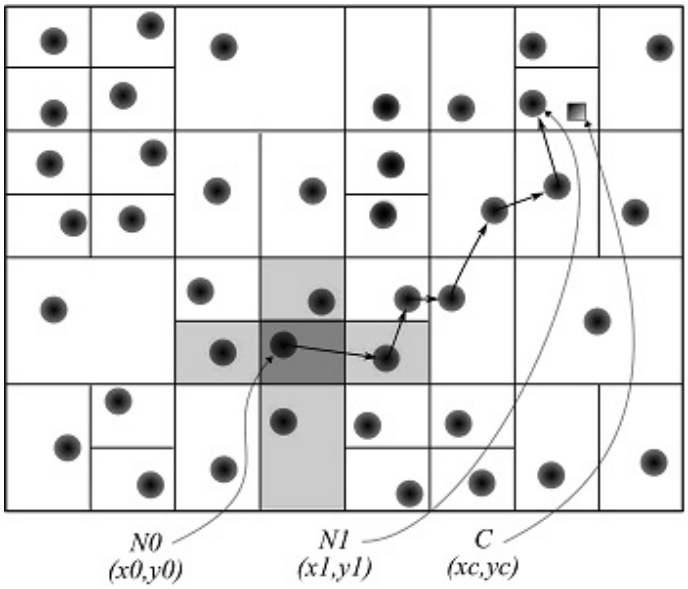

FIG. 3.5 – CAN : exemple de routage sur l'espace de nommage bidimensionnel.

de multiples requêtes. Donc, si un noeud est surchargé, il peut répliquer les ressources populaires sur ses voisins qui pourront ainsi soit répondre aux requêtes, soit les transmettre au noeud responsable.

Comme nous avons vu précédemment, CAN présente les mêmes vulnérabilités que Chord et Pastry face à des noeuds qui peuvent choisir leur identifiant et ainsi corrompre le routage et les ressources.

3.5 Réseaux P2P *vs* Réseaux recouvrants

Malgré le fait que les termes de réseaux P2P et de réseaux recouvrants soient actuellement deux concepts différents, il arrive parfois qu'on utilise les deux termes pour parler du même système car ces deux réseaux sont généralement très liés. Un réseau recouvrant, comme nous avons vu dans le premier chapitre, est un réseau virtuel avec une architecture un peu spécifique ; il décrit comment les noeuds sont localisés dans le réseau et comment ils communiquent entre eux. Ici, l'adjectif *recouvrant* signifie que ce réseau se situe sur un autre réseau déjà existant en exploitant ses ressources. Par contre, les communications P2P sont utilisées pour décrire le type de transmission employé par les réseaux recouvrants en utilisant des applications P2P (programmes de partage de fichiers : Napster, Gnutella, etc.). La caractéristique de ces applications comme P2P est qu'elles utilisent les réseaux recouvrants pour effectuer une recherche et un transfert des fichiers efficaces. Donc, les réseaux P2P offrent la logique et les informations tandis que les réseaux recouvrants s'occupent du routage des données de la source jusqu'à la destination. En d'autres termes, on peut dire que les pairs participants à un service P2P forment souvent un réseau recouvrant [DO03].

D'ailleurs, un réseau P2P ne constitue par obligatoirement un réseau recouvrant. Il se trouve que des applications reposant sur des protocoles comme NNTP9 [BL86] ou

RIP [Mal98] soient construites selon le modèle P2P dans le sens où il n'y a aucune centralisation et que chaque noeud agit comme un client et/ou un serveur sans pour autant constituer un réseau virtuel au-dessus du réseau IP.

De toute façon, comme les deux termes sont souvent très associés, on peut utiliser l'un des deux surtout quand on travaille sur un système qui les emploie ensemble.

3.6 Exemples de réseaux recouvrants

Comme nous avons vu précédemment, l'idée des réseaux recouvrants n'est pas récente. L'Internet a commencé comme un réseau de données recouvrant au-dessus du réseau téléphonique public. Avec la popularité énorme d'Internet, il est devenu un sujet essentiel et très important pour la recherche dans le domaine des réseaux. Par la suite, nous allons parler brièvement de plusieurs exemples de réseaux recouvrants dont MBone qui offre un service multipoint.

3.6.1 MBone

Le « Multicast Backbone » (Mbone) [Han94] est le plus grand réseau recouvrant déployé sur Internet. C'est un réseau virtuel superposé à l'infrastructure physique d'Internet qui utilise des routeurs sachant reconnaître l'IP multicast. Il permet d'acheminer les données depuis la source vers plusieurs destinataires tout en optimisant la bande passante. Donc Mbone est un réseau maillé de routeurs multicast interconnectés en utilisant les liens Internet comme des tunnels afin d'assurer le passage des données en multidiffusion à travers des routeurs qui ne sont pas configurés pour cela. Dans ce réseau, un routeur multicast transmet les paquets multicast au routeur multicast voisin qui se trouve dans une session multicast. Un processus d'élagage est effectué afin d'éviter l'irrigation des branches qui n'auraient pas d'abonné. Un seul paquet peut transiter entre deux routeurs multicast, peu importe le nombre d'émetteurs et de participants. Le routeur qui a des abonnés sur son réseau local sera chargé de la diffusion des paquets. Les paquets qui circulent sont des paquets multicast enveloppés dans une enveloppe IP normale (unicast). Ce mécanisme permet de traverser les routeurs qui ne reconnaissent pas les paquets multicast. MBone met en oeuvre le protocole Internet de multidiffusion et est utilisé pour la vidéoconférence, l'audioconférence, la diffusion de vidéo,la transmission de documents (html ou autres). Actuellement, il utilise deux types de protocoles : DVMRP (« Distance Vector Multicast Routing Protocol ») qui utilise la notion de distance et s'apparente à RIP (« Routing Information Protocol ») et PIM (« Protocol Independent Multicast ») qui est bien adapté aux réseaux multicast étendus.

3.6.2 6Bone

6Bone est un réseau mondial virtuel et expérimental, construit au-dessus de l'Internet IPv4. C'est un projet en collaboration démarré par l'IETF (« Internet Engineering Task Force ») entre l'Amérique du Nord, l'Europe (dont la France) et le Japon et mis en place en 1996. Il est destiné à tester le protocole IPv6 et pour assister le développement graduel

de l'Internet [Hin96]. Il est constitué de nuages de machines connectées en IPv6 et reliées par des liens point à point appelés tunnels et qui supportent directement IPv6. 6Bone est directement connecté à Internet, donc il constitue un sous réseau dont l'accès est totalement ouvert permettant à tout le monde d'y accéder à condition d'être équipé d'un ordinateur correctement configuré (supportant les piles IPv4 et IPv6). Comme 6bone est un réseau de test, pas de production, il est prévu après un certain temps que le 6Bone disparaisse, puisqu'il a été créé pour tester la transmission des trames IPv6 en mode native. De tout façon, il sera remplacé d'une façon transparente par les ISP (« Internet Service Providers ») et autres réseaux publics Internet [IGG00].

3.6.3 X-Bone

X-Bone [TH98] est l'infrastructure d'un projet destiné à accélérer le déploiement des réseaux recouvrants basiques comme MBone [Kum96]. C'est un système distribué déployé au niveau de la couche application permettant aux noeuds de participer sans faire aucune modification au niveau de la couche réseau. Il fournit une interface graphique pour automatiser les configurations des adresses IP avec les noms dans les serveurs DNS, les configurations du routage des réseaux recouvrants simples et il permet aussi de maintenir à distance les réseaux recouvrants dans des sessions http cryptées par l'algorithme SSL. De plus, ses fonctions permettent de gérer les adresses et les DNSs et de fournir des mécanismes pour manipuler l'insertion des paquets dans les réseaux recouvrants.

X-Bone gère d'une manière dynamique les réseaux recouvrants sur Internet afin de réduire le coût de leur configuration et d'augmenter le nombre d'éléments partagés dans le réseau. Autrement dit, X-Bone découvre, reconfigure et surveille les ressources du réseau afin de créer un réseau recouvrant au-dessus des réseaux IP existants. Il emploie le mécanisme multipoint pour simplifier la découverte des ressources et pour garantir un déploiement sécurisé sur un réseau recouvrant sécurisé.

Ce système permet aux différentes applications qui tournent sur la même machine (utilisateur ou routeur) d'être liées à plusieurs réseaux recouvrants différents. Il n'exige aucun système d'exploitation spécifié ni aucune modification mais il se base uniquement sur le système IP en utilisant les implémentations existantes comme le routage dynamique et le service du nommage. De plus, il utilise un mécanisme de tunnel composé de deux couches plutôt qu'une seule couche (qui est utilisé dans les réseaux recouvrants traditionnels). Et grâce à ce mécanisme, il peut utiliser les applications non modifiables et les services déjà employés dans le réseau recouvrant. Cela permet aussi aux ressources du réseau (hôte, routeur) de participer plusieurs fois dans le même réseau recouvrant (le réseau recouvrant qui supporte IPSec et le routage dynamique [TEW04]).

3.6.4 Yoid/Yallcast

Yoid [8] représente une architecture générale de distribution du contenu qui essaie d'unifier les avantages du multipoint d'IP native (l'efficacité des connexions entre des groupes locaux) avec les avantages du déploiement et de l'implémentation des systèmes terminaux multipoint [Fra00]. Il essaie de connecter les petits îlots et les serveurs multipoints afin d'obtenir une architecture rudimentaire multipoint globale. Il contient une

plateforme complète pour l'implémentation des réseaux recouvrants multipoint, mais le but principal est d'étudier tous les aspects des transmissions P2P comme la connectivité, le contrôle du flux, la fiabilité, etc. La base de Yoid est le protocole YTMP (« Yoid Tree Management Protocol ») [Fra99] qui permet à un groupe d'hôtes d'être auto-configuré d'une manière dynamique dans deux topologies : arbre et maille qui sont deux topologies créées indépendamment. En autres mots, ces protocoles génèrent une topologie d'arbre pour une distribution efficace du contenu et une topologie Mesh pour une distribution robuste du contenu. Yoid utilise les protocoles TCP ou RTP entre les noeuds afin de garantir la gestion de la congestion dans les liens du réseau recouvrant. Enfin, ce système est destiné à fournir une meilleure architecture multipoint.

Yallcast est, comme Yoid, utilisé pour prolonger l'architecture multipoint d'Internet en utilisant un groupe de protocoles basé sur la distribution du contenu plutôt que sur des connexions unicast. Il utilise un hôte principal (appelé : rendez-vous) qui a le même rôle du contrôleur de groupe dans ALMI, sauf qu'il est utilisé ici pour informer les nouveaux membres de la présence des membres actuels dans l'arbre. YAllcast crée un arbre multipoint distribué tout en utilisant un protocole de routage distribué. Il maintient aussi une topologie "Mesh" parmi les membres du groupe et cela afin de garantir que le groupe multipoint n'est pas partitionné. Yoid, comme son prédécesseur Yallcast, est en cours de développement.

3.6.5 ALMI

ALMI (« application Level Multicast Infrastructure ») [PSVW01] est un intergiciel de communication multipoint « many-to-many » au niveau applicatif destiné aux petits groupes (dizaines des noeuds). Il permet d'accélérer le déploiement des applications et d'avoir une configuration simple du réseau et cela sans avoir besoin du supporte de l'infrastructure du réseau. Contrairement au système terminal multipoint et au Yoid, ALMI utilise une méthode centralisée pour gérer ses arbres de distribution multipoint, ce qui diminue la complexité des algorithmes qui créent l'arbre tout en garantissant la consistance et l'efficacité de l'arbre. Par contre, cela nécessite un contrôleur central afin de garantir la fiabilité, la flexibilité et la tolérance aux pannes (mouvement ou panne des noeuds). En général, une session ALMI est composée d'un contrôleur et des membres qui sont des petits programmes. Le contrôleur doit se situer dans une place où il sera accessible par tous les membres tout en cohabitant avec l'un de ces membres. Il traite l'enregistrement des membres et il maintient aussi l'arbre multipoint afin d'assurer la connectivité dans l'arbre en cas de mouvement ou de panne des noeuds. Les membres participants dans une session ALMI sont connectés via un arbre multipoint virtuel qui se base sur des connexions unicast entre les terminaux tout en utilisant un format commun des paquets pour transférer tout genre de paquet (data, contrôle).

Enfin, pour évaluer ALMI, l'algorithme qui génère l'arbre (comme RON) utilise tous les noeuds dans la structure maillée pendant la construction de son arbre de distribution.

3.6.6 Système terminal multipoint

Le système terminal multipoint (ESM) "End system multicast" [YHGSH02]offre des services basés sur l'application multipoint en forme d'un réseau recouvrant au-dessus des connexions unicast sur Internet . Les systèmes qui participent dans un groupe multipoint communiquent via une structure recouvrante dans le sens où chacun de ses terminaux correspond à une connexion unicast entre deux systèmes existants sur Internet. Il est conçu en fonction de connexions multipoint de type « many-to-many » pour des groupes de taille moyenne (dizaines jusqu'à une centaines). Cette technologie permet à n'importe quelle personne connectée sur Internet d'envoyer des flux audio et vidéo de haute qualité en utilisant un réseau P2P pour distribuer ces flux, ce qui veut dire que les personnes qui regardent une vidéo peuvent l'échanger et l'envoyer aux autres en même temps. Par conséquent, la bande passante sera partagée entre eux d'où la nécessité d'une grande bande passante.

Dans ESM, chaque hôte participant au réseau, implémente ses propres arbres de multicast. Les systèmes terminaux s'organisent en forme d'arbre en utilisant un protocole distribué. Ils montrent leur performance aux autres noeuds (bande passante et latence) tout en maintenant leurs positions dans l'arbre afin de les améliorer quand cela est possible. De plus, dans l'arbre, les protocoles d'optimisation maintiennent des hôtes partout dans le monde afin d'améliorer la performance et d'avoir une distribution du contenu plus efficace. Ces systèmes apportent plus de flexibilité ce qui permet d'ajouter plus de fonctionnalités pour des communications multimédia. Toutes les fonctionnalités sont implémentées dans les pairs au lieu de les avoir dans les routeurs. Ainsi, la multi-diffusion des paquets sera au niveau applicatif. Cette approche permet aussi d'éliminer les messages redondants. Contrairement aux VPNs et MBone, ESM n'utilise que les hôtes Internet, les routeurs n'étant pas pris en compte dans le réseau recouvrant. Par ailleurs, ces hôtes s'échangent des messages en utilisant des tunnels UDP plutôt que des tunnels IP, ce qui rend ce genre de systèmes plus facile à implémenter.

3.6.7 Overcast

Overcast est un réseau recouvrant multipoint au niveau application, qui offre un service multipoint depuis une seule source vers plusieurs clients (architecture centralisée) [JKL+00]. Il représente une architecture multipoint fiable et extensible qui utilise un protocole simple afin de construire des arbres de distribution de données d'une manière efficace.

En comparant Overcast avec l'IP multipoint, on trouve que Overcast offre une distribution efficace de la bande passante (de 70% à 100%). De plus, il s'adapte rapidement à tout changement survenu à cause de l'arrivée des nouveaux noeuds ou de pannes des noeuds existants sans augmenter la charge sur la source, ce qui rend ce système plus avantageux. Et afin de garantir la tolérance aux pannes, la source centrale peut être dupliquée et cela permet à Overcast d'avoir des mécanismes de maintenance plus fiables.

Le seul défi auquel Overcast doit faire face est la conception et l'implémentation des protocoles qui peuvent construire des arbres de distribution adaptatifs et efficaces sans avoir besoin de connaître les détails de la topologie du réseau sous-jacent.

3.6.8 Réseaux de diffusion du contenu

Les réseaux de diffusion de contenu (CDNs) comme Akamai [SAR99] et Inktomi [Cor01] sont une sorte d'infrastructures parallèles au réseau internet. Ils utilisent des techniques des réseaux recouvrants et des fonctions de mise en cache afin d'améliorer la performance de la diffusion du contenu pour des applications spécifiques comme celles qui diffusent les flux multimédia. Le principe d'un CDN est simple et le but est de rendre les informations provenant d'une source précise (contrôle centralisé) plus rapide et accessible en utilisant plusieurs mécanismes dont la duplication multicast (solution globale intégrée d'équilibrage de charge), le contournement intelligent des caches, la gestion de la bande passante et la délocalisation des informations. Ces mécanismes permettent de rapprocher les informations le plus possible des utilisateurs finaux et d'optimiser les flux multimédia tout en améliorant la QoS. Cela peut être accompli en installant des serveurs de réplication à différents endroits dans le réseau. En d'autres termes, la mise en place d'un CDN permet de réduire la consommation de la bande passante, des ressources de serveurs et de la mémoire et donc de mieux servir les requêtes en toute transparence. Afin de rendre le CDN plus fiable, il utilise plusieurs fonctions pour maîtriser son routage (meilleur routage possible) tout en assurant la QoS sur internet de bout en bout. Ces fonctions sont :

1. le cache (« Content Cache ») ou le stockage du contenu sur un serveur où les informations sont soit téléchargées d'avance et mises à jour régulièrement, soit stockées au fur et à mesure des demandes.
2. le re-routage (« Local Director », « Content Director ») où les demandes sont redirigées vers le serveur le plus approprié (pas forcément le plus proche). Cette fonction est la base du concept CDN.
3. la gestion du contenu (« Content Manager », « Diffusion Manager ») qui sert à contrôler la distribution, les flux des informations, l'état des stockages et les utilisateurs.

Ces trois fonctions peuvent être intégrées aussi bien à travers Internet qu'à travers un réseau d'entreprise (ECDN : « Enterprise Content Distribution/Delivery Network »). Un autre avantage des CDNs est la sécurité. Quand un serveur de contenu est surchargé à cause d'une attaque d'un pirate, ce dernier serait automatiquement bloqué par le système de gestion et les utilisateurs seraient re-routés vers un autre site. Ce qui permet de limiter très fortement ce type d'attaque. Enfin, chaque CDN a ses spécificités, ses avantages compétitifs, son réseau de serveurs et de caches, mais tous les CDNs fonctionnent aujourd'hui en mettant en oeuvre la meilleure mixture entre réplication, redirection, routage et contrôle.

3.6.9 Réseaux recouvrants résilients

RON (« Resilient Overlay Network ») [ABKM01] est un réseau recouvrant qui a pour but de trouver des routes alternatives pour les applications unicast traditionnelles. Il se trouve parfois que l'envoi d'un message indirectement via un ou plusieurs intermédiaires est plus rapide que de l'envoyer directement à sa destination. En général, il existe une différence fondamentale entre l'extensibilité et l'optimalité d'un algorithme de routage. Par exemple, un BGP s'étend sur de très grands réseaux comme Internet, mais il ne choisit

pas forcément la meilleure route, il essaie seulement de trouver quelques routes possibles entre deux sites. De plus, il est lent à s'adapter aux pannes du réseau.

Par contre, RON ne s'occupe que d'une douzaine de noeuds seulement, car il utilise une N x N (où N est le nombre des noeuds dasn le réseau) stratégie de surveillance étroite, et cela à l'aide des sondes actives sur trois aspects de qualité des chemins entre chaque paire de sites.

1. Latence.
2. Bande passante disponible.
3. Probabilité de perte.

Le but de RON est de chercher et de sélectionner la meilleure route entre n'importe quel paire dans le réseau. Quand les conditions du réseau changent, il tente de changer rapidement de route. De plus, il est capable de délivrer de modestes améliorations de performance des applications et il récupère les erreurs du réseau beaucoup plus rapidement.

En faisant des tests sur une instance de RON composée de 12 noeuds pendant une période de 64 heures, on a détecté 32 pannes en une trentaine de minutes. De plus, il a corrigé toutes le pannes en moins de 20 secondes. Par conséquent, on peut comprendre que la propagation des données à travers ce réseau avec au moins un seul noeud intermédiaire sera souvent suffisant pour retrouver les erreurs d'Internet.

Donc, comme nous avons vu, le concept général qui illustre tous les réseaux recouvrants cités des réseaux d'ordinateurs est la virtualisation.

Ceci étant, il est possible de construire un réseau virtuel de ressources abstraites (logiques) au-dessus d'un réseau physique construit avec des ressources physiques comme Internet. D'ailleurs, il est possible d'empiler ces réseaux virtuels sur d'autres réseaux virtuels, et également de faire coexister plusieurs d'entres eux au même niveau. Alternativement, cela procure de nouvelles possibilités, d'applications ou de réseaux de niveaux supérieurs.

3.7 Conclusion

Avec l'évolution de la technologie, de plus en plus d'applications apparaissent aujourd'hui dont l'exploitation de la voix sur IP, les grilles de calcul, etc. Dans ce chapitre nous avons exposé les réseaux pair-à-pair où nous avons commencé par définir ces réseaux en étudiant les différentes étapes de leur évolution. Cette évolution a commencé avec de simples réseaux P2P (USENET et FidoNET) dont l'architecture était centralisée. Ensuite, après l'apparition de Napster dont le fonctionnement était défini pour échanger de la musique, d'autre réseaux sont apparus (Gnutella, Chord, etc) afin de commencer une nouvelle ère beaucoup plus avancée.

Nous avons vu que les réseaux pair-à-pair possèdent des capacités d'auto-organisation et de tolérance aux défaillances. De plus, la symétrie entre les noeuds, l'équilibrage de charge et la connaissance uniquement locale du système fournissent aux réseaux pair-à-pair une capacité inhérente de passage à l'échelle. Les applications pour les systèmes pair-

à-pair ne se limitent pas au partage de fichiers. Par exemple, de nombreuses applications ont été développées telles que l'archivage de données, la gestion de cache, la diffusion du contenu, etc.

Après avoir présenté une vue globale et théorique des réseaux recouvrants et de quelques applications P2P, nous allons dans le chapitre suivant présenter notre architecture en expliquant son fonctionnement théorique. Nous détaillerons aussi dans le prochain chapitre les simulations que nous avons effectuées afin d'évaluer notre architecture et ses performances.

Chapitre 4

Architecture autonome et dynamique pour réseaux recouvrants

4.1 Introduction

Actuellement, beaucoup d'applications distribuées déploient des réseaux recouvrants au-dessus d'Internet. Plusieurs problèmes restent non résolus au niveau de la couche réseau, ce qui explique cette tendance à implémenter des services réseaux comme la mobilité, la sécurité et le multipoint dans la couche application. D'une part, les réseaux recouvrants créant des topologies basiques sont habituellement limités au niveau de la flexibilité et de l'extensibilité et d'autre part, les réseaux recouvrants créant des topologies complexes exigent des mécanismes d'adressage, de nommage et de routage au niveau application. Notre but dans ce travail, est d'avoir un intergiciel d'adressage, de routage et de nommage robuste et efficace pour déployer et maintenir ce type de réseaux. La seule hypothèse retenue est que ces réseaux soient déployés au-dessus d'Internet. L'architecture proposée est basée sur la séparation de l'espace d'adressage et celui du nommage en fournissant un plan de convergence pour l'environnement hétérogène actuel d'Internet. Pour implémenter cette propriété, nous proposons un système distribué et extensible pour lier k résilient nom-adresse. Notre travail [SML07] décrit la conception d'un intergiciel et présente sa performance concernant l'extensibilité du routage, l'efficacité du chemin non optimal et sa résilience à la dynamique de réseau.

Théoriquement, concernant le routage local, la relation entre les tailles de la table de routage et les longueurs du chemin a été étudiée par [EGP98]. Ils ont prouvé qu'il est possible de limiter le stretch moyen (i.e., la longueur du chemin non optimal) par 3 et cela avec une taille de table de routage de $O(n^{3/2}\log^{3/2}n)$. De la même façon, [Cow99] a prouvé aussi qu'il est possible de limiter le stretch maximal par 3 avec une taille de tables de routage de $O(n^{2/3}\log^{4/3}n)$, et [KFY04] ont montré que le routage compact d'Internet mène à un stretch moyen de 1.1. Pour atteindre leur but, ils ont utilisé un étiquetage approprié pour chaque noeud mais personne n'a expliqué comment implémenter cette méthode dans un algorithme distribué. En ce qui concerne les topologies des réseaux recouvrants, [LM04] a montré que ces réseaux ont un impact sur les performances du routage ce qui a motivé notre travail. Donc, dans notre étude, nous présentons une infrastructure où les tailles des tables de routage ne sont pas calculées en fonction de la taille du réseau mais plutôt en fonction des degrés du noeud.

Dans ce chapitre nous présentons d'abord l'architecture proposée en détaillant des différentes couches qui existent. Puis, nous montrons les différents avantages de cet intergiciel et enfin, nous consacrons la dernière partie pour décrire et commenter les différentes expérimentations que nous avons effectuées.

4.2 Description de l'architecture

Comme l'architecture que nous proposons n'a aucune contrainte sur la topologie des réseaux recouvrants, nous avons besoin de définir un mécanisme d'adressage distribué afin de router les paquets des données dans ces réseaux. Notre intergiciel est actuellement surnommé DHARMA qui signifie en anglais : Dynamic Hierarchical Addressing, Routing and naMing Architecture.

4.2.1 Plan général

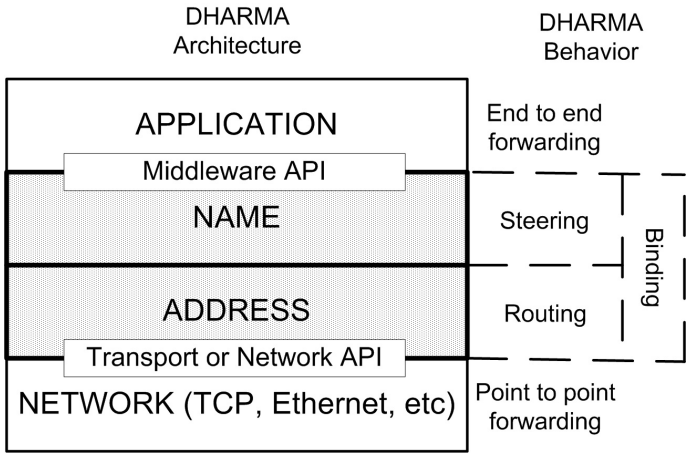

FIG. 4.1 – Architecture générale de DHARMA.

La figure 4.1 présente l'architecture proposée (infrastructure et intergiciel) où on peut voir que la couche d'adressage se situe au-dessus de la couche réseau et que la couche de nommage se situe au-dessus de celle d'adressage. Le processus du transfert des paquets dans le réseau recouvrant est divisé en deux phases. La première, appelée routage, ressemble au routage employé actuellement sur Internet. Le routage dans le réseau recouvrant utilise seulement ses propres adresses. Les connexions entre les noeuds au niveau transport ou au niveau réseau sont considérées comme des connexions point-à-point.

Si on utilise un protocole fiable dans la couche transport (comme le TCP) alors aucun mécanisme n'est nécessaire pour assurer la fiabilité de transmission. Par contre, si on utilise un protocole dans la couche liaison (tel que l'Ethernet) ou un protocole non fiable dans la couche transport (tel que l'UDP) alors un mécanisme pour assurer la fiabilité de transmission est nécessaire. Nous n'avons pas encore étudié cette problématique et pour l'instant nous supposons que l'application qui utilise notre intergiciel prendra en charge la fiabilité de transmission des données dans le cas où les connexions n'utilisent pas un

protocole fiable dans la couche transport. Donc, dans notre étude, nous supposons que les noeuds dans le réseau recouvrant se connectent via le protocole TCP.

La deuxième phase, appelée pilotage, est une étape spécifique à notre intergiciel qui offre plus de flexibilité dans les réseaux recouvrants et particulièrement pour les communications mobiles, multipoints et les communications sécurisées comme nous l'expliquerons plus tard. La phase de pilotage consiste à lier les noms des noeuds dans le réseau recouvrant à leurs adresses avant ou pendant la transmission des données entre les noeuds. Cette phase est considérée comme une seconde étape dans le système de routage. Il peut lier aussi plusieurs noms logiques (des noms d'utilisateurs, de service ou de groupe) aux noms des noeuds dans le réseau recouvrant avant ou pendant la transmission des données entre les noeuds. Ces liaisons sont contrôlées par des noeuds qui acceptent de jouer le rôle de serveurs de noms.

4.2.2 Adressage

Dans un réseau recouvrant, chaque noeud a une adresse qui est composée d'un ou plusieurs champs (appelé aussi label). Chaque champ contient des chiffres séparés par des points et il correspond à un niveau dans la hiérarchie des adresses. Le niveau d'une adresse est égal au nombre des champs dans l'adresse. Le préfixe d'une adresse est égal à l'adresse sans compter le dernier champ qui est appelé le champ local ou le label local. Le nombre des niveaux dans la hiérarchie n'est pas fixe et il reste toujours dynamique donc sa valeur dépend de la façon dont les adresses sont distribuées à un moment donné. La longueur du label doit être fixée dès la création du réseau recouvrant. Chaque noeud dans le réseau a au minimum une adresse mais de préférence plusieurs et cela afin de faire face à la dynamique du réseau comme nous l'expliquerons plus tard.

Le plan d'adressage contient des zones qui correspondent aux champs de l'adresse. Donc, la taille du label détermine la taille maximale de la zone. Supposons maintenant que toutes les zones aient une taille fixe n appelée la taille de la zone. Il y a un niveau 1 (i.e., niveau supérieur) qui contient n noeuds (déterminé dans le premier champ de l'adresse). Puis il y a au maximum n zones de niveau 2 où chacune contient au maximum n noeuds (déterminé par les deux premiers champs). Puis, il y a au maximum n^2 zones de niveau 3 où chacune contient au maximum n noeuds, etc. Ceci signifie que tout l'espace d'adressage est distribué et si nous avons k niveaux avec l bits par niveau, nous pouvons déterminer $2^{k \times l}$ noeuds. Le but de cet adressage hiérarchique est principalement d'imposer la construction des zones d'une taille limitée afin de rendre le routage extensible. L'adressage des noeuds du réseau recouvrant est complètement distribué : il se base seulement sur la connaissance locale dans un noeud qui est le degré du noeud, les adresses et les degrés de ses voisins. Chaque noeud dans le réseau devrait connaître ces informations. Supposons que la taille de la zone soit n, chaque noeud qui a l'adresse $w.x.y$ est responsable de l'attribution des adresses suivantes à ses voisins :

- l'adresse dite ***suivante*** $w.x.(y+1)$ où $(y+1) \leq n$,
- l'adresse dite ***descendante*** $w.x.y.1$,
- les adresses dites ***feuilles*** $w.x.y.z$ où $z > n$.

Le premier noeud dans un réseau recouvrant prend l'adresse 1. Il s'agit du noeud qui lance la construction du recouvrement. Les noeuds suivants se connectent à lui puis entre eux en utilisant les protocoles de la couche transport (e.g., TCP, etc.) ou liaison. Comme ces protocoles sont utilisés pour configurer des connexions point-à-point entre les pairs du réseau recouvrant, d'autres protocoles peuvent être utilisés simultanément entre d'autres pairs. Les protocoles de la couche 2 peuvent être utilisés et il n'est pas nécessaire d'avoir des adresses uniques au niveau de la couche réseau pour les noeuds du recouvrement.

Afin de se connecter à un réseau recouvrant, les nouveaux noeuds et/ou les noeuds mobiles en déplacement doivent connaître au moins l'adresse d'un noeud dans le réseau sous-jacent (normalement son adresse IP ou Ethernet), on appelle cela le « bootstrapping ». Cette adresse doit être fournie par une méthode hors bande (e.g., E-mail, WWW, SDP, etc.) via l'application qui utilise ce réseau recouvrant. Les noeuds joignent le recouvrement successivement en se connectant à des noeuds qui sont déjà connectés. Donc, quand un noeud veut joindre le recouvrement, il demande des propositions d'adresse à tous les noeuds dans le recouvrement pour lesquels il connait l'adresse réseau sous-jacente. En général il connait au moins l'adresse IP du noeud ayant l'adresse 1 dans le recouvrement. Chaque noeud contacté propose une adresse parmi celles qui existent dans sa liste et qu'il n'a pas déjà donnée à un autre noeud. Puis, le noeud joignant le recouvrement choisit une ou plusieurs adresses selon les priorités suivantes :

- L'adresse la plus courte,
- En cas d'égalité, l'adresse suivante la plus courte,
- Si aucune adresse parmi les adresses suivantes n'est disponible, l'adresse descendante la plus courte,
- Si aucune adresse parmi les adresses descendantes n'est disponible, l'adresse feuille la plus courte.

La figure 4.2 montre le mécanisme présenté ci-dessus où le noeud joignant le recouvrement demande des adresses de ses voisins. Et comme nous l'avons expliqué, l'adresse feuille est une adresse dont le label local a une valeur plus grande que celle de la taille de la zone (e.g., si la taille de la zone égale à 32, le premier label d'une feuille est 33). Les noeuds qui ont une adresse feuille ne peuvent router les données que vers leur *père*, et cela même s'ils sont connectés à d'autres noeuds. Ils sont considérés comme des noeuds feuilles par rapport au système de routage dans le réseau recouvrant. C'est pourquoi les adresses feuilles sont toujours choisies en dernier recours (priorité minimale).

4.2.3 Routage

Le routage dans les réseaux recouvrants consiste à utiliser les adresses des noeuds afin de transmettre les paquets de données au niveau du réseau recouvrant à leur propre destination. Comme nous avons vu précédemment, les adresses obtenues par un noeud dépendent de ses voisins et ainsi de sa localisation. Donc, notre mécanisme de routage est *piloté par les adresses* (i.e., une partie de l'information sur le chemin est enregistrée dans les adresses). L'idée principale de notre architecture est que chaque noeud a besoin

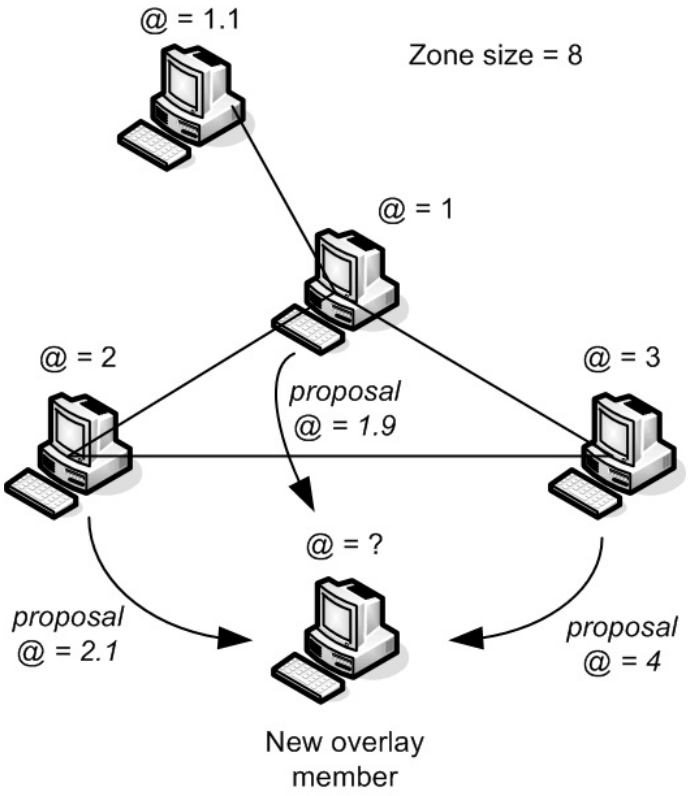

FIG. 4.2 – Le mécanisme suivi par le nouveau noeud pour joindre le réseau recouvrant.

seulement de savoir les adresses de ses voisins afin de router correctement les paquets vers leur destination. Par conséquent, sa table de routage contient seulement les adresses de ses voisins (nombre d'adresses minimal). Cependant, comme une partie du chemin vers la destination fait partie de l'adresse de la destination elle-même, la longueur de chemin dépend de l'efficacité de l'attribution des adresses.

Le fonctionnement du routage hiérarchique peut être expliqué de la manière suivante : quand un paquet est routé depuis le noeud source, si l'adresse de la destination est hiérarchiquement en dessous de ce noeud, le paquet sera routé au noeud qui existe dans la même zone et qui mène plus loin vers la zone de destination (nous l'appelons le noeud entrant). Si l'adresse de destination n'est pas sous celle du noeud, le paquet sera routé au premier noeud de la zone (i.e., celui avec un label local égal à 1) afin d'être envoyé vers la zone de niveau supérieur (nous l'appelons le noeud sortant). Quand un paquet est routé à l'intérieur d'une zone parce que la destination est dans la zone ou pour aller au noeud d'entrée ou de sortie, il sera routé en employant une technique que nous appelons la technique du label le plus proche où il faut seulement connaître les adresses des voisins. Donc, si le label local de la destination est inférieur à celui du noeud local actuel, le paquet sera transferé vers le noeud voisin qui a le label local le plus bas égal ou supérieur à celui de la destination. Mais si le label local de la destination est supérieur à celui du noeud local actuel, le paquet sera transferé vers le noeud voisin qui a le label local le plus haut égal ou inférieur à celui de la destination. Et comme l'attribution des labels aux voisins est faite d'une manière continue, cette technique garantit que le paquet atteindra sa destination mais pas nécessairement par le chemin le plus court.

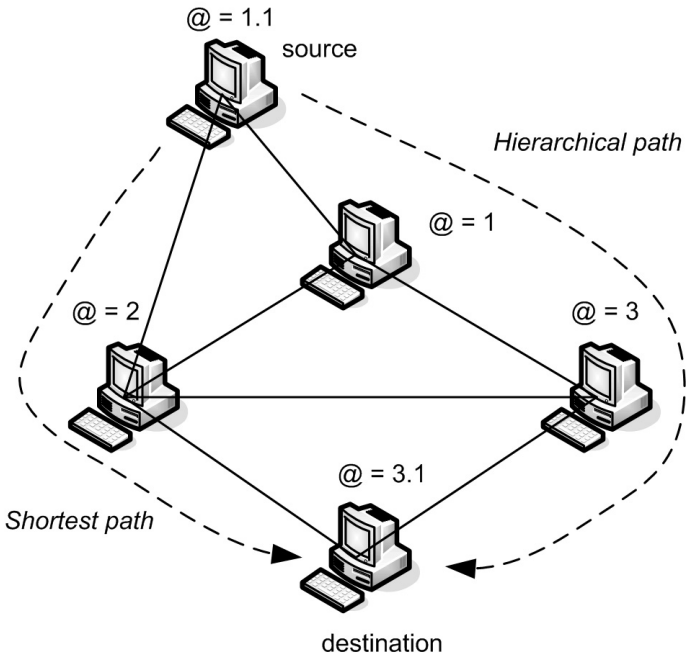

FIG. 4.3 – Le routage hiérarchique dans le réseau recouvrant.

La figure 4.3 montre les effets du routage hiérarchique et du routage par le chemin le plus court entre les noeuds 1.1 et 3.1. Ici, le routage hiérarchique oblige le message à être routé via 1 et 3, ce qui donne une longueur de chemin de 3 sauts tandis que le routage du chemin le plus court nécessite seulement 2 sauts via le noeud 2 pour qu'il arrive à destination. On peut définir l'inflation de la longueur du chemin (chemin non optimal) comme étant la valeur de la longueur du chemin du routage hiérarchique (en nombre de sauts) divisé par la longueur du chemin de routage le plus court.

Si un noeud se déplace ou tombe en panne, son adresse devient invalide et par conséquent, tous les paquets qui lui sont destinés ou à une destination dont l'adresse contient son adresse invalide ne pourront plus être transférés. Pour résoudre ce problème, deux techniques sont simultanément employées :

- Les adresses sont toujours valides temporairement (i.e., « soft state »). Ainsi les noeuds qui ont perdu leurs connexions récupéreront leurs adresses ou obtiendront de nouvelles adresses à partir des noeuds opérationnels (actifs) auxquels ils se reconnecteront.
- Chaque noeud peut utiliser simultanément plusieurs adresses fournies par des voisins différents, ce qui lui permet d'être toujours joignable par des chemins alternatifs si le chemin utilisé est rendu invalide par un noeud qui se déplace ou bien par un noeud tombé en panne.

Toutes les adresses ont une durée de validité v_1 qui doit être périodiquement renouvelée par un message *hello* envoyé par le(s) voisin(s) qui ont donné(s) ces adresses. Les adresses invalides et dérivées ne seront de ce fait pas régénérées. Si un noeud qui possède une adresse ne réapparaît pas (i.e., comme dans le cas d'un déplacement définitif ou bien

d'une panne permanente) après un certain temps v_2 ($v_2 > K_{depth} \times v_1$), le noeud responsable de cette adresse (i.e., celui qui a donné l'adresse) pourra attribuer cette adresse à un autre noeud. Dans ce cas, toutes les adresses dérivées de l'adresse invalide seront régénérées. Si toutes les adresses de la destination deviennent invalides pendant le transfert des données, le noeud source doit attendre jusqu'à ce que la connectivité vers la destination soit restaurée (i.e., les connexions sont rétablies et les adresses sont attribuées de nouveau).

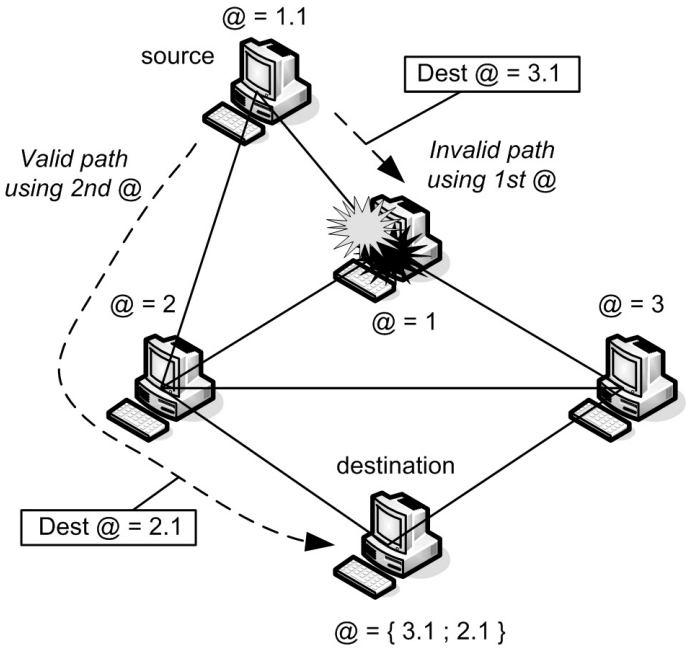

FIG. 4.4 – L'utilisation d'adresses multiples.

Donc comme nous l'avons expliqué précédemment, chaque noeud peut être localisé en utilisant plusieurs adresses (i.e., plus d'une adresse). Les adresses additionnelles peuvent être choisies dès l'insertion du noeud dans le réseau recouvrant ou plus tard quand la connectivité du réseau change, et par conséquent, le noeud aura plus d'adresses disponibles. De plus, toutes les adresses possédées par un noeud doivent être attribuées par des voisins différents. Ces adresses doivent être différentes et elles ne doivent pas avoir un préfixe commun. Sinon, si l'adresse invalide est inclue dans un préfixe commun, toutes les adresses dérivées ne fonctionneront pas. Cette attribution de plusieurs adresses augmente la quantité d'information de routage à stocker par un facteur égal au nombre d'adresses par noeud mais l'avantage est que la dynamique du réseau est gérée de manière transparente par le protocole d'adressage. Si un paquet ne peut pas être routé parce qu'un noeud a disparu, il peut utiliser une des adresses alternatives pour continuer vers la destination comme le montre la figure 4.4. Deux solutions sont possibles : soit le paquet contiendra toutes les adresses de la destination donc il peut être re-routé à la volée en utilisant ses adresses alternatives (ce qui augmente la consommation de la bande passante) ou bien soit un message d'erreur sera renvoyé vers la source qui va alors changer l'adresse de la destination par une autre adresse alternative qui sera utilisée dans les paquets suivants.

4.2.4 Nommage

Nous avons vu précédemment que chaque noeud dans un réseau recouvrant a une ou plusieurs adresses (pour faire face à la dynamique du réseau). Ces adresses dépendent de l'endroit où se trouve le noeud dans le réseau et de la dynamique du réseau (i.e., nouveau noeud, déplacement ou mouvement des noeuds). Afin de pouvoir communiquer d'une façon continue, chaque noeud dans le réseau recouvrant a un nom unique qui reste le même pendant toute sa durée de vie. Les applications employant notre architecture utilisent les noms des noeuds pour établir les liaisons entre elles. Par conséquent, tout changement d'adresse des noeuds sera transparent par rapport aux applications. De plus, la liaison entre les noms des noeuds et leurs adresses se fait par l'intermédiaire des serveurs de noms qui sont des noeuds réguliers de recouvrement qui acceptent d'effectuer des tâches de résolution des noms parce qu'ils ont plus de capacités et qu'ils se déplacent beaucoup moins que les autres noeuds (le noeud ayant l'adresse 1 est toujours un serveur de noms).

Afin d'assurer l'extensibilité du recouvrement, les serveurs de noms sont organisés dans une hiérarchie arborescente dont la profondeur plus un est égal au niveau maximal de la hiérarchie. Chaque serveur de noms a une table de hachage pour enregistrer les adresses des serveurs de noms dans le niveau suivant et une table de noms pour enregistrer les liaisons noms-adresses comme le montre la figure 4.5. Les noms sont composés de plusieurs parties où chaque partie correspond à un niveau dans la hiérarchie des serveurs de noms. En fait, ce n'est pas vraiment un problème si le nombre des niveaux est différent du nombre des parties. La hiérarchie des serveurs de noms est totalement indépendante de la hiérarchie d'adressage. De plus, l'extensibilité du stockage dans l'espace de nommage est assurée par une procédure basée sur le principe des tables de hachage distribuées.

En supposant qu'il n'y ait pas de dynamique au niveau des serveurs de noms (i.e., panne), quand un nouveau noeud joint le réseau recouvrant il va choisir une ou plusieurs adresses. Puis, il choisit un nom et envoie un message contenant ce nom et son adresse au noeud 1 afin d'enregistrer cette information dans le serveur de noms. A la réception, le noeud 1 fait un hachage sur la première partie du nouveau nom du noeud et envoie le message au serveur de noms correspondant dans le deuxième niveau. A son tour, à la réception, le serveur de noms dans le deuxième niveau fait un hachage sur la deuxième partie du nom et envoie le message comme avant vers le bas dans la hiérarchie des serveurs de noms. S'il n'y a plus de parties dans le nom pour faire le hachage ou s'il n'y a aucune entrée dans le résultat du hachage ou si la table de hachage dans le serveur de noms est vide, alors le dernier serveur de noms doit enregistrer le nom et les adresses dans sa table de nommage si ce nom n'existe pas déjà (sinon un autre nom doit être choisi par le nouveau noeud). Quand un noeud veut obtenir les adresses de la destination dont le nom est connu (le nom est censé être connu par un mécanisme externe), il envoie un message *requête* au noeud 1 contenant le nom à résoudre et sa propre adresse. Cette demande est expédiée au serveur de noms approprié en effectuant le hachage du nom exactement comme pendant la procédure d'enregistrement. Le serveur de noms qui contient le nom envoie un message de retour contenant les adresses du nom de la destination au noeud source (le noeud qui a envoyé la demande) en utilisant l'adresse de l'expéditeur qui existe dans la demande.

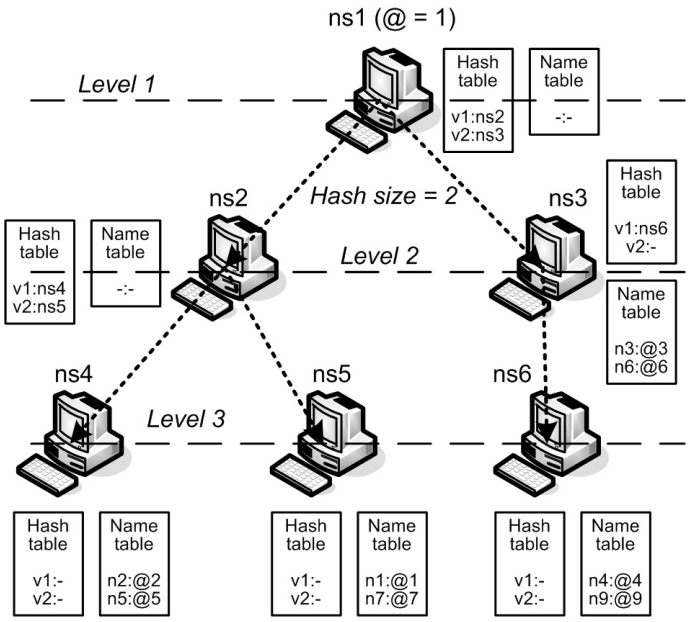

FIG. 4.5 – L'organisation hiérarchique des serveurs de noms.

Afin d'assurer l'équilibrage de charge et de faire face à la dynamique des serveurs de noms, nous employons une approche de réplication. Nous voulons que notre système de nommage soit résilient jusqu'à k échecs. Pour cela, nous supposons que le facteur de redondance k est choisi au début de la construction du réseau recouvrant où le nombre total de serveurs de noms est égal à s. Chaque serveur de noms a $k-1$ copies identiques appelés répliques. Les k noeuds dans le premier niveau ayant les adresses de 1 à k fonctionneront comme premier niveau de serveurs de noms et ils auront une copie des tables de hachage et de nommage du noeud qui a l'adresse 1. Par conséquent, ils auront les mêmes fonctionnalités agissant de ce fait en tant que serveurs de noms redondants. Les clients (i.e., les noeuds qui demandent des adresses) peuvent donc envoyer des messages de demande et d'enregistrement à tous les noeuds (de 1 jusqu'à k). En outre, chaque entrée de hachage dans n'importe quel serveur dans la hiérarchie enregistrera k serveur de noms au lieu d'un. Le résultat du hachage sera fourni aux k serveurs de noms appropriés qui sont tous opérationnels mais un seul parmi eux sera choisi aléatoirement pour recevoir les messages. Dans les niveaux les plus bas, et comme dans le premier niveau, les k serveurs de noms correspondants à une entrée du hachage devront maintenir les mêmes tables de hachage et de nommage puisqu'ils agissent en tant que serveurs de noms redondants. Ainsi, il y a un compromis entre assurer l'équilibrage de charge et la tolérance aux pannes et contrôler la réplication pour les s cliques de k serveurs de noms identiques. Nous envisageons également l'utilisation des caches de noms dans tous les noeuds du réseau recouvrant afin d'augmenter la performance.

La figure 4.6 montre la méthode utilisée pour enregistrer les noms dans le système de nommage avec $k = 2$ (i.e., une copie pour chaque serveur de noms). Nous supposons ici

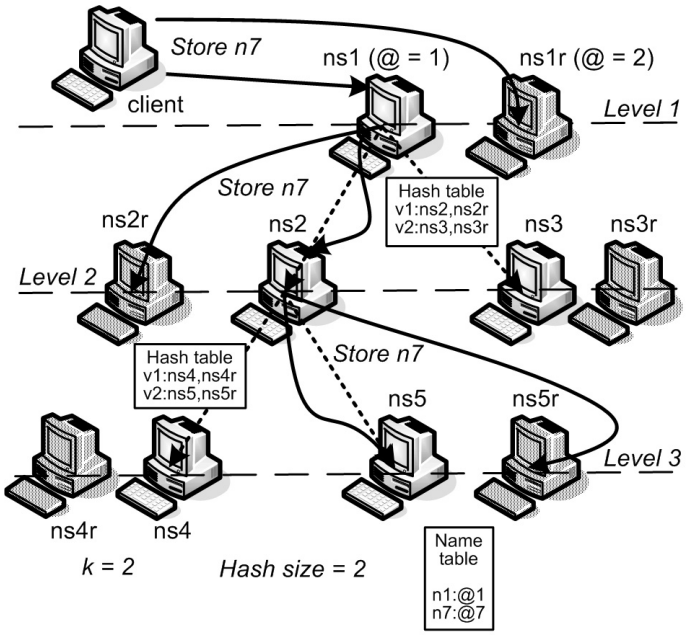

FIG. 4.6 – La procédure de stockage des noms.

qu'il n'y a aucun serveur en panne. Le noeud client n7 dans le réseau recouvrant envoie un message d'enregistrement au premier niveau de serveurs de noms qui sont les noeuds dont les adresses sont 1 et 2 (puisque $k = 2$). Puis, le premier serveur (qui a l'adresse 1 ici) transfère les messages d'enregistrement aux serveurs de noms dans le niveau 2 qu'il trouve dans sa table de hachage (ici, la première partie hachée de n7 est v1 et il réfère à ns2 et à ns2r). Ensuite, le serveur dans le deuxième niveau (ns2 ici) transfère les messages d'enregistrement aux serveurs de noms dans le niveau 3 qu'il trouve dans sa table de hachage (ici, la deuxième partie hachée de n7 est v2 et il réfère à ns5 et à ns5r). Enfin, ns5 et ns5r enregistrent l'adresse @7 du nom n7 dans leurs tables de nommage puisqu'ils sont dans le niveau le plus bas dans la hiérarchie.

Grâce à cette méthode toutes les copies de chaque serveur de noms ont la même table de nommage. De plus, chaque serveur de noms et chacune de ses copies maintiennent une liste d'eux-mêmes appelée « pool list ». Grâce à cette liste ils peuvent communiquer entre eux afin de garantir qu'ils ont la même table de hachage. Donc, comme nous pouvons remarquer dans la figure 4.6, chaque serveur de noms et chacune de ses copies a exactement les mêmes tables de nommage et de hachage. Si un serveur tombe en panne pendant une opération d'enregistrement, il peut après le redémarrage, attendre le prochain message d'enregistrement du client ou bien demander à une de ses copies en utilisant sa « pool list ».

La figure 4.7 montre la méthode utilisée pour réparer les noms dans un système de nommage avec $k = 2$. Nous supposons que le noeud qui a le nom n7 essaie de résoudre le nom de la destination afin d'obtenir son adresse. Pour cela, le client envoie un message de résolution à un des serveurs de noms dans le premier niveau. Ici, le serveur ns1 est

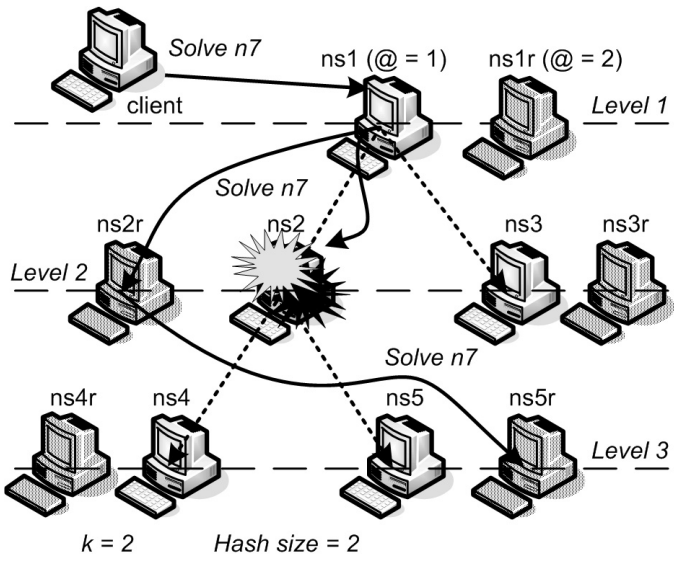

FIG. 4.7 – La procédure de résolution des noms.

fonctionnel, il essaie de transférer la demande à ns2. Et comme ns2 est en panne, ns1 essaie de transférer la demande vers la copie prochaine de ns2 qui se trouve dans sa table de hachage (à savoir : ns2r). Comme ns2r est fonctionnel, il reçoit la demande et il la transfère à ns5r au lieu de ns5 afin d'assurer l'équilibrage de charge. Cette méthode garantit la résilience du système de nommage.

Il y a au moins deux différences principales entre notre stratégie de nommage distribuée et le DNS [Moc87] qui lie les noms de domaines aux adresses IP. D'abord, les noms de domaines sont toujours des alias des adresses IP et quand une machine IP entre dans un réseau dont l'adresse IP est différente, elle obtient un préfixe IP différent, et par conséquent, elle ne pourra plus garder son nom de domaine (quelques services DNS dynamiques se mettent à jour en cas de changement d'adresse IP mais ils ne sont pas standards). Deuxièmement, notre solution ne se sert pas des appels itératifs comme dans le DNS. Les messages de client sont envoyés aux serveurs de noms au niveau supérieur qui les transfèrent aux serveurs qui sont plus bas dans la hiérarchie et finalement, le serveur qui a les adresses demandées répond directement au client. Enfin, notre infrastructure doit être totalement autonome pour tout réseau recouvrant afin de pouvoir être déployée par des hôtes sans aucune contrainte extérieure, ce qui exige de s'affranchir du DNS.

4.2.5 Pilotage

Le pilotage consiste à faire la liaison entre un nom et une adresse à tout moment pendant le processus de routage dans le réseau recouvrant. Ainsi, n'importe quel paquet dans le réseau contient les adresses et les noms du noeud source et du noeud destinataire (en cas d'envoi point-à-point). Le but principal du pilotage est de permettre aux applications d'établir des connexions en utilisant les noms plutôt que les adresses, ce qui rend tous les

changements de la topologie transparents aux applications. Le pilotage peut être fait au début de la connexion ou pendant la connexion si le noeud destinataire ou le noeud source change son adresse ou bien si un nom de groupe est utilisé comme nom de destination (en cas d'envoi multipoint).

La connexion entre deux noeuds dans un réseau recouvrant en utilisant leurs noms est appelée connexion virtuelle. Comme le nom de chaque entité est invariable et unique pendant toute la durée de sa vie, son utilisation peut restaurer les hypothèses originales mises sur les adresses d'Internet : unicité et pérénnité. Ces propriétés des adresses IP originales ont été exploités de plusieurs manières, particulièrement en les incorporant dans des identificateurs de transport. De plus, ils ont été construits dans des checksums de transport, des signatures cryptographiques, des documents Web, etc. En effet, la couche de nommage dans notre architecture peut garantir la propriété de transparence de bout-en-bout sur Internet qui n'existe pas actuellement à cause de l'attribution dynamique des adresses, des pare-feu, des NATs, etc. Enfin, la séquence de base pour établir une connexion point-à-point dans un réseau recouvrant entre deux noeuds A, B se déroule comme suit :

1. A joint le réseau en se connectant à un noeud C qui se situe près de A et qui est découvert par l'intermédiaire d'une source hors bande (Web, Email, etc.),
2. A extrait le nom de B à partir d'un annuaire distribué dans le réseau recouvrant (ceci est actuellement fait par les applications qui utilisent ce genre de réseau) ou d'une source hors bande (Web, Email, etc.),
3. A extrait l'adresse de B à partir d'un serveur de noms en utilisant le nom de B et en envoyant une demande à un des noeuds dans le réseau recouvrant qui a des adresses entre 1 et k (niveau 1 de serveurs de noms),
4. A envoie des paquets vers B à l'intérieur du réseau recouvrant en ajoutant sa propre adresse dans les premiers paquets pour que B puisse répondre sans questionner un serveur de noms.

4.3 Avantages de notre architecture

Les avantages de la séparation de l'espace de nommage de l'espace de l'adressage dans notre architecture sont multiples.

4.3.1 La convergence réseau

Quand il sera possible d'utiliser IPv6 partout avec en plus IPv6 mobile et du routage multipoint, on n'aura probablement plus besoin de notre intergiciel. Cependant, nous pensons que ce scénario idéaliste aura besoin de plusieurs années avant de devenir une réalité. Les applications distribuées pourraient vouloir utiliser maintenant des services de réseau avancés dans le monde réel où IPv4 et IPv6 co-existent, l'IP mobile et le routeur multipoint sont à peine déployés et l'adressage IP public prend en compte celui de IP privé employé par le NAT. Notre intergiciel est conçu pour fournir un espace d'adressage uniforme, une gestion de mobilité et un support multipoint pour les applications déployées actuellement sur Internet. Comme notre intergiciel est complètement autonome en terme

d'adressage, de routage et de nommage, seuls les ordinateurs qui veulent utiliser une application en profitant de notre architecture devront exécuter une instance de notre intergiciel. Bien évidemment, afin de passer dans un NAT, un chemin IP fonctionnel doit exister entre un ordinateur se situant dans un réseau privé qui utilise notre intergiciel et un autre ordinateur se situant dans un réseau public qui utilise aussi notre intergiciel.

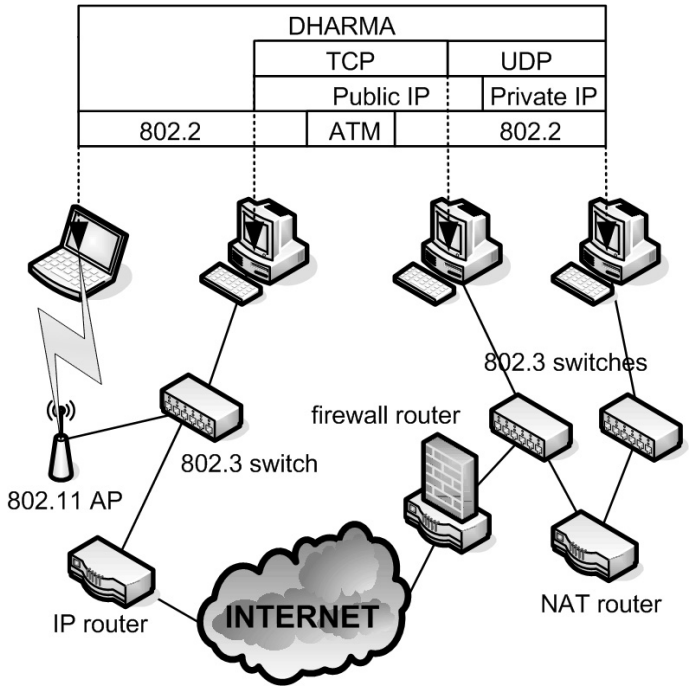

FIG. 4.8 – La convergence réseau avec notre intergiciel.

Le schéma 4.8 montre un scénario où quatre noeuds d'un réseau recouvrant se connectent en utilisant notre intergiciel. Chaque noeud est le voisin du noeud du réseau recouvrant situé à sa droite et d'un autre situé à sa gauche. Dans cette figure, nous pouvons voir la connectivité réelle entre les noeuds du réseau (au lieu de la connectivité du réseau recouvrant qui figure dans les autres schémas). L'ordinateur portable est connecté à l'ordinateur du bureau qui se situe à l'extrême gauche et cela est fait au niveau de la couche liaison du système OSI en utilisant le IEEE 802.2. L'ordinateur à l'extrême gauche se connecte à celui de centre et cela est fait au niveau de la couche transport en utilisant une connexion TCP. L'ordinateur du centre est connecté à celui de l'extrême droite aussi au niveau de la couche transport mais en utilisant UDP. Comme nous avons expliqué précédemment, ces connexions au niveau des couches transport ou liaison sont considérées comme des connexions point-à-point dans le réseau recouvrant.

D'un autre côté, la fiabilité doit être fournie par l'application si nous n'utilisons pas un protocole de transport fiable. Cependant, nous implémenterons la fiabilité dans notre intergiciel afin de libérer l'application de cette tâche. Dans ce scénario, si l'ordinateur portable change son adresse IP (à cause de DHCP ou de déplacement dans un autre sous-réseau IP) les communications ne vont pas être interrompues puisqu'elles se basent sur

les noms dans le réseau recouvrant. Et même si l'ordinateur portable se déplace dans un autre réseau WiFi, il aura seulement besoin de se connecter (procédure de démarrage) à un autre noeud du réseau recouvrant et d'avoir une nouvelle adresse dans ce réseau mais les communications seront toujours maintenues. De plus, le pare-feu peut être traversé par l'encapsulation des données de DHARMA dans des connexions TCP et en utilisant les ports ouvertes.

Le NAT peut être rendu transparent à l'application une fois qu'une connexion est établie entre un noeud dont l'adresse IP est privée et un autre noeud dont l'adresse IP est publique (la connexion est initialisée à partir du noeud privé). Ainsi, une application distribuée ou P2P peut entièrement fonctionner au-dessus de notre intergiciel même si les noeuds se situent dans un sous-réseau mixte où on trouve des adresses IP privées et publiques en même temps.

4.3.2 La mobilité

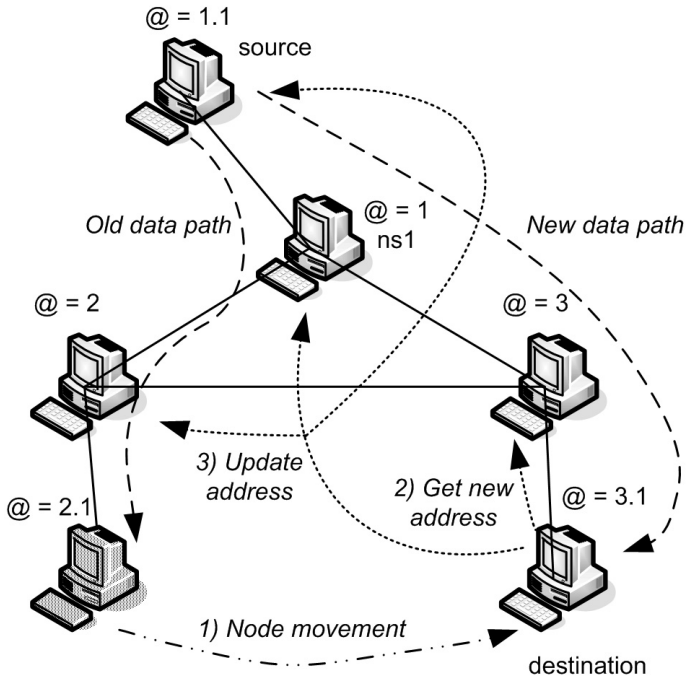

FIG. 4.9 – Scénario de mobilité.

Supposons qu'une machine destinataire mobile établisse une connexion avec une source correspondante ayant l'adresse 1.1. Le schéma 4.9 montre le noeud mobile, le noeud source et trois noeuds d'un réseau recouvrant. Une liaison est créée entre l'adresse originale mobile 2.1 et la source. Puis, les paquets sont routés dans le réseau recouvrant de la source en passant par les noeuds dont les adresses sont 1 et 2 vers le noeud mobile qui possède l'adresse 2.1 (indiqué par une flèche de ligne épaisse). Si le noeud mobile se déplace vers un nouvel endroit, il obtiendra une nouvelle adresse de son nouveau voisin et puis il enverra des messages de mise à jour (flèches légèrement pointillées) au serveur

de noms ns1 et à la source et en option à son voisin précédent ayant l'adresse 2 afin de leur donner sa nouvelle adresse (i.e., pour mettre à jour la liaison entre son nom et son adresse dans le cache de la source, du voisin précédent et dans le système de nommage). Seuls quelques paquets seront peut-être perdus mais alors, la source les enverra au noeud mobile par les noeuds 1 et 3 sans couper la liaison dans la couche application. Le message facultatif de mise à jour vers le noeud du réseau recouvrant dont l'adresse est 2 lui permettra de rerouter les paquets qui sont en retard vers la nouvelle adresse (i.e., paquets de pilotage). Ce mécanisme fonctionnera également pour la mobilité de réseau. Cependant, quand un réseau se connecte à un nouveau point du réseau, le nouveau préfixe devra être propagé partout à l'intérieur et des sous-préfixes devront être régénérés.

4.3.3 La sécurité

L'utilisation d'IPSec peut être entravée si on veut prendre en compte la mobilité dans l'architecture actuelle d'Internet. Dans IPSec, une association de sécurité (SA) est uniquement identifiée par un triplet se composant d'un index de paramètre de sécurité (SPI), d'une adresse de destination IP et d'un identificateur de protocole de sécurité (AH ou ESP) [KA98b]. Cependant, si la destination est mobile, SA sera invalide quand l'adresse IP du mobile change. Il est possible d'utiliser des boîtiers intermédiaires comme les pare-feu pour transférer le trafic en toute sécurité sur Internet mais comment faire si un espion se trouve à l'intérieur du réseau chez l'utilisateur ! La sécurité appliquée strictement de bout-au-bout est toujours plus sûre que l'utilisation d'un tunnel (encapsulation) et cela peut empêcher les pare-feu d'être des points sensibles des attaques. Dans notre intergiciel, en utilisant le nom au lieu de l'adresse, SA peut rester valide même si la topologie change (e.g., le mobile se déplace et change son adresse).

De plus, un nom dans un réseau recouvrant peut définir un utilisateur ou un groupe plutôt que simplement un noeud, ce qui est plus riche sémantiquement et permet ainsi de faire un contrôle des politiques de communication plus fin. Il facilite aussi la gestion et la surveillance. Au lieu d'avoir à autoriser tous les noms des noeuds possibles pour se connecter à un service donné, seulement le nom de la personne devra être enregistré dans la base des données.

Les pare-feu offrent également un grand avantage en utilisant le nom plutôt que l'adresse. En employant un nom se rapportant à un service au lieu d'un noeud (pour un serveur), les pare-feu pourront filtrer le trafic selon sa signification peu importe les conditions du réseau (e.g., il sera possible de filtrer des données chiffrées venant d'une source mobile). Ceci n'est pas possible actuellement, particulièrement quand un protocole ESP est utilisé [KA98a]. Car en chiffrant l'en-tête, le ESP empêche les pare-feu de filtrer selon le type de l'application. Enfin, la configuration des pare-feu sera plus facile et plus significative en employant des noms (pour les mêmes raisons que pour la sécurité de bout-au-bout).

Toutes ces problématiques nous amènent à conclure que le déploiement d'un réseau recouvrant sécurisé avec notre intergiciel est beaucoup plus flexible et efficace que le déployer au-dessus d'un réseau IP régulier.

4.3.4 Le multipoint

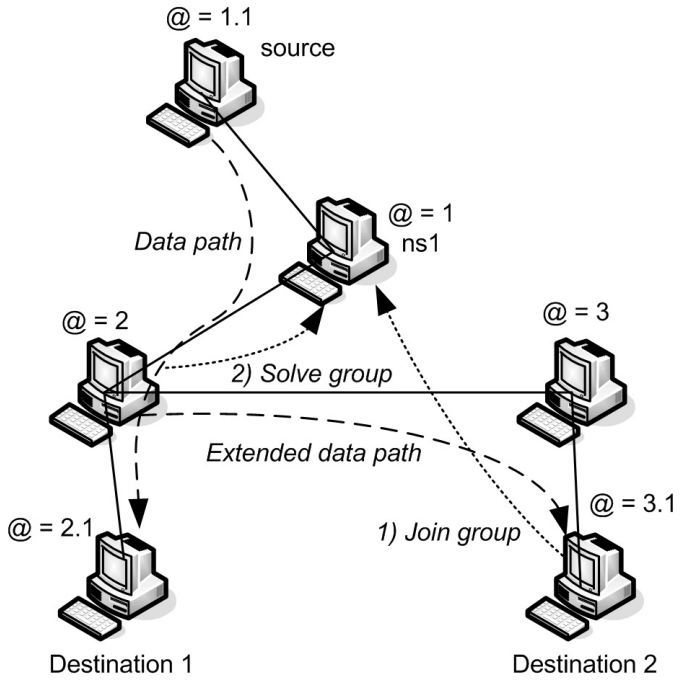

FIG. 4.10 – Scénario multipoint.

Dans la communication multipoint, deux niveaux de noms sont utilisés : un nom virtuel pour le groupe des noeuds et les noms des noeuds du réseau recouvrant participant au groupe. Les serveurs de noms enregistrent et résolvent tous les types de noms (groupe ou noeud). Supposons qu'une source multipoint dont l'adresse est 1.1, ayant enregistré le nom du groupe dans le système de nommage du réseau recouvrant, commence à transmettre un flux vidéo vers ce groupe. Comme nous pouvons le remarquer dans la figure 4.10, ce groupe est composé d'un noeud destinataire 1 dont l'adresse est 2.1 et qui sera accompagné par un autre noeud destinataire 2 dont l'adresse est 3.1. Le noeud 2 envoie un message pour joindre le réseau (flèche légèrement pointillé) au serveur de noms ns1 afin d'être lié au nom du groupe.

Les noeuds du réseau recouvrant qui se trouvent dans le trajet des données (i.e., les noeuds qui ont les adresses 1 et 2) résoudront le nom du groupe enregistré dans les paquets aux noms des noeuds dans le réseau recouvrant et reproduiront les paquets si nécessaire (i.e., si les noeuds membres du groupe ne sont pas accessibles par la même connexion de sortie). Ainsi, le noeud qui a l'adresse 2 reproduira les paquets comme il est indiqué dans le schéma par des lignes tirées épaisses.

4.4 Expérimentations

4.4.1 Paramètres

Afin d'évaluer notre architecture, nous avons employé 12k noeuds IPv4 (juillet 2004) et 4k noeuds IPv6 (Juin 2003), tous rassemblés par notre logiciel de topologie IP appelé NEC [HM]. Nous supposons dans une première approximation que les réseaux recouvrants déployés sur Internet peuvent être représentés comme sous-graphes de cette topologie.

Pour construire ce réseau, le premier noeud est aléatoirement choisi avec un degré moyen au niveau réseau ($>$10). Les noeuds et les liens sont graduellement ajoutés au réseau à partir des noeuds et des liens dans notre topologie.

Pour étudier la dynamique du réseau, nous avons analysé le pourcentage périodique du déplacement aléatoire des noeuds variant de 0 à 50% de la taille totale en attribuant de 1 à 4 adresses maximum à chaque noeud. La dynamique du réseau est une manière macroscopique pour simuler l'addition, le déplacement, le mouvement et l'échec des noeuds du réseau recouvrant. Au début des simulations tous les noeuds appartiennent au réseau recouvrant. Avant de commencer la simulation, x% des noeuds sont aléatoirement choisis et enlevés du réseau. Après chaque 10 tests, tous les noeuds enlevés du réseau sont réinsérés avec le même pourcentage, x noeuds sont aléatoirement choisis et enlevés du réseau. Bien que x reste le même, les noeuds actuels qui sont enlevés chaque fois seront différents la plupart du temps et particulièrement quand x est bas. Ceci simule l'addition, le déplacement, le mouvement et l'échec des noeuds du réseau recouvrant tout en gardant la taille du réseau égal $100 - x$%.

Pour ce qui concerne les serveurs de noms, nous avons choisi des noeuds aléatoires ayant un degré moyen du réseau ($>$5). Donc, pour la dynamique des serveurs de noms, nous avons étudié le pourcentage périodique du déplacement aléatoire des noeuds variant de 0 à 50% du nombre total des serveurs de noms. La dynamique du serveur de noms est traitée en tant qu'une dynamique régulière du noeud. Au début de la simulation tous les serveurs de noms sont actifs. Avant de commencer la simulation, x% des serveurs de noms sont aléatoirement choisis et marqués comme des serveurs de noms en panne. Après chaque 10 tests, tous les serveurs de noms marqués en panne sont réinsérés avec le même pourcentage, x serveurs de noms sont aléatoirement choisis et marqués comme des serveurs de noms en panne. Bien que x reste le même, les serveurs de noms actuels qui sont en panne seront différents la plupart du temps et particulièrement quand x est bas. Ceci simule l'échec aléatoire des serveurs de noms du réseau recouvrant tout en gardant la taille des serveurs actifs égal $100 - x$%.

Comme les processus de génération des adresses, de sélection des serveurs de noms et de sélection des noeuds source et destination nécessitent une sélection aléatoire, nous avons utilisé un scénario de simulation séquentiel [LK00] afin d'obtenir les résultats présentés dans les paragraphes suivants. Nous avons effectué les simulations en choisissant à chaque fois un noeud source et un noeud destinataire du réseau recouvrant pour déterminer :

– le succès de la résolution de nom du noeud destinataire,
– la distance (en sauts) de la résolution de nom (réponse incluse),
– la somme totale des paquets envoyés pour effectuer la résolution de nom,
– les distances hiérarchiques et plates entre le noeud source et le noeud destinataire,
– et le succès du routage hiérarchique en tenant compte de la dynamique du réseau.

Afin de réduire les coûts du calcul, la création des plans d'adressage et de nommage avec le placement des serveurs de noms sont faits chaque 100 tests, les noeuds du réseau et la dynamique des serveurs de noms décrits précédemment sont faits chaque 10 tests et les points du contrôle sont effectués chaque 5 tests. Ceci explique les petites fluctuations apparues dans les schémas. Tous les résultats des simulations ont été obtenus en supposant que le niveau de confiance est de 0.95 avec un seuil d'erreur statistique et relatif de 5% pour toutes les matrices mesurées. Les simulations ont été effectuées dans notre simulateur statique appelé NEM [Mag].

4.4.2 Résultats

Dans toutes les simulations effectuées [SML06], nous avons remarqué que les résultats (en pourcentage) obtenus avec IPv6 sont très proches de ceux qui sont obtenus avec IPv4 (e.g., nombre des noeuds dans le réseau recouvrant, la longueur moyenne du chemin, etc.). Pour cela, nous allons présenter seulement les résultats obtenus en utilisant IPv4, sauf si on mentionne le contraire d'une manière explicite.

Le schéma 4.11 montre le pourcentage des essais du routage réussis en fonction du pourcentage de la dynamique du réseau. Comme nous avons expliqué précédemment, il y a un certain pourcentage de noeuds absents, donc le réseau recouvrant ne peut pas être connecté mais il sera composé de plusieurs composants connectés. Le pourcentage est calculé en tant que nombre des essais réussis du routage hiérarchique divisé par le nombre des essais réussis du routage plat. Comme le chemin hiérarchique est plus long que le chemin plat (i.e., le plus court), il peut dévier du composant source - destination et par conséquent, il fera échouer le routage. Nous pouvons constater cela avec une seule adresse (i.e., aucune route alternative) où 20% de la dynamique de réseau peut diminuer le taux de réussite au-dessous de 20%. Cependant, l'addition des adresses aux noeuds augmente fortement le succès du routage. Par exemple, jusqu'à 4 adresses par noeud et 20% de dynamique, le succès atteint 55%. De plus, nous remarquons que l'augmentation du nombre maximal des adresses par noeud n'améliore pas linéairement le succès parce que le nombre maximal des adresses par noeud est encore lié à sa taille de voisinage (et ceci est petit pour la plupart des noeuds à cause de la topologie d'Internet).

Nous présentons dans la figure 4.12 le chemin non optimal en fonction du pourcentage de la dynamique du réseau. D'abord, nous pouvons remarquer que la valeur du chemin non optimal est d'environ 2.3 quand tous les noeuds du réseau recouvrant sont opérationnels. Ce résultat obtenu sans tenir compte de la dynamique du réseau semble être cohérent quand on le compare avec le travail de D. Magoni [Mag03a]. De plus, ce pourcentage

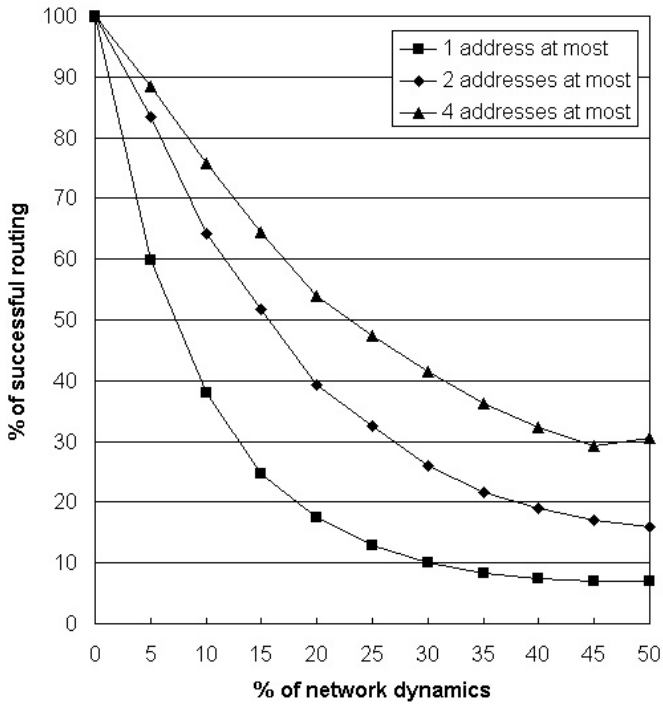

FIG. 4.11 – Le succès du routage *vs* la dynamique du réseau.

est considéré comme un bon résultat pour un protocole du routage au niveau application. D'un autre côté, on peut constater que le chemin non optimal diminue quand le pourcentage de la dynamique du réseau augmente et cela est justifié par l'augmentation du nombre de composants du réseau. Ces composants deviennent de plus en plus petits à cause de la fragmentation du réseau. De plus, les chemins à l'intérieur de ces composants se comportent ainsi.

Nous avons vu précédemment que les adresses sont stockées dans un état « soft-state » et doivent être régénérées à un intervalle de temps régulier prédéterminé. Nous appelons cette durée *période de temps*. Nous supposons que cette période de temps a la même valeur pour tous les noeuds dans le réseau recouvrant mais elle pourrait également dépendre de la mobilité du noeud (nous laissons cette évaluation comme perspective).

Le schéma 4.13 montre le pourcentage des essais réussis du routage, comme défini précédemment, en fonction du nombre des périodes de temps. Chaque ligne correspond à un pourcentage donné de la dynamique du réseau variant de 10% à 50%. Dans ces lignes, chaque noeud pourrait avoir jusqu'à 4 adresses maximum (les lignes avec les adresses maximales les plus bas par noeud sont considérées comme les cas les plus mauvais mais elles ont les mêmes caractéristiques). Ce schéma montre le pourcentage de la chance qu'un noeud source dans un réseau recouvrant atteigne un noeud destinataire dans ce réseau. Nous pouvons remarquer aussi qu'avec quelques périodes du temps (pratiquement plus de 5), il y a une convergence de l'algorithme d'adressage qui garantit quand même le succès du routage (au-dessus de 95%).

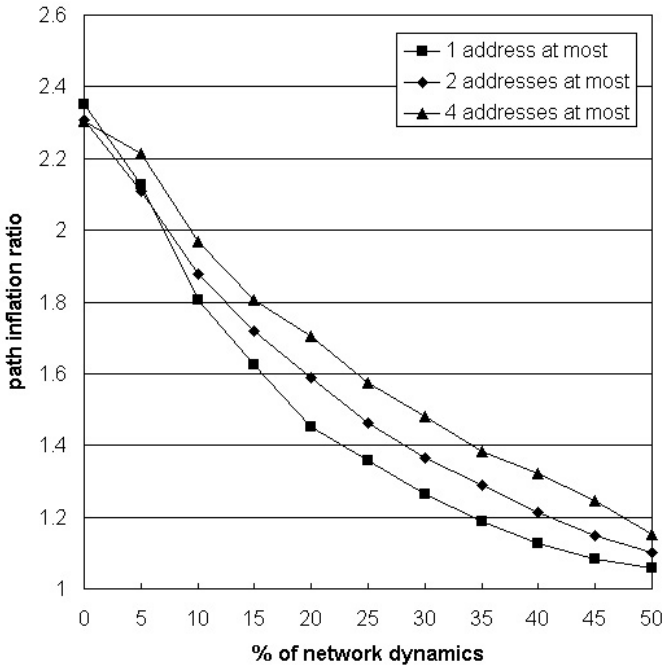

FIG. 4.12 – Le chemin non optimal *vs* la dynamique du réseau.

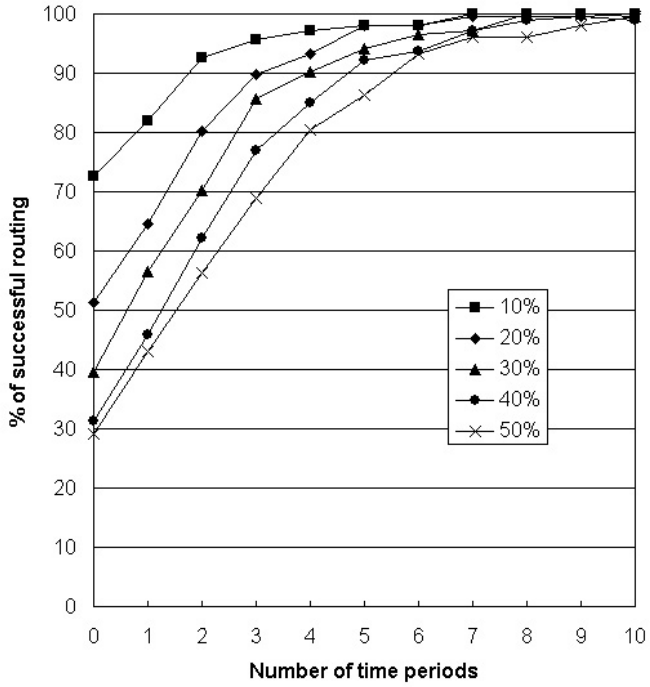

FIG. 4.13 – Le succès du routage *vs* le nombre de périodes de temps.

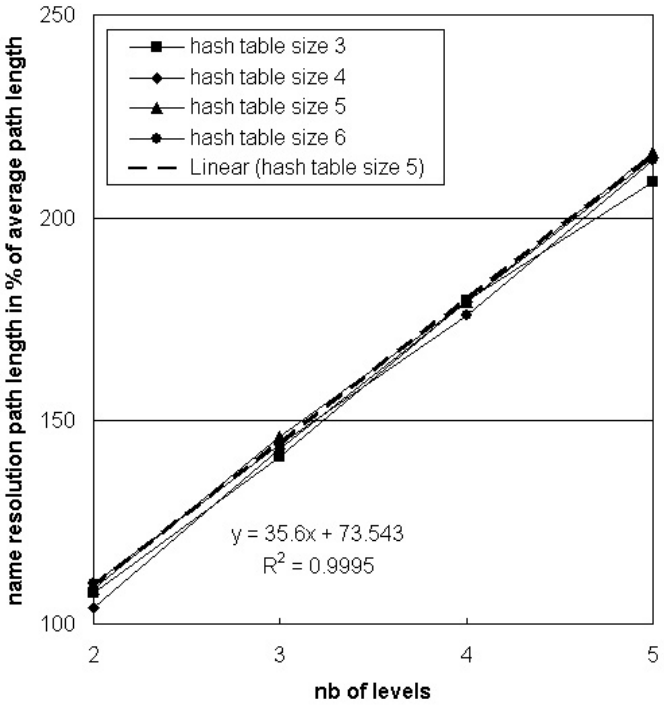

FIG. 4.14 – La longueur du chemin de la résolution de nom *vs* le nombre de niveaux.

Nous allons maintenant évaluer les performances de l'efficacité et de l'extensibilité de notre système de résolution de nom. Nous pouvons voir dans la figure 4.14 la longueur moyenne du chemin ou la distance d (représentée comme un pourcentage de la distance aller-retour moyenne en nombre de sauts) de la résolution de nom (réponse incluse) en respectant le nombre des niveaux dans la hiérarchie l et la taille de la table de hachage h. Rappelons que h est le nombre maximal de serveurs de noms dans le prochain niveau qui se situe au-dessous d'un seul serveur de noms (degré de sortie/nb de sorties). La distance augmente quand les niveaux augmentent et les lignes montrent un ajustement linéaire (comme on peut le remarquer dans le schéma pour une taille de table de hachage égale à 5). Cependant, les valeurs restent toujours au-dessous de 2.5 fois l'aller-retour moyen entre n'importe quelle paire de noeuds. Ces résultats prouvent que la résolution de nom a un coût de distance (ainsi un délai) raisonnable. En effet, avec 5 niveaux dans la hiérarchie on peut traiter une très grande quantité de noms. De plus, nous pouvons également remarquer que la distance ne change pas en fonction de la taille de la table de hachage car toutes les lignes sont très proches les unes des autres. La taille de la table de hachage a un effet uniquement sur la répartition de la charge des noms à chaque niveau. Ainsi nous pouvons écrire :

$$d \propto l \tag{4.1}$$

avec :

$$l << 1 \tag{4.2}$$

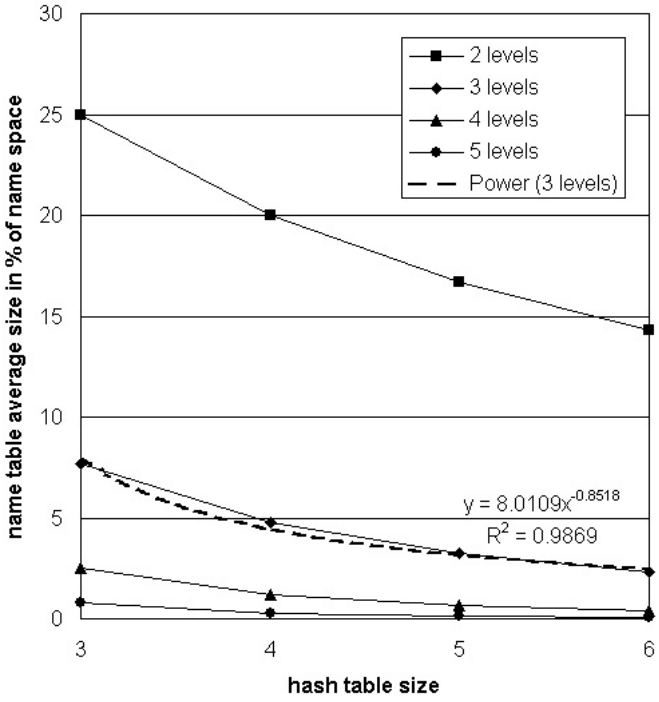

FIG. 4.15 – La taille de la table de nommage *vs* la taille de la table de hachage.

La figure 4.15 montre la taille de la table de noms n (exprimé comme un pourcentage dans l'espace de nommage) en fonction de la taille de la table de hachage h et du nombre de niveaux dans la hiérarchie l. Ici, nous pouvons remarquer que n est une fonction de h élevé à la puissance d'un nombre fixe où l est fixe aussi (comme on peut voir dans la figure pour 3 niveaux dans la hiérarchie) et qu'il diminue quand h augmente. Ceci peut être traduit en équations mathématiques. En supposant que s soit le nombre des serveurs de noms dans notre système, nous obtiendrons :

$$s = \frac{h^l - 1}{h - 1} \tag{4.3}$$

De plus, si les noms des membres du réseau recouvrant m sont équitablement distribués dans les s serveurs, ces noms seront stockés dans les feuilles d'un arbre h équilibré, donc on peut écrire ainsi :

$$n \approx \frac{m}{h^{l-1}} \tag{4.4}$$

Nous pouvons expliquer ceci par le fait que l'augmentation de la taille de la table de hachage augmente le nombre des serveurs et diminue ainsi la charge sur chaque serveur. Pour la même raison, nous pouvons remarquer que le nombre des niveaux a aussi un effet important sur les tailles de la table de noms.

La figure 4.16 montre la distance moyenne d de la résolution de nom en fonction de la taille de la table de nom n. Les équations 4.1 et 4.4 donnent :

$$d \propto \frac{\log \frac{n}{m}}{\log h} \tag{4.5}$$

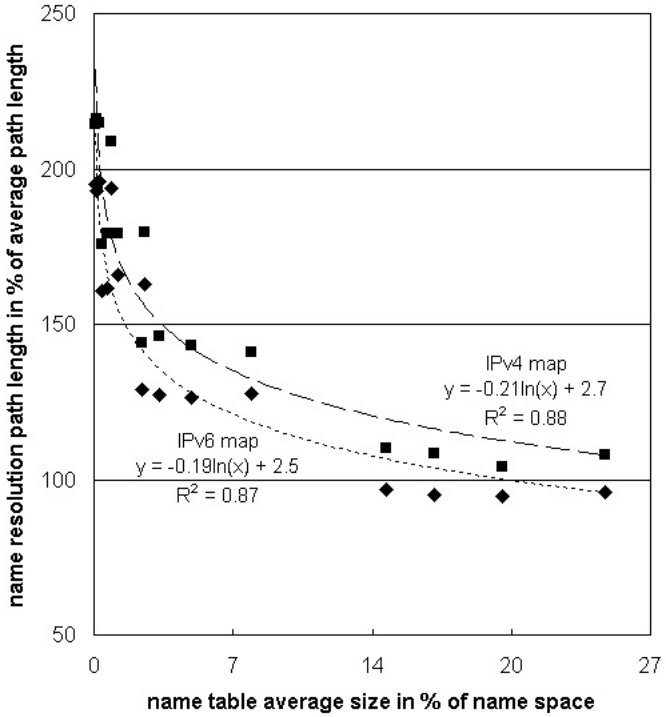

FIG. 4.16 – La longueur du chemin de la résolution de nom *vs* la taille de la table de nommage.

Cependant, nous pouvons remarquer que les lignes ne s'accordent pas parfaitement avec les données (avec seulement 0.88 comme coefficient de corrélation). Ce phénomène s'explique par les hypothèses que nous avons faites. Néanmoins, cette figure montre aussi que nous pouvons avoir un bon compromis entre le coût de la résolution de nom et le coût de stockage dans chaque serveur de noms en choisissant les valeurs dans le secteur bas-gauche dans la figure.

Le schéma 4.17 montre la taille de la table de nommage n (exprimée comme un pourcentage du nombre de noms) en fonction du nombre des serveurs de noms (exprimé comme un pourcentage du nombre des membres dans le réseau recouvrant). Nous pouvons voir que n est inversement proportionnel au m. Ceci est peut être exprimé aussi mathématiquement. Les équations 4.3 et 4.4 donnent :

$$s \propto \frac{\frac{mh}{n} - 1}{h - 1} \tag{4.6}$$

ce qui donne :

$$s \propto \frac{mh}{n(h-1)} \tag{4.7}$$

si :

$$\frac{mh}{n} \gg 1 \tag{4.8}$$

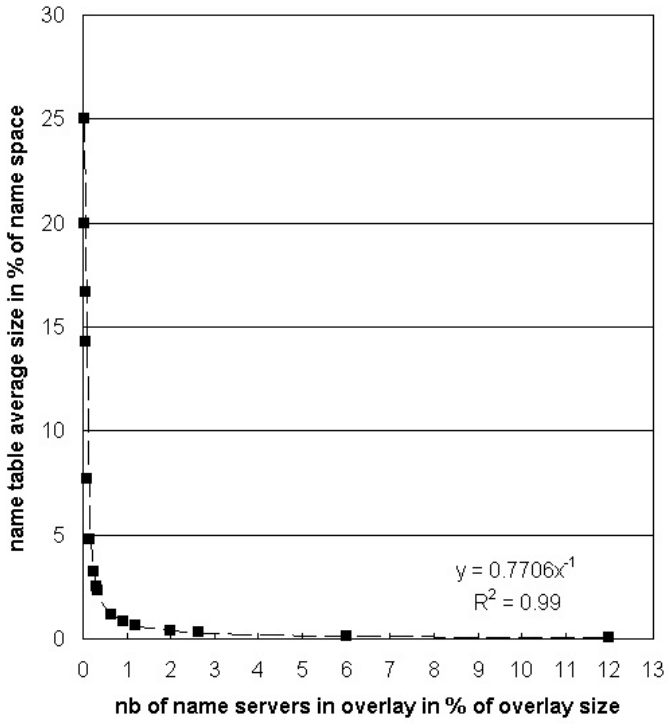

FIG. 4.17 – La taille de la table de nommage *vs* le nombre de serveurs de noms.

Quand m et h sont des valeurs fixes nous obtenons :

$$n \propto s^{-1} \tag{4.9}$$

Le schéma montre aussi que nous pouvons avoir un bon compromis entre le pourcentage des serveurs de noms et la taille des tables de noms à stocker dans chacune d'elles en choisissant des valeurs dans le secteur bas-gauche dans la figure.

Nous évaluons maintenant les performances de la résilience de notre système de résolution de noms. Nous fixons d'abord l et h afin de réduire l'espace des paramètres. Les résultats représentés précédemment dans les schémas 4.14 et 4.15 nous encouragent à choisir $l = 3$ niveaux d'hiérarchie afin de maintenir une distance de résolution de nom raisonnable et une taille de table de hachage $h = 6$ afin de limiter la taille de la table des noms. Ces paramètres donnent un certain nombre de serveurs de noms de 43, chacun a une table de nommage qui contient presque 300 noms. Rappelons nous que la taille totale du réseau recouvrant est égale à 12977 noeuds. Et les serveurs de noms représentent ainsi 0.33% des noeuds du réseau et ils tiennent chacun 2.3% de l'espace de nommage. Si nous regardons le schéma 4.17, nous pouvons remarquer que ces valeurs se représentent par un point situé dans le secteur en bas-gauche de la figure où le compromis entre la taille de la table de nommage et la quantité des serveurs de noms est optimal. Nous avons évalué aussi les valeurs redondantes k variantes de 1 (aucune copie) à 8 (7 copies par serveur de noms). Ainsi le nombre des serveurs de noms varie de 43 à 344. Ceci reste appro-

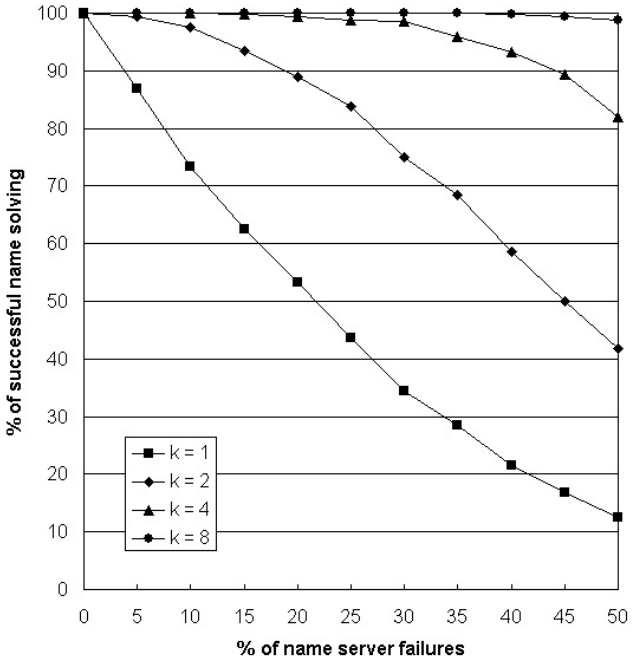

FIG. 4.18 – Le pourcentage des résolutions réussies *vs* les pannes de serveurs de noms.

prié et donne le nombre total des noeuds du réseau recouvrant. Bien que les duplicatas augmentent le nombre global des serveurs de noms, la taille de la table de nommage des serveurs de noms reste la même (i.e., autour de 300).

Le schéma 4.18 montre le pourcentage de la résolution de nom réussi en fonction du pourcentage de la dynamique des serveurs de noms. Comme nous avons expliqué précédemment, le pourcentage des serveurs de noms indiqué est en baisse. Nous pouvons voir cela sans avoir des copies des serveurs de noms où 20% de la dynamique peuvent faire chuter le taux de réussite de la résolution de noms au-dessous de 50%. Cependant, l'addition de 3 duplicatas pour chaque serveur de noms augmente fortement le succès de la résolution de nom. Donc, avec jusqu'à 3 duplicatas par serveur de noms et 50% de la dynamique, le taux de réussite reste toujours au-dessus de 80%. Par contre, l'augmentation du nombre des duplicatas de plus de 3 par serveur de noms n'améliore pas linéairement le succès puisqu'il s'approche de 100%.

La figure 4.19 montre le nombre moyen des temps durant laquelle une demande échoue entre un client et un serveur de noms ou entre deux serveurs de noms car un serveur de noms tombe en panne. La valeur maximale est de $l \times (k-1)$ pour $k > 1$ et de 1 pour $k = 1$. Comme nous avons 3 niveaux dans la hiérarchie, la valeur maximale est 3 pour $k = 2$, 9 pour $k = 3$ et 21 pour $k = 8$. Donc en fonction de k, le schéma montre que cette valeur augmente quand le pourcentage des échecs augmente. Sans réplication, le schéma montre une tendance rapide vers 1 mais pour $k > 1$, nous pouvons constater que le système est loin d'utiliser tous les essais possibles quand le pourcentage des échecs

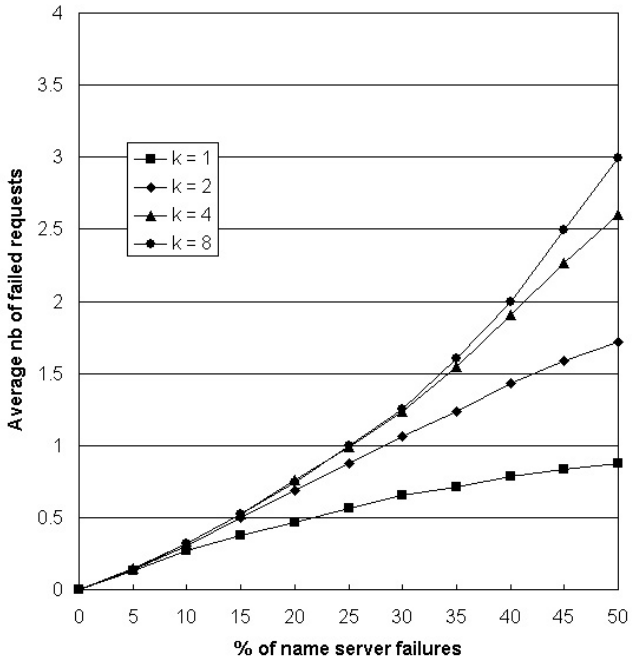

FIG. 4.19 – Le nombre moyen des demandes échouées *vs* les pannes des serveurs de noms.

est égal ou moins de 50%. Pour $k = 8$ et 50% d'échecs, cette valeur est juste au-dessus de 3 tandis qu'elle pourrait atteindre 21.

La figure 4.20 montre la distance moyenne couverte par la résolution des noms et représentée en sauts (réponse inclue) qui est exprimée comme le pourcentage de la distance moyenne de l'aller-retour mesurée dans le réseau recouvrant. Pour cela, nous avons effectué cette mesure en fonction des niveaux représentés dans la figure 4.14 (sans réplication) et nous avons remarqué que cette valeur est égale à 150% pour 3 niveaux dans la hiérarchie. De plus, nous avons obtenu la même valeur pour 0% de panne (en tenant compte de l'erreur relative et statique de 5%). Pour $k > 3$, cette valeur augmente presque linéairement quand la quantité des demandes échouées augmente (et ainsi la résolution des serveurs de noms). Pour $k > 2$, cette valeur augmente un peu quand le pourcentage des pannes augmente. Cela prouve que la plupart de demandes réussissent (en faisant ainsi l'aller-retour) même si le nombre des demandes relancées augmente (afin d'éviter les serveurs en panne) ce qui rend les chemins parcourus un peu longs.

Enfin, le schéma 4.21 montre le nombre total moyen des paquets envoyés pour chaque résolution de nom (réponse inclue). Nous pouvons remarquer que les lignes dans cette figure sont très proches de celles dans la figure 4.20 bien que les valeurs sont différentes. De plus, le nombre total des paquets diminue quand k diminue. Par contre, ce nombre augmente quand k augmente et on aura plus de possibilité d'avoir un grand nombre de demandes alternatives via les copies des serveurs de noms.

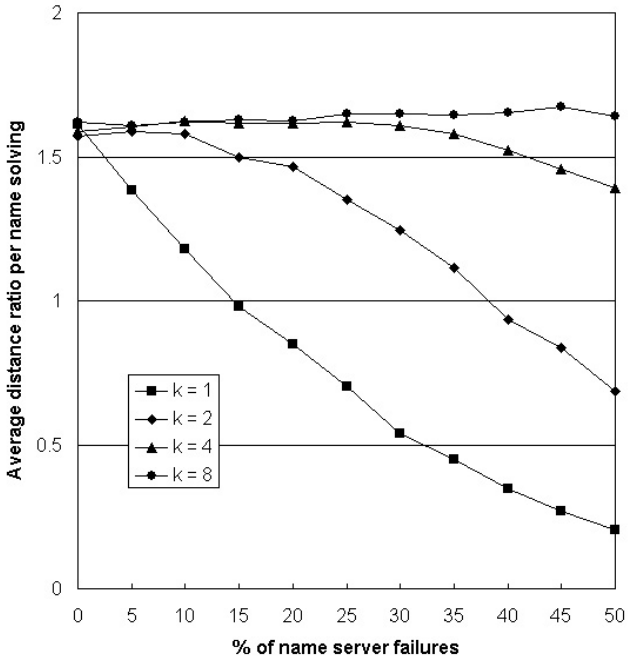

FIG. 4.20 – La distance moyenne de la résolution de nom *vs* les pannes des serveurs de noms.

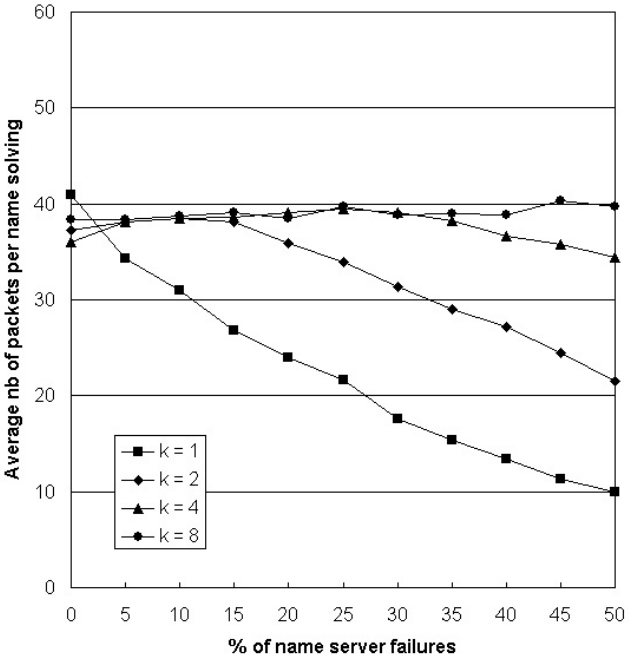

FIG. 4.21 – La quantité moyenne des paquets envoyés *vs* les pannes des serveurs de noms.

4.5 Conclusion

Dans ce chapitre, nous avons présenté un intergiciel extensible, résilient et autonome basé sur une plateforme d'adressage, de routage et de nommage hiérarchique et distri-

buée. Cet intergiciel est conçu pour des applications créant et employant des réseaux recouvrants déployés sur Internet. Nous avons défini un algorithme de routage local basé sur une localisation efficace dirigée par l'adressage.

Dans ce travail, nous avons expliqué comment faire face aux aspects dynamiques du réseau Internet et les simulations ont prouvé que notre schéma d'allocation de plusieurs adresses multiplie par 2 le pourcentage de réussite du routage quand la dynamique du réseau est égale ou supérieure à 10%. Nous avons également prouvé que le routage est presque garanti avec assez de temps de convergence (5 périodes de temps ou plus). Enfin, nous avons conçu un système de liaison entre les noms et les adresses qui est autonome, extensible et distribué afin de séparer efficacement les espaces d'adressage et de nommage. Cela permet aux applications d'utiliser des services réseaux avancés comme la mobilité et le multipoint.

Les résultats des simulations montrent que la relation entre le nombre de serveurs de noms et les tailles des tables de nommage peut être optimisée. Nous avons également montré qu'en utilisant une quantité raisonnable de repliques de serveurs de noms (3 ou plus), nous pouvons faire face à des pannes de serveurs de noms allant jusqu'à 50% tout en maintenant un pourcentage de résolutions de noms réussies au-dessus de 80%. Nos simulations ont été faites sur des réseaux recouvrants ayant plus de 12000 noeuds ce qui peut prouver que notre intergiciel est extensible aux réseaux recouvrants très grands.

Chapitre 5

Utilisation de DHARMA dans le cadre d'une application P2PTV

5.1 Introduction

La télévision sur Internet connaît un essor fulgurant. Cette technologie qui grandit et se développe rapidement commence à concurrencer la télévision traditionnelle. Avec un accès à haut débit, d'innombrables chaînes peuvent être visualisées sur l'ordinateur, sans contrainte horaire ni géographique, y compris des chaînes originaires de pays lointains. Toutes les applications qui emploient actuellement la technique IPTV se basant sur des réseaux pair-à-pair sont capables de transmettre la télé n'importe où et à n'importe qui.

Dans ce chapitre nous allons étudier certains problèmes liés à la diffusion de la vidéo sur Internet. Nous allons utiliser des données réelles d'un système P2PTV fournies par une entreprise appelée Zattoo [Zat]. Ici, l'idée principale est d'étudier l'effet du NAT et de le surmonter en utilisant notre architecture DHARMA.

Nous allons tout d'abord aborder brièvement le principe du P2PTV. Puis, nous allons aborder l'extensibilité de la redistribution dans les réseaux P2PTV. Après, nous allons détailler des données réelles obtenues d'un système P2PTV et décrire quelques caractéristiques des pairs dans les réseaux P2PTV. Enfin, nous allons montrer des résultats de l'évaluation de performance qui sont obtenus par des simulations effectuées sur un réseau P2PTV en utilisant notre architecture DHARMA.

5.2 Principe du P2PTV

Un réseau P2PTV est composé d'une source et de plusieurs clients. Ces clients sont connectés directement à la source afin de regarder des flux vidéo. Dans le cas où la source est saturée, les clients cherchent d'autres clients qui sont déjà connectés afin de continuer à recevoir les flux vidéo et ils construisent ainsi un réseau P2P. Normalement, une source est aidée par un répéteur qui est utilisé pour augmenter la capacité du système. Le flux vidéo est un flux de télévision diffusé en ligne. Ce flux peut être de type analogique ou digital codé en H264. Il est présenté par des tranches/couches (unité d'amplitude) et par bloc *chunks* (unité de temps). Ce type de flux est transféré par des connexions TCP/UDP

et de type IP unicast. Nous supposons l'absence d'IP multicast car actuellement il n'est pas encore disponible sur Internet. Pratiquement, il y a un seul réseau P2P recouvrant pour chaque chaîne diffusée (chaque source peut diffuser une ou plusieurs chaînes à la fois). Ainsi, les réseaux recouvrant P2PTV sont représentés par un graphe acyclique routé et dirigé par la source.

D'ailleurs, la notion du transfert P2P est différente de celle du partage de données ou du transfert des flux audio/vidéo asynchrones (VoD). Car ici, l'augmentation de la vitesse du téléchargement au dessus du débit des données vidéo du flux n'est pas nécessaire (sauf pour la récupération en cas d'erreur). De plus, il n'y a pas besoin de stockage des données et la mémoire tampon est minimisée seulement pour certains cas (l'utilisation d'une fenêtre temporelle significative). Par contre, les paramètres les plus importants dans le réseau recouvrant sont la capacité de téléversement et la durée/la longueur de la session car ils déterminent l'extensibilité et le remous (i.e., la quantité de pairs qui apparaissent ou disparaissent dans une période donnée) ou *churn* en anglais. Enfin, les applications P2PTV sont très sensibles au délai et à la moindre augmentation du taux de perte.

En fait, les motivations de ce travail sont :
- Les réseaux P2PTV ne sont pas forcement extensible, pour cela nous devrons essayer d'assurer l'extensibilité en appuyant sur la redistribution.
- Les réseaux P2PTV peuvent être inefficaces, pour cela nous devrons essayer d'assurer ou d'améliorer leur efficacité en améliorant la sélection des pairs.

Actuellement, les caractéristiques des réseaux P2PTV sont mieux connus et plusieurs études ont été publiées sur le P2PTV. Cependant, plusieurs mécanismes proposés n'ont pas été bien développés et à ce jour, il y a très peu d'études proposées afin de garantir une amélioration importante. Pour cela, nous proposons quelques algorithmes de sélection des pairs basés sur les caractéristiques des pairs suivants :

- La capacité du téléchargement vers une machine distante (téléversement)
- La durée de la session

Le concept des flux multimédia, et plus précisément les flux multimédia en ligne, n'est pas nouveau. Depuis 2003, plusieurs prototypes de recherche ont été crées et évalués, parmi ces prototypes citons ZigZag [THD03], PRO [RS04], Anysee [LJL+06], CoolStreaming [ZLLY05] et enfin PULSE [Pia06, PPKB07].

Logiquement, plusieurs systèmes opérationnels ont apparu en 2005 et les systèmes les plus populaires ont été développés en Chine (PPLive, PPStream, SopCast, TVAnts). D'autres initiatives remarquables ont émergé en Grande Bretagne (LiveStation), aux Pays-Bas (Joost) et en Suisse (Zattoo). Par conséquent, plusieurs articles ont été publiés récemment sur le charge de travail à gérer [SMZ04], les topologies [WLZ07], des systèmes spécifiques (e.g., Telefonica IPTV service [CRMC08] et Cool-Streaming [XKL07]). Comme PPLive est devenu l'un des systèmes le plus important actuellement, il a été largement étudié dans [HLL+07], [VGLN07], [AMZ06] (avec SopCast) et dans [MPRG07].

D'autres projets qui ont pour but d'améliorer les systèmes de transfert des flux en ligne P2P sont en cours. Parmi ces travaux il y a ceux qui étudient la faisabilité [SGMH04], la capacité du téléchargement vers une machine distante [DLM07], la connaissance de la localisation [PP07], l'altruisme [CZ04] et la contribution de connaissance [SBR06]. Enfin, [KLR07] ont proposé une théorie stochastique pour les systèmes de transmission P2P .

5.3 L'extensibilité de la redistribution dans le P2PTV

Dans cette section, nous étudions un réseau recouvrant P2PTV qui transmet une seule chaîne et nous définissons k comme la variable qui représente le flux qui sera redistribué. La variable k que nous appelons variable de redistribution peut varier de 0 jusqu'à l'infini et il peut être aussi une variable fractionnée. Quand k égal à 0, cela veut dire que le client ne redistribue rien. Et quand k est égal à 1, cela veut dire que le client redistribue tout le flux. Lorsque k est égal à 2, cela signifie que le client redistribue deux copies du flux et lorsque k est égal à 0.5, cela signifie que le client redistribue seulement la moitié du flux à cause de la limitation imposée par la bande passante. Ici, les valeurs fractionnées sont possibles car le flux peut être divisé en tranches / couches. Cela permet aux clients de redistribuer seulement une fraction du flux qui peut être composée de plusieurs tranches. Le nombre total des tranches par flux définit la granularité selon laquelle le flux peut être divisé. Dans Zattoo [Zat], le flux est divisé en 16 tranches.

5.3.1 Analyse théorique

La figure 5.1 représente la topologie d'un système P2PTV. Dans cette figure, la capacité de la source C est égale à 3 (i.e., le client peut se connecter directement à la source) et le facteur de la redistribution k est égal à 2. Comme le réseau recouvrant est un graphe acyclique et direct (DAG) "Directed Acyclic Graph", nous pouvons définir la profondeur d'un client comme la valeur qui correspond au nombre des liens entre le client et la source à laquelle il est connecté. En supposant que tous les clients ont la même valeur k, nous pouvons déterminer le nombre maximal de pairs en utilisant le facteur de la redistribution k, la profondeur du graphe n et la capacité de la source C comme le nombre total des clients/pairs égale à la somme d'une série géométrique. En effet, si U_n représente le nombre des pairs à une profondeur n nous aurons :

$$U_0 = C \tag{5.1}$$

$$U_n = k \times U_{n-1} \tag{5.2}$$

Le nombre total des clients sera ainsi :

$$S_n = \sum_{i=0}^{n} U_i = C \times \frac{1 - k^{n+1}}{1 - k} \tag{5.3}$$

Le nombre total des pairs qui sont capables de se connecter et ainsi l'extensibilité du système dépendra de la valeur k :

– Si $k < 1$ le nombre total maximal des pairs sera $\sum U_n \rightarrow \frac{C}{1-k}$ et ainsi le système ne passera pas à l'échelle.
– Si $k = 1$ le nombre total maximal des pairs va diverger linéairement, $\sum U_n \rightarrow +\infty$ et ainsi le système passera à l'échelle en fonction de sa taille.
– Si $k > 1$ le nombre total maximal des pairs va diverger exponentiellement, $\sum U_n \rightarrow +\infty$ et ainsi le système passera à l'échelle en fonction de sa taille.

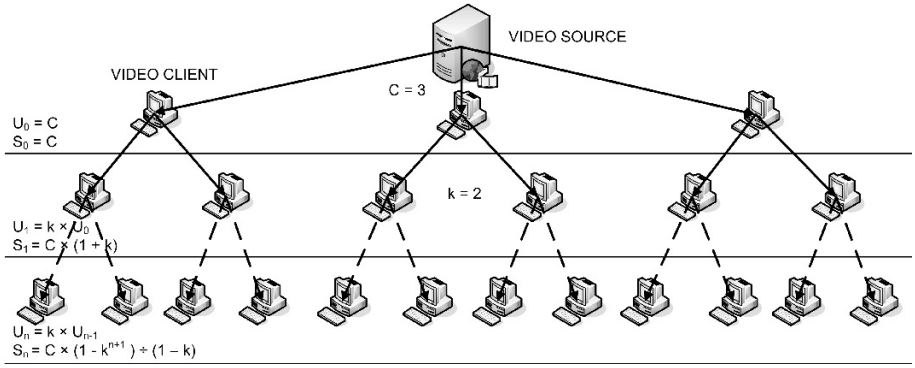

FIG. 5.1 – Réseau recouvrant de distribution P2PTV.

Maintenant, prenons le cas général où pour un niveau l, un client i a un taux/une capacité de téléversement (ul_i) et de téléchargement (dl_i), le facteur k pour ce niveau l qui contient p clients sera :

$$k_l = \frac{\sum_{i=1}^{p} ul_i}{\sum_{i=1}^{p} dl_i} \tag{5.4}$$

où :

D_{act} est le nombre des tranches téléchargées par chaque client actif
U_{act} est le nombre des tranches téléversées par chaque client actif
D_{idl} est le nombre des tranches téléchargées par chaque client inactif
U_{idl} est le nombre des tranches téléversées par chaque client inactif
U_{hyp} est le nombre des tranches téléversées par chaque client hyperactif
R_{idl} est la proportion des clients inactifs *vs* le nombre total des clients
R_{hyp} est la proportion des clients hyperactifs *vs* le nombre total des clients

Nous obtiendrons ainsi l'équation suivante :

$$k = \frac{U_{act} \times (1 - R_{idl} - R_{hyp}) + U_{idl} \times R_{idl} + U_{hyp} \times R_{hyp}}{D_{act} \times (1 - R_{idl} - R_{hyp}) + D_{idl} \times R_{idl} + D_{act} \times R_{hyp}} \tag{5.5}$$

Dans le meilleur des cas nous pourrions avoir $k \geq 1$ pour tous les clients et ainsi il n'y aura pas de limitation concernant la taille du réseau recouvrant (malgré les problèmes

liés à la profondeur du réseau recouvrant quand k n'est pas assez grand).

Pratiquement, par rapport aux problèmes d'extensibilité nous avons :
- k peut être inférieur à 1 :
  - *Freeriders* ($k = 0$)
  - Pairs bloqués derrière un NAT et/ou un pare-feu ($k = 0$)
  - Hypo pairs, i.e., les pairs qui ont une faible capacité de téléversement ($k < 1$)
- k peut être supérieur à 1 :
  - Serveur subsidiaire, i.e., répéteur ($k >> 1$)
  - Hyperpairs, i.e., les pairs qui ont une grande capacité de téléversement ($k > 1$)
  - Pairs inactifs, i.e., les paris qui ne regardent pas le flux mais ils redistribuent un sous ensemble de tranches du flux ($k > 1$ pour quelques tranches)

Le tableau 5.2 montre le nombre total maximal des pairs dans un réseau P2PTV quand la localisation des pairs est optimale (chaque niveau est rempli avant de continuer vers le niveau prochain). Nous supposons ici que la capacité de la source est de 100 (fournie par la source elle-même ou par les répéteurs). Dans ce cas, le nombre total est calculé pour plusieurs valeurs de la profondeur n et en fonction du facteur de la redistribution k, et la convergence peut être alors vu à une profondeur 10 et $k < 0.75$.

$n \downarrow$	2	1.10	1	0.90	0.75	0.50	0.33	0.25
0	100	100	100	100	100	100	100	100
1	300	210	200	190	175	150	133	125
2	700	331	300	271	231	175	144	131
3	1500	464	400	344	273	188	148	133
4	3100	611	500	410	305	194	149	133
5	6300	772	600	469	329	197	150	133
6	12700	949	700	522	347	198	150	133
7	25500	1144	800	570	360	199	150	133
8	51100	1358	900	613	370	200	150	133
9	102300	1594	1000	651	377	200	150	133
10	204700	1853	1100	686	383	200	150	133

FIG. 5.2 – Taille du réseau recouvrant pair à pair *vs* facteur de redistribution k.

5.3.2 Les pairs inactifs

Afin de faire face au problème de la faible valeur du facteur k, une solution propose d'utiliser les pairs inactifs pour distribuer une partie du flux. Un pair qui ne regarde pas le flux peut recevoir une fraction de ce flux (quelques tranches qui ont besoin d'une faible

bande passante) et redistribuer des copies de cette fraction (en jouant le rôle d'un multiplicateur).

La figure 5.3 montre la capacité du réseau P2PTV en fonction du pourcentage de ses pairs inactifs tout en supposant que les clients actifs ont $k = 0.5$ et que les clients inactifs redistribuent 1/16ème du flux quatre fois (donc $k = 4$ pour 1 tranche). Nous supposons aussi que les pairs inactifs multiplient différentes tranches afin d'optimiser leur efficacité. Les lignes dans la figure 5.3 sont obtenues à partir de l'équation 5.5. Ici, la capacité du réseau est égale au nombre total des pairs divisé par la capacité de la source (répéteurs inclus quand c'est nécessaire). Donc, quand la capacité du réseau est égale à 1, cela veut dire qu'il n'y a pas de relation entre les pairs et que les pairs peuvent se connecter seulement à la source. Lorsque la capacité du réseau égale à 10 cela veut dire que le nombre total des pairs peut atteindre dix fois la capacité de la source. Dans ce cas, si la capacité de la source est de 50 et que la capacité du réseau est de 10, le nombre total des pairs peut atteindre 500. Le schéma montre les capacités des réseaux actif et inactif. Il indique que les pairs inactifs n'aident pas vraiment à augmenter la capacité du réseau actif et cela quand la valeur réelle des pairs inactifs est de <30%. Cependant, les pairs actifs dans le réseau recouvrant ont besoin d'un grand nombre des pairs inactifs pour s'étendre (i.e., pour avoir une capacité >10). Par conséquent, l'utilisation des paris inactifs n'aide pas à rendre le système extensible.

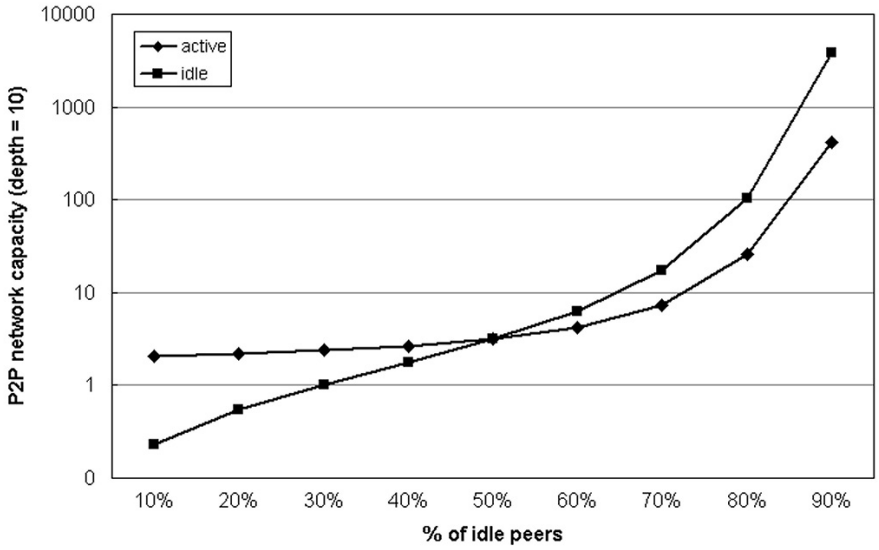

FIG. 5.3 – Capacité du réseau *vs* nombre de pairs inactifs.

5.3.3 Les Hyperpairs

Afin de faire face au problème de la faible valeur du facteur k, une autre solution consiste à utiliser les pairs hyperactifs pour redistribuer plusieurs flux. On peut définir un pair hyperactif comme le pair qui reçoit le flux et redistribue plusieurs copies du flux ou au moins une seule copie de ce flux ainsi que plusieurs tranches supplémentaires de ce flux.

La figure 5.4 montre la capacité du réseau P2PTV en fonction du pourcentage des pairs hyperactifs. On suppose ici que les clients réguliers et actifs ont $k = 0.5$ et les clients hyperactifs ont $k = 1.5$. Nous avons supposé aussi que les membres actifs multiplient des différentes tranches afin d'optimiser leur efficacité. Les lignes dans ce schéma sont obtenues en utilisant l'équation 5.5. Nous pouvons remarquer aussi qu'il y a une divergence de 60% des pairs hyperactifs. Quand il y a moins des pairs hyperactifs, ils doivent avoir des valeurs de k plus élevées car ils vont redistribuer plusieurs copies du flux et cela afin d'assurer une meilleure performance dans le réseau.

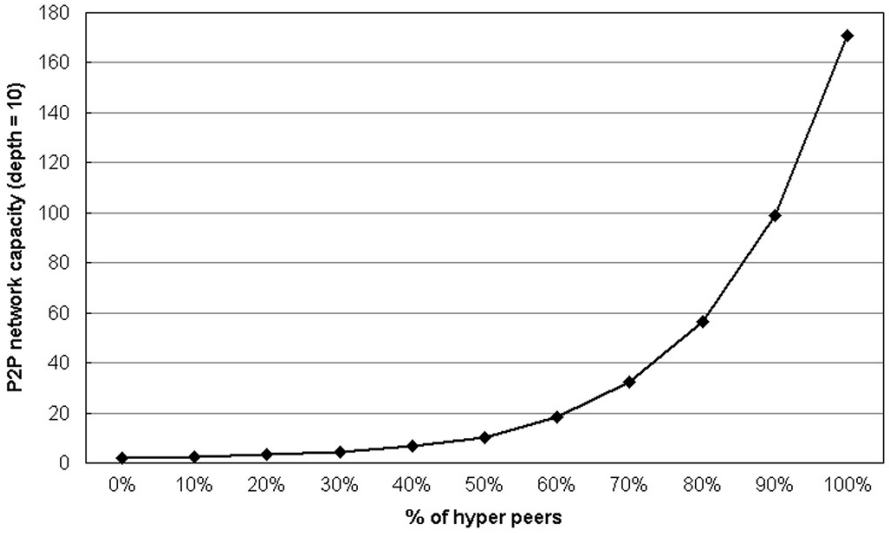

FIG. 5.4 – Capacité du réseau *vs* nombre de pairs hyperactifs.

5.4 Les caractéristiques des pairs dans un réseau P2PTV

Dans cette section nous allons présenter quelques résultats de simulations et de tests que nous avons effectués sur des données réelles afin d'étudier les caractéristiques typiques d'un réseau P2PTV et ceux des pairs qui appartiennent à ce réseau. Nous essayons ici d'identifier les distributions qui concernent la capacité de téléversement (le facteur k), la sécurité du NAT et la durée des sessions.

5.4.1 La collecte des données

Afin de savoir si les systèmes actuels sont extensibles et de pouvoir les simuler, nous avons collecté quelques données d'un système P2PTV commercial. Les données présentées ici sont généreusement fournies par la compagnie Zattoo qui travaille sur un réseau P2PTV étendu sur le continent. Le logiciel Zattoo est un logiciel gratuit mais c'est un système P2PTV possédé par l'entreprise. Chaque fois un client/pair s'y connecte, il regarde et se déconnecte d'une chaîne TV (i.e., cela nous l'appelons une session) fournie par Zattoo [Zat]. L'entrée de la session est enregistrée dans une base de données centralisée (appelée : la base des données de la session). Cette entrée contient les éléments suivants :

– Le temps de la connexion
– Le temps de la déconnexion
– Le statut des NAT et des pare-feux
– Le taux du téléchargement approximatif
– Le taux du téléversement approximatif
– Le nombre total des octets téléchargés
– Le nombre total des octets téléversés
– L'IP publique et le numéro d'AS (système autonome) du client
– Le code du pays du client k
– Le nom de la chaîne TV regardée
– La durée de la session

Les données sont collectées à partir des informations concernant une session enregistrée et stockée dans la base des données de la session et cela pendant deux semaines. Le nombre des sessions collectées est 9,8 millions de sessions et cela pour 198 chaînes et 14 pays.

5.4.2 Les caractéristiques

Nous allons présenter maintenant quelques résultats liés aux caractéristiques qui sont indépendantes du système P2PTV. Ces caractéristiques sont liées à la configuration et à la performance de la machine sur laquelle l'application P2PTV est utilisée.

5.4.2.1 La capacité de téléversement

La figure 5.5 montre la distribution du pourcentage k qui représente le taux du téléversement divisé par le taux du téléchargement (i.e., ce pourcentage est une évaluation du facteur potentiel de la distribution k). Ces taux ont été mesuré par le client et un serveur spécifique afin de déterminer les possibilités de la bande passante du client. Comme la mesure de la bande passante est difficile à faire et dépend de plusieurs facteurs, les valeurs présentées ici peuvent ne pas être tout à fait correctes. La fonction cumulative de la distribution (CDF) "Cumulaitve Distribution Fonction" est représentée par une ligne logarithmique. Ainsi, nous pouvons remarquer que la distribution est très décentrée et la valeur moyenne de k calculée à partir de toutes les valeurs de la distribution est égale à 0.89 et elle reste au-dessous de 1 (sa valeur n'est pas significative car la distribution n'est pas normale). Nous remarquons aussi que 50% des clients/pairs peuvent redistribuer moins de 50% du flux entier (i.e., ils ont $k < 0.5$) et 82% des clients/pairs peuvent redistribuer moins d'un flux entier (i.e., ils ont $k < 1$). Donc, nous pouvons conclure que le réseau P2PTV basé sur ces valeurs de redistribution ne peut pas être extensible.

5.4.2.2 La compatibilité NAT

Tous les clients ont une configuration NAT et une configuration pare-feu qui peut bloquer dans certains cas les communications entre deux clients/pairs. Chaque configuration différente est attribuée à un chiffre qui identifie le type du NAT (e.g., 1 est un hôte situé

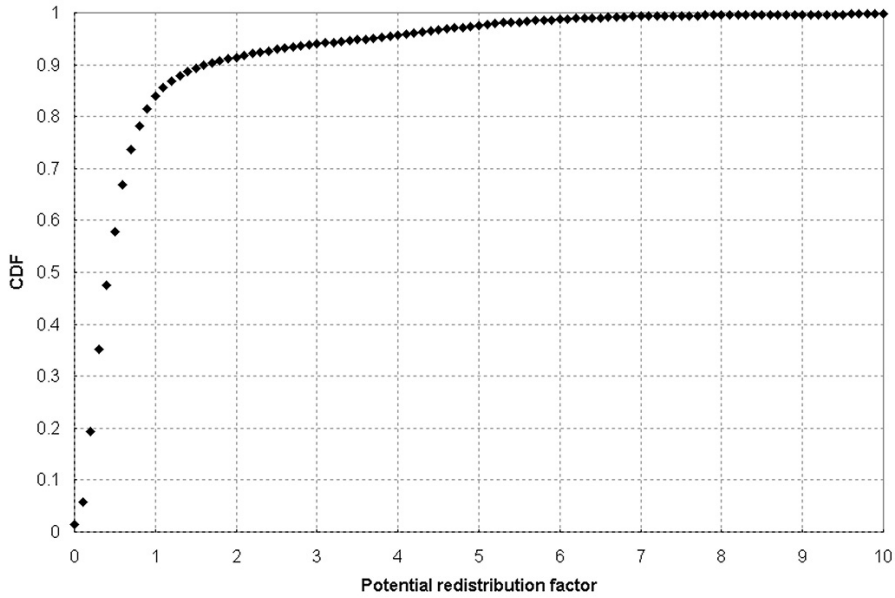

FIG. 5.5 – Facteur de redistribution potentiel.

sur Internet). Toutes les autres valeurs signifient que le client est situé derrière un routeur NAT. Par conséquent, tous les messages UDP reçus par le NAT seront rejetés sauf s'il y a un port qui réfère à une destination derrière le NAT. Ce genre de correspondance est créée dès qu'un client situé derrière un routeur NAT envoie un message UDP hors bande, et il persiste pour un certain temps - ce temps est réinitialisé quand le client envoie un message UDP hors bande. On a donc besoin de messages de maintien de connexion STUN afin de garder le port de correspondance ouvert. Une fois que le client situé derrière un routeur NAT a envoyé le message UDP et que la boîte NAT a créée le port de correspondance, l'accessibilité aux pairs externes dépend de la nature restrictive du NAT. Les autres types de NAT représentent les différents niveaux de restriction sur les boîtes NAT commençant par le moins restrictif jusqu'au plus restrictif. Chaque type de NAT supérieur à 5 signifie que le client est traité comme un UDP invalide. Les 6 types définis actuellement sont les suivants :

- Type 1 : hôte ouvert - ce client n'est pas derrière un NAT.
- Type 2 : Cône NAT - une fois qu'un port de correspondance est ouvert, il sera ouvert pour tout le monde - tous les messages envoyés de n'importe quelle adresse IP sur n'importe quel port seront transférés par la boîte NAT. A part l'adresse IP externe et probablement quelques ports utilisés pour la correspondance, les cônes NATs peuvent être considérés comme de type 1.
- Type 3 : NAT restreint aux adresses IP - une fois qu'un port de correspondance est créé, il sera ouvert seulement aux adresses IP auxquelles le client a envoyé les messages UDP. Tous les messages de n'importe quelle adresse IP vers laquelle le client a envoyé un message UDP (sur n'importe quel port) seront transférés. D'un autre côté, les messages envoyés à partir d'autres adresses IP (sur n'importe quel port) seront rejetés.
- Type 4 : NAT restreint aux ports - une fois qu'un port de correspondance est créé, il sera ouvert seulement à certaines combinaisons adresse IP/port et auxquelles le

client a envoyé des messages UDP. Les messages qui ne viennent pas d'une combinaison adresse IP/port vers laquelle le client a envoyé explicitement un message UDP seront rejetés. Cela pourrait être classifié comme *NAT restreint aux ports et aux adresses IP* mais par simplicité, nous avons préféré classer ce type sous *NAT restreint aux portes*.

– Type 5 : NAT symétrique - ce type ressemble au type précédant sauf que pour chaque combinaison adresse IP/port vers laquelle le client a envoyé un message UDP, la boîte NAT crée une correspondance de port différente. Donc, chaque adresse IP/port externe va échanger le trafic en utilisant un port différent dans la boîte NAT et cela afin de le transférer au client.
– Type 6 : UDP invalide - le client apparaît dans ce cas totalement incapable d'envoyer et de recevoir les messages UDP. Il fonctionne en connexion TCP.

La figure 5.6 montre la distribution des différents types NAT des clients. Nous pouvons remarquer que la plupart des clients ont un NAT de type 4 ou 5 mais quelques clients ont un NAT de type 1, 2 ou 6. Néanmoins, il existe une faible quantité de clients qui ont un NAT de type 3.

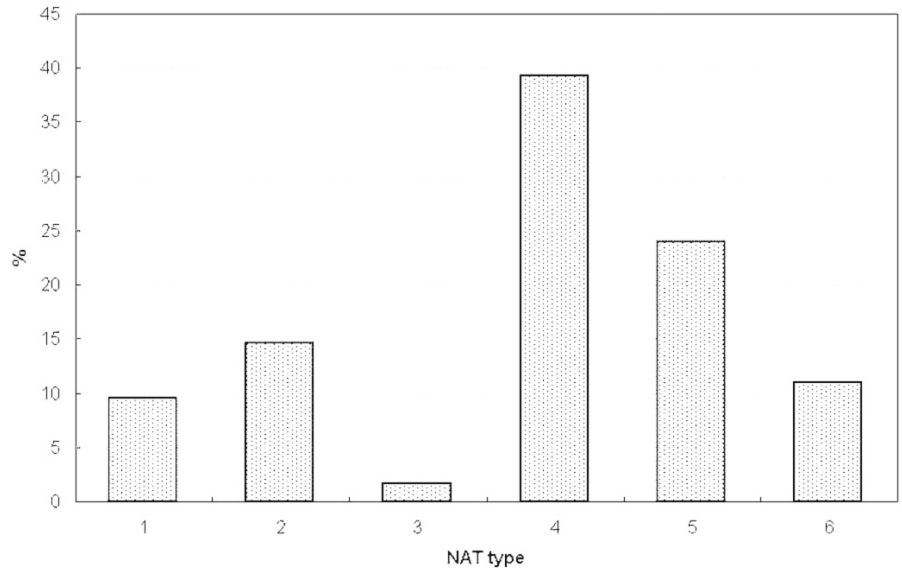

FIG. 5.6 – Types NAT.

La figure 5.7 montre la compatibilité des différents types de NAT. Si on regarde une ligne qui correspond à un seul type de NAT et une colonne qui correspond un autre type de NAT, nous pouvons remarquer que la matrice est symétrique, ce qui veut dire que ce n'est pas vraiment important si le client parent ou bien le client enfant sont pris d'une ligne ou bien d'une colonne. En fait, la compatibilité est ici représentée par les intersections. Donc, si la cellule est blanche, les clients se connectent en TCP, et si la cellule est grise, les clients se connectent en UDP. Mais si la cellule est noire, les clients ne peuvent pas se connecter entre eux.

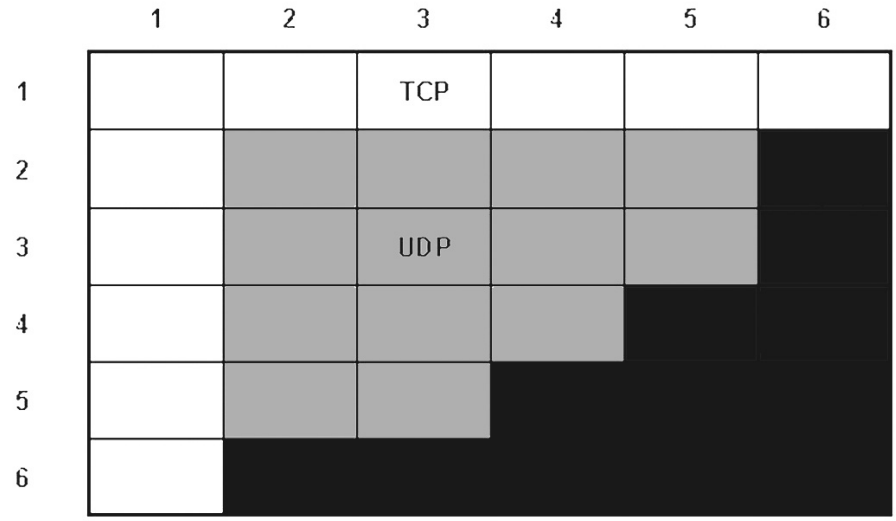

FIG. 5.7 – Compatibilité NAT.

Donc, en tenant compte de cette distribution et avec la matrice, pour n'importe quelle couche d'une profondeur n non directe au dessous de la source, nous pouvons remarquer que la probabilité de la connectivité des pairs dès la première tentative à une couche d'une profondeur n est de 57%. La probabilité moyenne de la connectivité maximale après plusieurs essais à une couche d'une profondeur n est environ de 90%.

Nous présentons maintenant des résultats qui concernent les caractéristiques qui dépendent du système P2PTV. Ces caractéristiques sont liées au comportement de l'utilisateur/client final qui utilise l'application P2PTV.

5.4.2.3 La durée des sessions

La figure 5.8 montre la distribution des durées de session. Comme nous l'avons remarqué précédemment, sa CDF "Cumulaitve Distribution Fonction" correspond à une courbe logarithmique.

5.5 Simulations sur un réseau P2PTV

Après avoir étudié les différents aspects théoriques, les problèmes liés à la capacité de téléversement et au NAT d'un système P2PTV et pris en compte les caractéristiques des pairs dans un système P2PTV réel, nous allons maintenant présenter des résultats obtenus par des simulations que nous avons effectuées dans le but de souligner les différents problèmes d'un système P2PTV typique et en particulier le problème du NAT qui peut être résolu par l'utilisation de notre intergiciel DHARMA [SMW+09].

5.5.1 Les paramètres de simulation

Le code de simulation d'un réseau P2PTV typique a été écrit en C ++ et il a été inséré et utilisé dans un logiciel de manipulation de réseau (NEM) "network manipula-

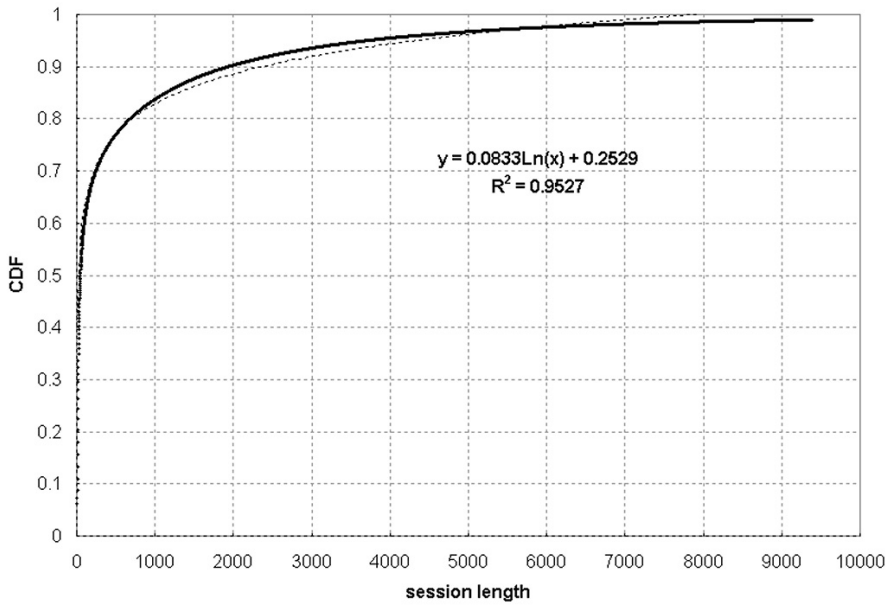

FIG. 5.8 – Durée des sessions en secondes.

tor" [Mag] afin d'effectuer les différentes simulations. NEM contient un moteur à événements discrèts très similaire à ns-2. Cependant, NEM n'intervient pas comme ns-2 dans les protocoles ce qui lui permet de faire des simulations sur des grandes topologies. La topologie que nous avons utilisée est représentée par une carte d'Internet de 4.2k noeuds. Nous considérons seulement une seule distribution d'un réseau recouvrant et une seule chaîne TV pour le moment. De plus, nous n'allons pas prendre en compte les variations quotidiennes de la taille de la chaîne. Chaque essai dure 12 heures mais nous analysons uniquement les dernière 6 heures (i.e., après la 6ème heure nous sommes dans un état stable) et le résultat final représente la moyenne de 30 essais où les valeurs des écart-types ont été fournies.

Nous utilisons les résultats présentés dans la section précédente afin de configurer les paramètres d'entrée suivants : la durée de la session, le type de NAT et la capacité de téléversement (le facteur k). Nous avons choisi ces valeurs aléatoirement afin de suivre les distributions mesurées dans la section précédente. Enfin, nous faisons varier le nombre de nouveaux pairs essayant de se connecter par heure. La table 5.9 montre les autres paramètres.

Pour évaluer la performance du système P2PTV, nous avons choisi d'étudier les variables de sortie suivantes :

1. Le temps de la visualisation (en pourcentage égale 100 fois le temps de la visualisation par le pair divisé par le temps qui représente la durée de la vie du pair) : le temps de la visualisation des pairs terminés pendant cette période.
2. Le nombre des pairs rejetés : le nombre total de pairs qui n'ont pas pu se connecter au réseau recouvrant P2PTV pendant une période prédéterminée.

Paramètres	Valeurs
Granularité (nb de tranches)	16
Capacité de la source	50 clients
Durée de recherche	5 s
Nb maxi de recherches	10
Durée de scan	0.5 s
Taille maxi de la liste des pairs	40 pairs
Nb d'instances par scénario	30

FIG. 5.9 – Paramètres de simulation.

5.5.2 Résultats des simulations

Nous allons maintenant étudier l'effet de la compatibilité du NAT et la capacité de téléversement et comment on peut les améliorer en utilisant notre architecture DHARMA à travers le temps moyen de la visualisation quand le facteur k est égal à 1 pour tous les clients. La figure 5.10 montre le temps moyen de la visualisation en fonction du nombre des clients démarrés par heure. Nous pouvons remarquer que quand le nombre de clients est petit, tout le monde peut se connecter directement à la source. Dans ce cas, le pourcentage de visualisation est proche de 100%. Mais quand le nombre des clients augmente, ils doivent se connecter entre eux afin de créer un réseau P2P. A cause des problèmes de compatibilité du NAT expliqués précédemment, le pourcentage diminue progressivement jusqu'à 87%. Cette différence entre un réseau recouvrant avec des clients normaux et un réseau recouvrant avec des clients DHARMA peut atteindre progressivement jusqu'à 10 % quand le nombre de clients est élevé. En utilisant DHARMA, tous les clients peuvent se connecter entre eux et par conséquent le pourcentage de la visualisation ne change pas tout en restant proche de 100%. Comme chaque client redistribue un flux complet, il n'y a pas de limite à cause de la capacité de téléversement et pour cela le temps de la visualisation est presque de 100%.

Nous allons maintenant regarder le nombre moyen des rejets quand le facteur k est égal à 1 pour tous les clients. La figure 5.11 montre le nombre moyen des clients qui n'arrivent pas à se connecter au réseau P2P en fonction nombre des clients démarrés par heure quand $k = 1$. Dans ce cas, quand le nombre de clients est petit, tout le monde peut se connecter directement à la source et le nombre des rejets est proche de 0. Mais quand le nombre des clients augmente, le nombre des rejets augmente progressivement jusqu'à atteindre environ 60 par heure. Cette lacune entre un réseau recouvrant avec des clients normaux et un réseau recouvrant avec des clients DHARMA peut atteindre progressivement jusqu'à 60 clients rejetés par heure quand le nombre des clients est élevé. En utilisant DHARMA, tous les clients peuvent se connecter entre eux et par conséquent le nombre des rejets reste proche de 0. Comme chaque client redistribue un flux complet, il n'y a pas de limite à cause de la capacité de téléversement et pour cela les rejets restent proches de 0.

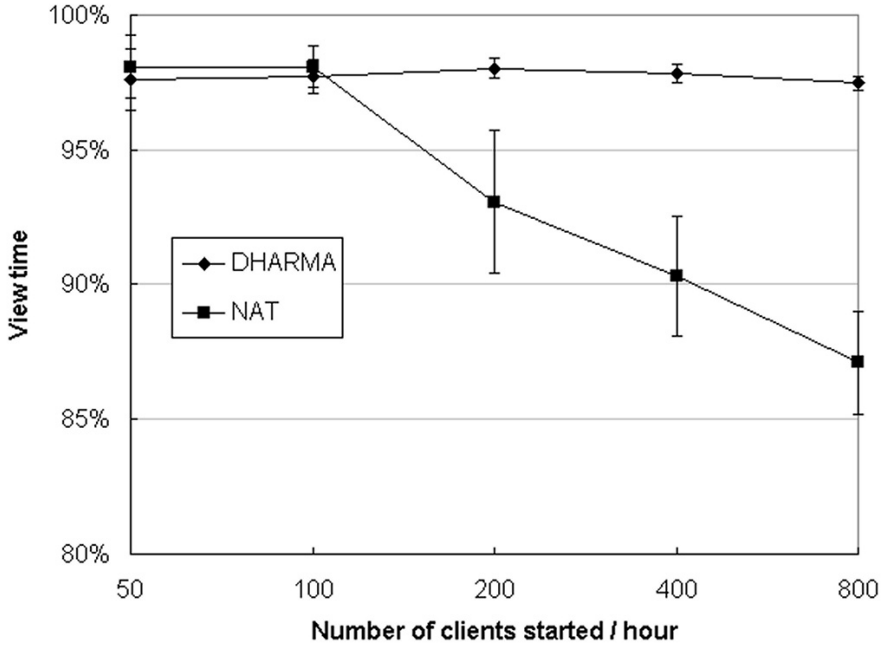

FIG. 5.10 – Durée de visionnage moyenne *vs* nombre de clients démarrés par heure lorsque k = 1.

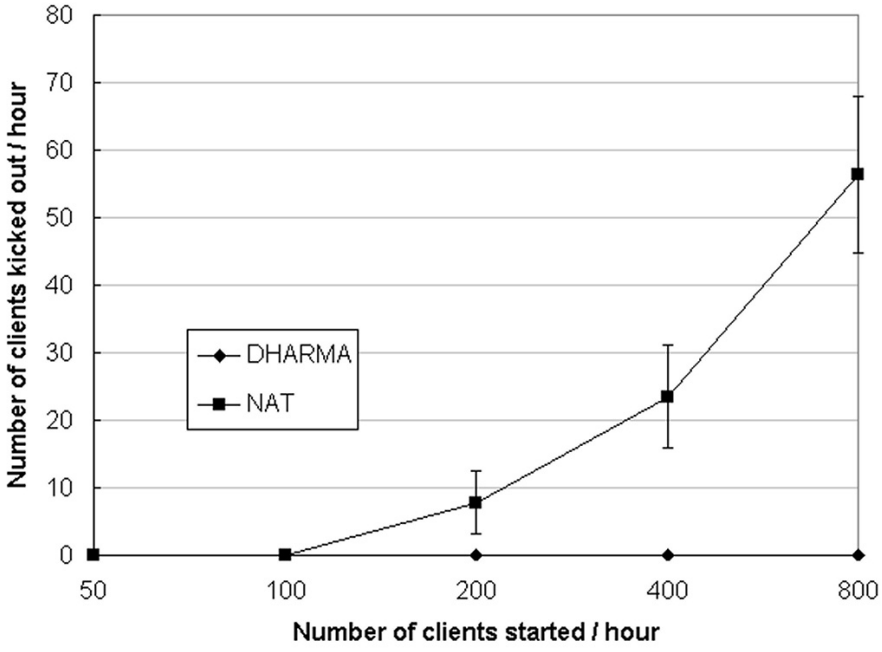

FIG. 5.11 – Nombre de clients éjectés *vs* nombre de clients démarrés par heure lorsque k = 1.

Ensuite, nous allons étudier le temps moyen de la visualisation quand le facteur k est basé sur une distribution dans le monde réel comme celle qui est expliquée dans la sec-

tion précédente. La figure 5.12 montre le temps moyen de la visualisation par le client en fonction du nombre de clients démarrés par heure et cela quand le facteur k de tous les clients suit une distribution similaire à celle présentée dans la section précédente. Comme nous avons expliqué précédemment, quand le nombre des clients est petit, tout le monde peut se connecter directement à la source et le pourcentage de la visualisation est proche de 100%. Mais quand le nombre des clients augmente, ils doivent se connecter entre eux afin de créer un réseau P2P, et à cause des problèmes de compatibilité du NAT et les limitations provoquées par le facteur k (expliqués précédemment), le temps de visualisation diminue progressivement jusqu'à 30%. Cette différence entre un réseau recouvrant avec des clients normaux et un réseau recouvrant avec des clients DHARMA dépend du nombre des clients démarrés. Elle commence à partir de 0% où il y a 100 clients démarrés par heure ou moins, et puis elle augmente jusqu'à 10% quand le nombre des clients est de 200 ou plus. Cette lacune entre l'utilisation de DHARMA ou pas reste autour de 10% ce qui est presque l'équivalant de la valeur définie dans la section précédente. Car en utilisant DHARMA, tous les clients peuvent se connecter entre eux et pourtant, malgré le fait que le pourcentage de la visualisation diminue à cause du facteur réel de la distribution k, il reste toujours autour de 10% au-dessus de la valeur des clients dans un réseau recouvrant qui ne l'utilise pas. Comme chaque client redistribue seulement une partie du flux, il y a toujours une limitation à cause de la capacité de téléversement et pour cela le temps de la visualisation diminue jusqu'à un peu près de 40% quand le nombre des clients démarrés par heure est égal à 800.

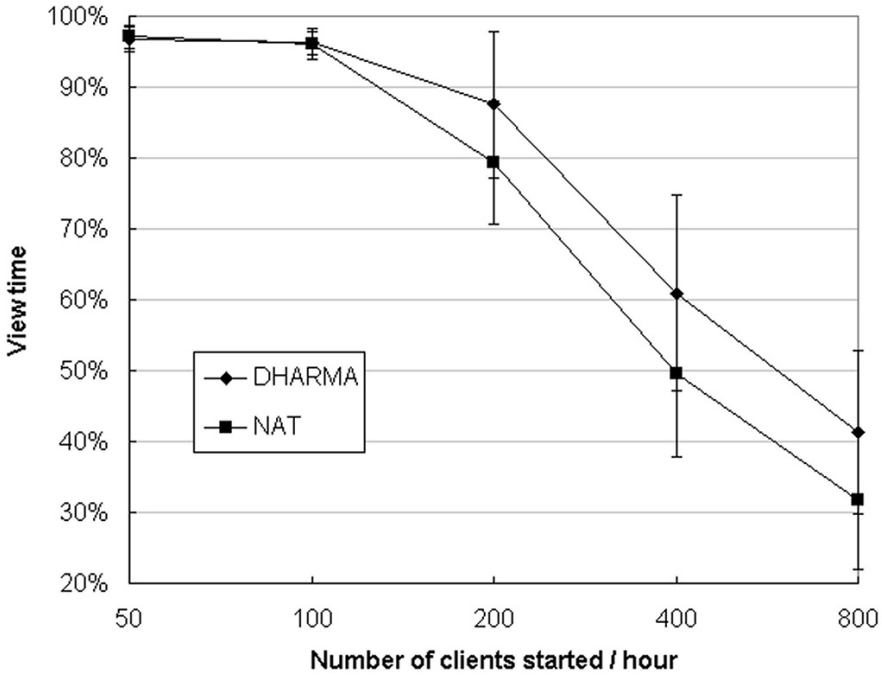

FIG. 5.12 – Durée de visionnage moyenne *vs* nombre de clients démarrés par heure lorsque k suit sa distribution réelle.

Enfin, nous allons analyser le nombre moyen des rejets quand le facteur k est basé

sur une distribution dans le monde réel (présentée précédemment). La figure 5.13 montre le nombre moyen des clients qui n'arrive pas à se connecter à un réseau P2P en fonction du nombre des clients démarrés par heure quand le facteur k de tous les clients suit une distribution similaire à celle présentée dans la section précédente. Donc quand le nombre des clients est égal ou inférieur à 100 clients démarrés par heure, tout le monde peut se connecter directement à la source et le nombre des rejets est proche de 0. Mais quand le nombre des clients augmente, le nombre des rejets augmente progressivement jusqu'à 400 par heure et cela quand on n'utilise pas DHARMA. Donc, la différence entre un réseau recouvrant avec des clients normaux et un réseau recouvrant avec des clients DHARMA augmente régulièrement quand le nombre des clients démarrés par heure augmente. En utilisant DHARMA, tous les clients peuvent se connecter et pourtant, malgré l'augmentation du nombre de rejets à cause du facteur réel de la distribution k, il reste toujours autour de 10% au-dessous du nombre de rejets des clients dans un réseau recouvrant qui ne l'utilise pas. Comme chaque client redistribue seulement une partie du flux, il y a toujours une limitation à cause de la capacité de téléversement et pour cela le nombre des rejets augmente jusqu'à 300 quand le nombre des clients démarrés par heure est égal à 800.

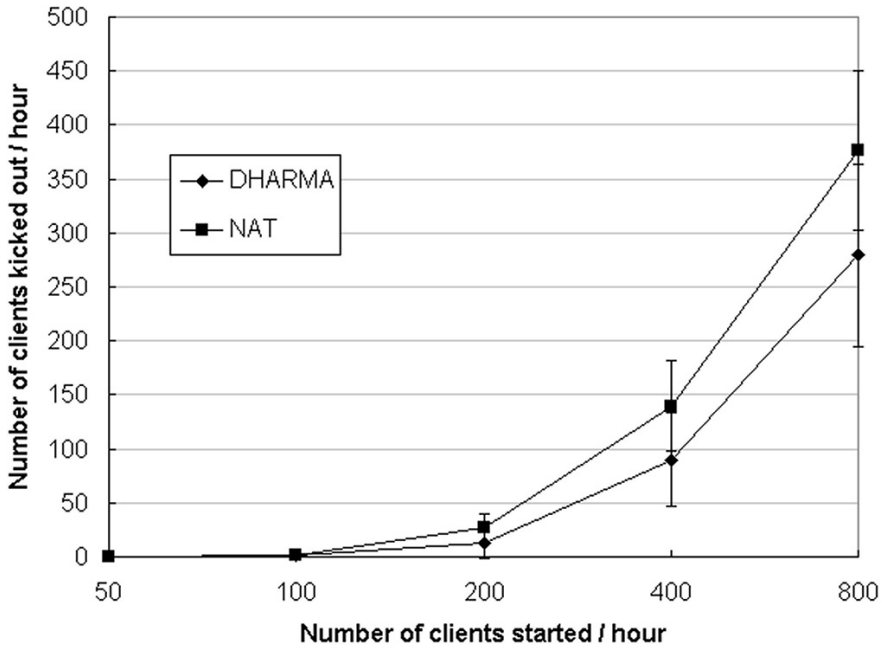

FIG. 5.13 – Nombre de clients éjectés *vs* nombre de clients démarrés par heure lorsque k suit sa distribution réelle.

Tous ces résultats permettent de constater que dans un réseau recouvrant P2PTV, le problème de la compatibilité du NAT crée une perte moyenne de 10% des clients. Cela veut dire que dans une couche prédéterminée et à une profondeur n dans l'arbre de la distribution, toutes les connexions possibles seront établies avec 90% par les enfants (pairs) qui sont déjà connectés. Autrement dit, l'utilisation de DHARMA comme réseau recouvrant sous-jacent permet de surmonter le problème de la compatibilité du NAT et les résultats des simulations prouvent que la perte est presque nulle, ce qui montre ainsi l'ef-

ficacité de DHARMA. De plus, quand les facteurs k des clients sont proches de 1, l'utilisation de DHARMA peut rendre le système extensible, alors que si on ne l'utilise pas, cela va rendre la taille du système jusqu'à une valeur maximale égale à 10 fois la capacité de la source. En effet, comme nous l'avons montré précédemment, la perte de 10% des clients si chacun a $k = 1$ est équivalante à 10% des parents qui ne redistribuent rien, et par conséquent, la valeur moyenne de k va diminuer jusqu'à 0.9 menant à une convergence de la taille du système jusqu'à 10 fois la capacité de la source. Quand la valeur moyenne du facteur k des clients est suffisante pour compenser cette compatibilité du NAT (la perte de 10% inclus), dans ce cas, quand on utilise DHARMA cela va seulement augmenter la performance du système de 10% mais sa taille restera capable d'augmenter sans limite.

5.6 Conclusion

Nous avons présenté dans ce chapitre une application P2PTV qui utilise notre architecture DHARMA afin d'améliorer la performance des transmissions des flux multimédia en surmontant les obstacles rencontrés par le NAT.

Dans ce travail, nous avons au préalable effectué une étude importante de l'extensibilité en se basant sur quelques caractéristiques fondamentales des réseaux P2PTV, à savoir, les capacités de téléversement vers les machines distantes, les durées des sessions et les types NAT. Afin de faire face à la faible valeur du facteur de distribution k, nous avons pensé à deux solutions : la première solution consiste a utiliser les pairs inactifs pour distribuer une partie du flux. Cependant, nous avons constaté que cette solution n'aide pas vraiment à augmenter la capacité du réseau actif quand la valeur réelle des pairs inactifs est inférieure à 30%. La deuxième solution consiste à utiliser les pairs hyperactifs pour redistribuer plusieurs copies du flux ou au moins une seule copie du flux et plusieurs tranches supplémentaires de ce flux.

En étudiant l'effet de la compatibilité du NAT et la capacité de téléversement, nous avons montré la différence entre un réseau recouvrant avec des clients normaux et un réseau recouvrant avec des clients DHARMA. Nous avons observé que les types NAT entrainent une baisse des performances d'environ 10%. Cette baisse peut être totalement compensée par l'utilisation de DHARMA. En effet, nous avons montré que l'utilisation de DHARMA peut faire augmenter k. Cette étude montre que l'utilisation de DHARMA peut rendre le système extensible tout en améliorant sa performance.

Chapitre 6

Conclusion

Ce chapitre conclut cette thèse, il récapitule nos principales contributions et il présente quelques directions pour les futurs travaux qui permettraient d'étendre de manière substantielle la portée de nos travaux.

6.1 Contributions

Les travaux de cette thèse ont commencé avec une étude théorique et approfondie présentant les réseaux recouvrants. Les réseaux recouvrants qu'on peut définir tout simplement comme des réseaux virtuels qui se situent au-dessus d'une infrastructure réseau déjà existante, ont pour but d'offrir de nouveaux services, ce qui était difficile à implémenter dans l'infrastructure réseau actuelle, et permettent d'améliorer la performance des services existants. Les caractéristiques principales de ces réseaux sont :

- Le recouvrement des données garanti,
- L'équilibrage de charge automatique,
- L'auto-organisation.

Ensuite, nous avons présenté les deux principaux types de ces réseaux : les réseaux structurels et non structurels. De nombreux travaux ont été réalisé afin d'étudier et d'améliorer ces réseaux. Nous avons vu que les réseaux recouvrants non structurés sont simples à construire et à maintenir, mais malgré leur popularité, leur performance est incertaine et aléatoire. Contrairement à ces derniers, les réseaux recouvrants structurés sont construits en se basant sur une structure particulière qui permet d'atteindre efficacement un objectif. Cependant, les réseaux recouvrants structurés demandent une construction et une maintenance nettement plus complexe. Enfin, nous avons présenté quelques fonctionnalités des réseaux recouvrants telles que leur structure en couches, la méthode de sélection de voisinage, le placement du voisin, les communications entre les noeuds, la localisation des réseaux recouvrants et les étapes à suivre pour construire un réseau recouvrant. Ce chapitre ayant pour objectif de présenter les bases de l'environnement dans lequel cette thèse a évolué.

Le chapitre suivant a abordé le domaine des réseaux P2P en présentant tout d'abord et d'une manière simple l'évolution de ces réseaux. Cette évolution a débuté par l'apparition des réseaux comme USENET et FidoNET dont l'architecture était simple et centralisée. Ensuite, d'autres types de réseaux sont apparus dont le but est de partager des fichiers (e.g. Napster, gnutella, etc). Avec l'arrivée de nouvelles technologies, de nouveaux réseaux P2P modernes ont connu le jour, et une nouvelle génération de logiciels technologiquement plus avancée est apparue, représentée par BitTorrent, Overnet et Grabit, qui optimisent au maximum la bande passante grâce à l'envoi et à la réception en offrant un débit maximal en flux continu. Le reste de ce chapitre a été consacré à la présentation brève des applications les plus représentatives du domaine des réseaux recouvrants, tout en analysant les différentes architectures permettant la prise en charge de ces applications.

Dans la suite, nous avons consacré le quatrième chapitre à décrire notre architecture DHARMA (Dynamic Hierarchical Addressing, Routing and naMing Architecture). Une analyse de l'existant et les différentes étapes de conception, nous ont conduit à proposer une architecture logicielle basée sur un modèle qui offre plusieurs fonctionnalités au niveau applicatif : adressage, routage, nommage et pilotage.

Cette architecture est basée sur un système hiérarchique et distribué qui permet de séparer l'espace d'adressage de celui du nommage tout en fournissant un plan de convergence pour l'environnement hétérogène actuel d'Internet. Donc, notre but a été de concevoir un intergiciel d'adressage, de routage et de nommage robuste et efficace pour déployer et maintenir des réseaux recouvrants sur Internet.

Le routage dans notre architecture ressemble au routage employé actuellement sur Internet mais ici notre mécanisme est piloté par les adresses, ce qui veut dire que le réseau recouvrant utilise seulement ses propres adresses où les adresses obtenues par chaque noeud dépendent uniquement de ses voisins (degrés du noeud) et ainsi de sa localisation. Pour faire face à la dynamique du réseau (déplacement des noeuds, pannes), nous avons proposé deux solutions complémentaires : nous considérons d'une part que les adresses sont toujours valides temporairement, donc les noeuds peuvent récupérer leurs adresses après interruption ou bien ils peuvent obtenir de nouvelles adresses. D'autre part, nous considérons que les noeuds utilisent plusieurs adresses fournies par des voisins différents, donc ils seront toujours joignables par des chemins alternatifs. Nous avons montré aussi que la longueur de chemin dépend de l'efficacité de l'attribution des adresses car une partie de l'adresse de la destination correspond au chemin vers cette destination.

De plus, dans notre système, chaque noeud a un nom unique qu'il gardera pendant toute sa durée de vie dans le réseau recouvrant. Afin de rendre tout changement des adresses des noeuds transparent, nous avons fait en sorte que les applications qui emploient notre intergiciel puissent établir des connexions en utilisant les noms et pas les adresses. Ce mécanisme est contrôlé par des noeuds appelés serveurs de noms. En effet, la couche de nommage dans notre architecture peut garantir la propriété de transparence de bout-en-bout sur Internet qui n'existe pas actuellement à cause de l'attribution dynamique des adresses, des pare-feux, des NATs, etc.

Nous avons montré aussi que la hiérarchie des serveurs de noms est complètement indépendante de la hiérarchie d'adressage. De plus, l'extensibilité du stockage dans l'espace de nommage est assurée par une procédure basée sur le principe des tables de hachage distribuées. D'un autre côté, et afin d'assurer l'équilibrage de charge et pour faire face à la dynamique des serveurs de noms, nous avons employé une approche appelée "réplication" où chaque serveur de noms a $k-1$ copies identiques appelées répliques. Ces serveurs de noms sont tous opérationnels mais un seul parmi eux est choisi aléatoirement pour recevoir et envoyer les messages. Enfin, pour assurer l'extensibilité du recouvrement, les serveurs de noms sont organisés dans une hiérarchie arborescente. Par conséquent, notre intergiciel est extensible, résilient et autonome.

En fait, l'originalité dans notre architecture se matérialise surtout par la phase de pilotage qui consiste à lier les noms des noeuds à leurs adresses avant ou pendant la transmission des données entre les noeuds. Pendant cette phase, plusieurs noms logiques (des noms d'utilisateurs, de service ou de groupe) peuvent être liés aux noms des noeuds dans le réseau recouvrant avant ou pendant la transmission des données entre les noeuds. Enfin, nous avons présenté les différents avantages de notre intergiciel, citons :

- La convergence réseau,
- La mobilité,
- La sécurité,
- Le multipoint.

Dans la suite de ce chapitre nous avons examiné les caractéristiques de notre intergiciel par l'intermédiaire de divers scénarios de simulations. Les résultats de ces simulations étaient assez encourageants.

A travers nos simulations nous avons montré que le succès du routage s'améliore en augmentant le nombre d'adresses par noeud. De plus, nous avons prouvé qu'en utilisant notre algorithme d'adressage, le succès du routage est garanti (au-dessus de 95%) en utilisant suffisamment de périodes du temps (plus de 5) afin que le routage converge en cas de perturbations réseaux. Par ailleurs, avec 5 niveaux dans la hiérarchie de nommage, on peut traiter une très grande quantité de noms. Enfin, nous avons expliqué comment augmenter le succès de la résolution de nom même avec 50% de dynamique dans le réseau. Ce succès peut atteindre 100% en utilisant seulement 3 répliques pour chaque serveur de noms. Afin de prouver que notre intergiciel est extensible, nous avons effectué nos simulations sur des réseaux recouvrants ayant plus de 12000 noeuds.

La deuxième contribution apportée au sein de cette thèse s'est focalisée sur la conception d'une application P2PTV où nous avons profité des composants de l'architecture DHARMA pour améliorer une nouvelle application qui emploie la technique IPTV en se basant sur des réseaux pair-à-pair qui ont pour but d'assurer l'extensibilité des réseaux P2PTV en améliorant la redistribution des pairs. Le principal objectif a été de surmonter les problèmes rencontrés à cause de l'effet du NAT et cela afin d'améliorer la transmission des flux multimédia en live.

Après avoir étudié les différents aspects théoriques et les problèmes liés à la capacité de téléversement, au NAT d'un système P2PTV et après avoir étudié quelques caractéristiques des pairs dans un système P2PTV réel, nous avons présenté des résultats obtenus par des simulations que nous avons effectuées en utilisant des données réelles collectées à partir d'un système P2PTV commercial. Ces simulations ont été écrites en C++ et effectuées dans le simulateur *nem* en utilisant une topologie représentée par une carte d'Internet de 4200 noeuds.

Dans nos simulations, en étudiant la valeur du facteur de redistribution k, nous avons montré qu'en utilisant notre architecture DHARMA, tous les clients arrivent à se connecter (i.e. le nombre des clients rejetés s'approche de 0). Par conséquent, le temps de visionnage est presque de 100% dans le cas où nous supposons que chaque client redistribue un flux complet, donc il n'y a pas de limite à cause de la capacité de téléversement. Par contre, en étudiant le cas où chaque client redistribue seulement une partie du flux, il y a toujours une limitation à cause de la capacité de téléversement et alors le nombre des rejetés augmente jusqu'à peu près 40%. Cependant dans ce cas l'utilisation de DHARMA améliore de 10% le temps de visionnage.

En conclusion, nous avons montré que l'utilisation de DHARMA comme réseau recouvrant sous-jacent permet de surmonter le problème de la compatibilité du NAT et les résultats des simulations prouvent que la perte est presque nulle ce qui permet de montrer l'efficacité de DHARMA. De plus, quand les facteurs k des clients sont proches de 1, l'utilisation de DHARMA rend le système extensible.

Actuellement, nous implémentons notre architecture sous forme d'intergiciel à installer sur des machines hôtes. Un prototype de base écrit en C est disponible, il utilise l'API des sockets et s'exécute sous LINUX. Les applications qui utilisent notre intergiciel seront capables d'établir des réseaux recouvrants auto-organisés, efficaces et extensibles. Ces réseaux recouvrants fourniront un soutien autonome pour la gestion d'adressage, du nommage et permettent de libérer les applications de plusieurs limitations au niveau réseau.

6.2 Perspectives

A l'issue de notre travail et malgré les résultats satisfaisants obtenus, beaucoup de questions restent ouvertes et plusieurs voies de recherche peuvent être suggérées en continuité des travaux entamés dans cette thèse. Elles concernent aussi bien les développements algorithmiques que les aspects architecture et implémentation. Parmi les perspectives envisagées nous pouvons distinguer :

- Dans un premier temps, nous envisageons d'effectuer des tests supplémentaires afin de valider l'intégralité de nos résultats de conception et cela dans des conditions réelles.
- Nous allons conduire de nouvelles expérimentations afin d'évaluer les performances du transfert des données dans un réseau recouvrant (chaînage de connexions TCP).

- Une autre perspective consiste à améliorer notre schéma d'allocation d'adresses surtout en ce qui concerne les nouveaux noeuds et les serveurs de noms.
- Un autre axe de recherche intéressant serait d'étudier la possibilité d'ajouter une autre fonctionnalité à notre architecture. Elle consisterait à contrôler la vitesse du transfert des données dans les terminaux eux-mêmes surtout en cas de congestion en ajoutant un buffer dynamique qui sera contrôlé automatiquement par notre intergiciel.
- Dans ce travail, nous avons considéré que l'application qui utilise notre intergiciel prendra en charge la fiabilité de transmission des données dans le cas où les connexions n'utilisent pas un protocole fiable dans la couche transport. Donc, dans notre étude nous avons supposé que les noeuds dans le réseau recouvrant peuvent se connecter via le protocole TCP. Toutefois, cette hypothèse n'est pas toujours valable, donc il serait intéressant d'étudier la possibilité d'implémenter la fiabilité du transfert des données quand on en aura besoin (e.g. sur IEEE 802.2 et sur UDP).
- Dans ce mémoire, nous avons énuméré un ensemble de paramètres permettant de régler le comportement des mécanismes. Nous envisageons d'effectuer les modifications nécessaires dans le but d'obtenir les meilleures performances possibles en tenant compte d'un côté des paramètres que nous avons considérés comme paramètres fixes et d'un autre côté en exploitant tous les autres paramètres éventuels que nous n'avons pas étudiés dans notre architecture et qui peuvent affecter la performance.

Concernant l'application de DHARMA au P2PTV nous avons posé plusieurs hypothèses qui sont les suivantes :

- Nous avons considéré pour l'étude de l'extensibilité que le système P2PTV est fixe, ce qui veut dire que les aspects temporels ne sont pas pris en compte.
- Nous avons étudié un réseau recouvrant P2PTV qui transmet une seule chaîne TV avec une seule distribution. De plus, nous n'avons pas pris en compte les variations quotidiennes de la taille de la chaîne (i.e. nombre de pairs regardant la chaîne à un instant donné).
- Nous avons supposé que la période de temps (l'intervalle nécessaire pour la régénération des adresses) a la même valeur pour tous les noeuds dans le réseau recouvrant. Cette période pourrait également dépendre de la mobilité du noeud.

Avec ces hypothèses, la performance de notre architecture pourrait être non optimale en utilisation réelle, c'est pourquoi il serait envisageable d'effectuer des modifications sur celle-ci afin de s'affranchir de ces hypothèses restrictives et par conséquent d'améliorer notre architecture.

Nous avons vu dans cette dernière section que les améliorations possibles concernant notre architecture DHARMA sont nombreuses et ouvrent des perspectives de travaux de recherche encore importantes dans ce domaine vaste que sont les réseaux recouvrants.

Bibliographie

[ABC+02] Atul Adya, William J. Bolosky, Miguel Castro, Gerald Cermak, Ronnie Chaiken, John R. Douceur, Jon Howell, Jacob R. Lorch, Marvin Theimer, and Roger P. Wattenhofer. Farsite : federated, available, and reliable storage for an incompletely trusted environment. In *OSDI '02 : Proceedings of the 5th symposium on Operating systems design and implementation*, pages 1–14, Boston, Ma, December 2002.

[ABKM01] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *Proceedings of the 18th ACM SOSP*, October 2001.

[ABKR05] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Rohit N. Rao. Improving web availability for clients with monet. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design and Implementation*, volume 2, pages 115–128, Boston, MA, May 2005.

[AGDI03] Halevy Alon, Ives Zachary G, Suciu Dan, and Tatarinov Igor. Schema mediation in peer data management systems. *ScholarlyCommons@Penn*, pages 505–516, 2003.

[AJY00] Cengiz Alaettinoglu, Van Jacobsen, and Haobo Yu. Towards mili-second igp convergence. Technical report, Internet Engineering Task Force, Novembre 2000.

[AMS+03] Aditya Akella, Bruce Maggs, Srinivasan Seshan, Anees Shaikh, and Ramesh Sitaraman. A measurement-based analysis of multihoming. In *Proceedings of the SIGCOMM Symposium on Communications Architectures and Protocols ACM SIGCOMM*, pages 353 – 364, Karlsruhe, Germany, August 2003.

[AMZ06] S. Ali, A. Mathur, and H. Zhang. Measurement of commercial peer-to-peer live video streaming. In *Proceedings of the ICST Workshop on Recent Advances in Peer-to-Peer Streaming*, 2006.

[AP01] Rowstron Antony and Druschel Peter. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP'01)*, volume 35, pages 188–201, Banff, Alberta , Canada, October 2001.

[APM+04] Aditya Akella, Jeff Pang, Bruce Maggs, Srinivasan Seshan, and Anees Shaikh. A comparison of overlay routing and multihoming route control. In *SIGCOMM 04 : Proceedings of the 2004 conference on Applications*,

technologies, architectures, and protocols for computer communications, volume 34, pages 93–106, Portland, Oregon, USA, August 2004.

[ASS04] Aditya Akella, Srinivasan Seshan, and Anees Shaikh. Multihoming performance benefits : An experimental evaluation of practical enterprise strategies. In *Proceedings of the USENIX Annual Technical Conference 2004 on USENIX Annual Technical Conference*, page 9, Boston, MA, June 2004.

[AWSBL99] William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley. The design and implementation of an intentional naming system. In *Proceedings of 17th ACM SOSP*, 1999.

[AWtTW02] John Apostolopoulos, Tina Wong, Wai tian Tan, and Susie Wee. On multiple description streaming with content delivery networks. In *INFOCOM 2002 Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies Proceedings*, volume 3, pages 1736–1745, June 2002.

[BKK+03a] Hari Balakrishnan, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Looking up data in p2p systems. *Communications of the ACM*, 46(2) :43–48, February 2003.

[BKK+03b] Suman Banerjee, Christopher Kommareddy, Koushik Kar, Bobby Bhattacharjee, and Samir Khuller. Construction of an efficient overlay multicast infrastructure for real-time applications. In *Proceedings of IEEE INFOCOM'03*, 2003.

[BL86] Kantor Brian and Phil Lapsley. Network news transfer protocol. Request For Comments 977, Internet Engineering Task Force, February 1986.

[BLJ05] David A. Bryan, Bruce B. Lowekamp, and Cullen Jennings. Sosimple : A serverless, standards-based, p2p sip communication system. In *In Proceedings of the International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications (AAAIDEA)*, pages 42–49, June 2005.

[BLV05] Alberto Blanc, Yi-Kai Liu, and Amin Vahdat. Designing incentives for peer-to-peer routing. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 1, pages 374– 385, March 2005.

[BPTU03] Faruk Bagci, Jan Petzold, Wolfgang Trumler, and Theo Ungerer. Ubiquitous mobile agent system in a p2p network. In *Proc. of System Support for Ubiquitous Computing Workshop at the Fifth Annual Conference on Ubiquitous Computing (UbiComp 2003)*, pages 12–15, Seattle, USA, October 2003.

[BSR07] Hariri Behnoosh, Shirmohammadi Shervin, and Pakravan Mohammad Reza. Distributed topology control algorithm for p2p based simulations. In *Distributed Simulation and Real-Time Applications, 2007 DS-RT 2007 11th IEEE International Symposium*, pages 68–71, Chania, Greece, October 2007.

[Bus93] Randy Bush. Fidonet : Technology, tools, and history. *Communications of the ACM*, 36(8) :31–35, August 1993.

[CCR04] Miguel Castro, Manuel Costa, and Antony Rowstron. Peer-to-peer overlays : structured, unstructured, or both ? Technical report, Microsoft Research, Cambridge, July 2004.

[CDG+02a] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Security for structured peer-to-peer overlay networks. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI'02)*, December 2002.

[CDG+02b] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *OSDI '02 : Proceedings of the 5th symposium on Operating systems design and implementation*, volume 36, pages 299–314, Boston, Massachusetts, December 2002.

[CDGN01] Bharat Chandra, Mike Dahlin, Lei Gao, and Amol Nayate. End-to-end wan service availability. In *In Proceeding 3rd USITS*, pages 97–108, San Francisco, CA, 2001.

[CDK+03] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. Splitstream : High-bandwidth multicast in a cooperative environment. In *Proceedings of the nineteenth ACM symposium on Operating systems principles (SOSP)*, volume 37, pages 298–313, Bolton Landing, NY, USA, October 2003.

[CGM02] Arturo Crespo and Hector Garcia-Molina. Semantic overlay networks for p2p systems. Technical report, University of California at Berkeley, 2002.

[CMM02] Russ Cox, Athicha Muthitacharoen, and Robert Morris. *Serving DNS using Chord*, pages 155–165. Springer-Verlag, 2002.

[Coh03] Bram Cohen. Incentives build robustness in bittorrent. In *In Proc. of the Workshop on Economics of Peer-to-Peer Systems (P2PEcon'03)*, Berkeley, USA, June 2003.

[Cor01] Inktomi Corp. *Inktomi Content Networking*. http ://www.inktomi.com/products/cns/, 2001.

[Cow99] Lenore Cowen. Compact routing with minimum stretch. In *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms*, January 1999.

[CRMC08] Meeyoung Cha, Pablo Rodriguez, Sue Moon, and Jon Crowcroft. On next-generation telco-managed p2p tv architectures. In *Proceedings of the 7th International Workshop on Peer-to-Peer Systems*, 2008.

[CS+01] Ian Clarke, , Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet : A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009 :46–66, 2001.

[CZ04] Y. Chu and H. Zhang. Considering altruism in peer-to-peer internet streaming broadcast. In *Proceedings of the ACM NOSSDAV*, 2004.

[DLM07] John Douceur, Jay Lorch, and Thomas Moscibroda. Maximizing total upload in latency-sensitive p2p applications. In *Proceedings of the 19th ACM SPAA*, pages 270–279, 2007.

[DO03] Diego Doval and Donal O'Mahony. Overlay networks : A scalable alternative for p2p. *IEEE Internet Computing*, 7(4) :79–82, July-August 2003.

[DR01] Peter Druschel and Antony I. T. Rowstron. Past : A large-scale, persistent peer-to-peer storage utility. In *In Proceedings of the 8th IEEE Workshop on Hot Topics in Operating Systems (HotOS)*, pages 75–80, May 2001.

[Dru02] Peter Druschel. *Peer-to-Peer Systems*. Springer, 2002.

[EGP98] Tamar Eilam, Cyril Gavoille, and David Peleg. Compact routing schemes with low stretch factor. In *Proceedings of the 17th ACM Symposium on Principles of Distributed Computing*, pages 11–20, August 1998.

[FFD+01] DABEK Frank, KAASHOEK M. Frans, KARGER David, MORRIS Robert, and STOICA Ion. Wide-area cooperative storage with cfs. In *In Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, volume 35, pages 202–215, Chateau Lake Louise, Banff, Alberta, Canada, October 2001.

[FG01] Paul Francis and Ramakrishna Gummadi. Ipnl : A nat-extended internet architecture. In *Proceedings of ACM SIGCOMM'01*, 2001.

[Fra99] Paul Francis. *Yoid Tree Management Protocol (YTMP)*. http ://www.icir.org/yoid/docs/ycHtml/node16.html, 1999.

[Fra00] Paul Francis. Yoid : Extending the internet multicast architecture. Technical report, ATT Center for Internet Research at ICSI, 2000.

[FYT91] F.Teraoka, Y. Yokote, and M. Tokoro. A network architecture providing host migration transparency. In *Proceedings of ACM SIGCOMM'91*, 1991.

[GBL+03] Indranil Gupta, Ken Birman, Prakash Linga, Al Demers, and Robbert van Renesse. Kelips : Building an efficient and stable p2p dht through increased memory and background overhead. *Lecture Notes in Computer Science*, 2735 :160–169, 2003.

[GSAA04] Abhishek Gupta, Ozgur D. Sahin, Divyakant Agrawal, and Amr El Abbadi. Meghdoot : Content-based publish/subscribe over p2p networks. *Springer-Verlag*, 3231 :254–273, 2004.

[GSG02] P. Krishna Gummadi, Stefan Saroiu, and Steven Gribble. A measurement study of napster and gnutella as examples of peer-to-peer file sharing systems. In *In Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, volume 32, page 82, January 2002.

[Han94] Eriksson Hans. Mbone : the multicast backbone. *Communications of the ACM*, 37(8) :54–60, August 1994.

[HBMS04] Oliver Heckmann, Axel Bock, Andreas Mauthe, and Ralf Steinmetz. The edonkey file-sharing network. In *In Peter Dadam and Manfred Reichert, editors, Proceedings of the Workshop on Algorithms and Protocols for Efficient Peer-to-Peer Applications (Informatik),GI Jahrestagung (2)*, volume 51 of *LNI*, pages 224–228, September 2004.

[hCRSZ01] Yang hua Chu, Sanjay Rao, Srinivasan Seshan, and Hui Zhang. Enabling conferencing applications on the internet using an overlay multicast architecture. In *Proceedings of ACM SIGCOMM'01*, August 2001.

[hCRZ00] Yang hua Chu, Sanjay Rao, and Hui Zhang. A case for end system multicast. In *Proceedings of ACM SIGMETRICS'00*, 2000.

[HDA05] Qi He, Constantinos Dovrolis, and Mostafa Ammar. On the predictability of large transfer tcp throughput. In *in Proceedings of ACM SIGCOMM*, pages 145–156, Philadelphia, Pennsylvania, USA, August 2005.

[Hel02] S. Helder, D.A. Jamin. End-host multicast communication using switch-trees protocols. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, page 419, Berlin, May 2002.

[HIMT03] Alon Y. Halevy, Zachary G. Ives, Peter Mork, and Igor Tatarinov. Piazza : data management infrastructure for semantic web applications. In *Proceedings of the 12th international conference on World Wide Web (WWW'03)*, pages 556–567, Budapest, Hungary, May 2003.

[Hin96] Robert M. Hinden. Ip next generation overview. *Communications of the ACM*, 39(6) :61–71, June 1996.

[HLL+07] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and Keith Ross. A measurement study of a large-scale p2p iptv system. *IEEE Transactions on Multimedia*, 9(8), 2007.

[HM] Mickaël Hoerdt and Damien Magoni. *network cartographer (nec)*. Université Louis Pasteur, https ://dpt-info.u-strasbg.fr/~magoni/nec/.

[IGG00] Paolo Fasano Ivano Guardini and Guglielmo Girardi. *Ipv6 operational experience within the 6bone*. http ://www.isoc.org/inet2000/cdproceedings/1e/1e_1.htm, January 2000.

[ITU94] T. S. S. International Telecommunication Union ITU. Terms and definitions related to quality of service and network performance including dependability. Technical report, ITU Recommendation E.800 (08/94), August 1994.

[JKL+00] Jannotti John, Gifford David K., Johnson Kirk L., Kaashoek Frans M., and O'Toole James W. Overcast : Reliable multicasting with an overlay network. In *In Proceedings of the 4th USENIX Symposium on Operating System Design and Implementation*, pages 197–212, San Diego, CA, October 2000.

[KA98a] S. Kent and R. Atkinson. Ip encapsulating security payload (esp). Request For Comments 2406, Internet Engineering Task Force, November 1998.

[KA98b] S. Kent and R. Atkinson. Security architecture for the internet protocol. Request For Comments 2401, Internet Engineering Task Force, November 1998.

[KAHC05] K.Aisopos, A.P.Kakarountas, H.E.Michail, and C.E.Goutis. High throughput implementation of the new secure hash algorithm through partial unrolling. In *in Proceeding of 2005 IEEE International Workshop on Signal Processing Systems (SIPS '05)*, pages 99– 103, Athens, Greece, November 2005.

[Kar05] Pradnya Karbhari. *Throughput and Fairness Considerations in Overlay Networks for Content Distribution*. PhD thesis, College of Computing Georgia Institute of Technology, Georgia, USA, December 2005.

[KB04] Gu-In Kwon and John Byers. Roma : Reliable overlay multicast with loosely coupled tcp connections. In *Proceedings of IEEE INFOCOM'04*, March 2004.

[KBC+00] John Kubiatowicz, David Bindel, Yan Chen, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westly Weimer, Christopher Wells, and Ben Zhao. Oceanstore : An architecture for global-scale persistent storage. In *Proceedings of ACM ASPLOS*, Cambridge, MA, November 2000.

[KBS02] Pete Keleher, Bobby Bhattacharjee, and Bujor Silaghi. Are virtualized overlay networks too much of a good thing. In *Peer-to-Peer Systems, First International Workshop, IPTPS*, volume 2429, pages 225–231, Cambridge, MA, USA, March 2002.

[KFY04] Dmitri Krioukov, Kevin Fall, and Xiaowei Yang. Compact routing on internet-like graphs. In *Proceedings of IEEE INFOCOM'04*, March 2004.

[KLR07] R. Kumar, Y. Liu, and K. Ross. Stochastic fluid theory for p2p streaming systems. In *Proceedings of the IEEE Infocom*, 2007.

[KRRS04] Brad Karp, Sylvia Ratnasamy, Sean Rhea, and Scott Shenker. Spurring adoption of dhts with openhash, a public dht service. In *Proceeding of the 3rd International Workshop on Peer-to-Peer Systems IPTPS*, February 2004.

[Kum96] Vinay Kumar. *MBONE : Interactive Multimedia on the Internet*. New Riders Publishing, 1996.

[LB99] Jorg Liebeherr and Tyler K. Beam. Hypercast : A protocol for maintaining multicast group members in a logical hypercube topology. In *Proceedings of the First International COST264 Workshop on Networked Group Communication*, volume 1736, pages 72–89, 1999.

[LJL+06] X. Liao, H. Jin, Y. Liu, L. Ni, and D. Deng. Anysee : Peer-to-peer live streaming. In *Proceedings of the IEEE Infocom*, 2006.

[LK00] A.M. Law and W.D. Kelton. *Simulation Modelling and Analysis*. McGraw-Hill, 3rd edition, 2000.

[LKR04] Jian Liang, Rakesh Kumar, and Keith W. Ross. *Understanding KaZaA*. http ://cis. poly.edu/ ross/papers/UnderstandingKaZaA.pdf, 2004.

[LM03] Zhi Li and Prasant Mohapatra. Hostcast : a new overlay multicasting protocol. In *IEEE International Communications Conference (ICC)*, volume 1, pages 702– 706, Anchorage (Alaska), May 2003.

[LM04] Zhi Li and Prasant Mohapatra. Impact of topology on overlay routing service. In *Proceedings of IEEE INFOCOM'04*, March 2004.

[Loo03] Alfred W. Loo. The future of peer-to-peer computing. *Commun. ACM*, 46(9) :56–61, September 2003.

[LRS02] Qin Lv, Sylvia Ratnasamy, and Scott Shenker. Can heterogeneity make gnutella scalable ? *Lecture Notes in Computer Science*, 2429 :94–103, 2002.

[Mag] Damien Magoni. *network manipulator (nem)*. Université Louis Pasteur, https ://dpt-info.u-strasbg.fr/~magoni/nem/.

[Mag03a] Damien Magoni. Hierarchical addressing and routing mechanisms for distributed applications over heterogeneous networks. In *Proceedings of the 3rd International Conference on Computational Science – Workshop on Innovative Solutions for Grid Computing*, pages 1093–1102, Melbourne, Australia, June 2003.

[Mag03b] Damien Magoni. A scalable and unifying architecture for deploying advanced protocols in the internet. In *Proceedings of the 10th International Conference on Telecommunications*, pages 1001–1007, Papeete, Tahiti, French Polynesia, February 2003.

[Mal98] Gary Malkin. Rip (routing information protocol) version 2. Request For Comments 2453, Internet Engineering Task Force, November 1998.

[MKG+03] J.E. McGeehan, S. Kumar, D. Gurkan, S.M.R.M. Nezam, A.E. Willner, K.R. Parameswaran, M.M. Fejer, J. Bannister, and J.D. Touch. All-optical decrementing of a packet's time-to-live (ttl) field and subsequent dropping of a zero-ttl packet. *Lightwave Technology, Journal of Publication*, 21(11) :2746– 2752, November 2003.

[MMGC02] Athicha Muthitacharoen, Robert Morris, Thomer M. Gil, and Benjie Chen. Ivy : a read/write peer-to-peer file system. *SIGOPS Oper. Syst. Rev.*, 36(SI) :31–44, 2002.

[Moc87] P. Mockapetris. Domain names - implementation and specification. Request for comments, Internet Engineering Task Force, November 1987.

[MPRG07] Gustavo Marfia, Giovanni Pau, Paolo Di Ricoy, and Mario Gerla. P2p streaming systems : A survey and experiments. In *Proceedings of the STreaming Day*, 2007.

[MS90] M. V. Mannino and L. D. Shapiro. Extensions to query languages for graph traversal problems. *IEEE Transactions on Knowledge and Data Engineering*, 2(3) :353–363, September 1990.

[NAN99] NANOG. *The North American Network Operators Group mailing list archive*. http :// www.cctec.com/maillists/nanog/index.html, November 1999.

[NHB04] Daswani Neil, Garcia-Molina Hector, and Yang Beverly. *Open Problems in Data-Sharing Peer-to-Peer Systems*, chapter 1, pages 1–15. Springer-Verlag, 2004.

[NWD03] Tsuen-Wan Johnny Ngan, Dan S. Wallach, and Peter Druschel. Enforcing fair sharing of peer-to-peer resources. *Lecture Notes in Computer Science*, 2735 :149–159, 2003.

[NWQ+02] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmér, and Tore Risch. a p2p networking infrastructure based on rdf. In *Proceedings of the 11th international conference on World Wide Web (WWW'02)*, pages 604–615, Honolulu, Hawaii, USA, May 2002.

[Ora01] Andy Oram. *Peer-to-Peer : Harnessing the Power of Disruptive Technologies*. O'Reilly and Associates, 2001.

[Pap01] Christos H. Papadimitriou. Algorithms, games, and the internet. *Lecture Notes in Computer Science*, 2076 :749–753, July 2001.

[Pax97a] Vern Paxson. End-to-end internet packet dynamics. In *Proceedings of ACM SIGCOMM'97*, Cannes, France, September 1997.

[Pax97b] Vern Paxson. End-to-end routing behavior in the internet. *IEEE / ACM Transactions on Networking*, 5(5) :601–615, October 1997.

[PF03] Tom Pepper and Justin Frankel. *Gnutella*. http ://gnutella.wego.com, December 2003.

[PF05] Tom Pepper and Justin Frankel. *Gnutella2*. http ://www.gnutella2.com, 2005.

[Pia06] Fabio Pianese. Pulse, a flexible p2p live streaming system. In *Proceedings of the 9th IEEE Global Internet Symposium*, 2006.

[PP07] Fabio Pianese and Diego Perino. Resource and locality awareness in an incentive-based p2p live streaming system. In *Proceedings of the ACM P2P-TV*, 2007.

[PPKB07] Fabio Pianese, Diego Perino, Joaquin Keller, and Ernst Biersack. Pulse : An adaptive, incentive-based, unstructured p2p live streaming system. *IEEE Transactions on Multimedia*, 9(6), 2007.

[PRR97] C. Greg Plaxton, Rajmohan Rajaraman, and Andréa W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *SPAA '97 : Proceedings of the ninth annual ACM symposium on Parallel algorithms and architectures*, pages 311–320, Newport, Rhode Island, United States, 1997.

[PST04] Larry Peterson, Scott Shenker, and Jonathan Turner. Overcoming the internet impasse through virtualization. In *In Proceeding 3rd Workshop on Hot Topics in Networks (HotNets-III)*, November 2004.

[PSVW01] Dimitrios Pendarakis, Sherlia Shi, Dinesh Verma, and Marcel Waldvogel. Almi : An application level multicast infrastructure. In *In Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS)*, volume 3, pages 49–60, San Francisco, CA, USA, March 2001.

[PWCS02] Venkata N. Padmanabhan, Helen J. Wang, Philip A. Chou., and Kunwadee Sripanidkulchai. Distributing streaming media content using cooperative networking. In *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video NOSSDAV*, pages 177–186, Miami, Florida, USA, May 2002.

[RD01] Antony Rowstron and Peter Druschel. Pastry : Scalable, distributed object location and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218 :329–350, 2001.

[RFaKS00] Sylvia Ratnasamy, Paul Francis, Mark Handley andRichard Karp, and Scott Shenker. A scalable content addressable network. Technical report, University of California at Berkeley, 2000.

[RFH+01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications ACM SIGCOMM*, volume 31, pages 161–172, San Diego, California, United States, August 2001.

[RFI02] Matei Ripeanu, Ian Foster, and Adriana Iamnitchi. Mapping the gnutella network : Properties of largescale peer-to-peer systems and implications for system design. *LIEEE Internet Computing Journal*, 6(1) :50–57, January 2002.

[RKCD01] Antony Rowstron, Anne-Marie Kermarrec, Miguel Castro, and Peter Druschel. *Scribe : The Design of a Large-Scale Event Notification Infrastructure*, pages 30–43. Springer-Verlag, 2001.

[RL95] Yakov Rekhter and Tony Li. A border gateway protocol 4 (bgp-4). Request For Comments 1771, Internet Engineering Task Force, March 1995.

[RS04] R. Rejaie and S. Stafford. A framework for architecting peer-to-peer receiver-driven overlays. In *Proceedings of the ACM NOSSDAV*, pages 42–47, 2004.

[SAR99] Akamai Technologies SARL. *Akamai*. http ://www.akamai.com, 1999.

[SAZ+02] Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, and Sonesh Surana. Internet indirection infrastructure. In *Proceedings of ACM SIGCOMM'02*, 2002.

[SB00] Alex Snoeren and Hari Balakrishnan. An end-to-end approach to host mobility. In *Proceedings of 6th ACM MobiCom*, 2000.

[SBR06] Y.-W. Sung, M. Bishop, and S. Rao. Enabling contribution awareness in an overlay broadcasting system. In *Proceedings of the ACM SIGCOMM*, 2006.

[SCH+99] Stefan Savage, Andy Collins, Eric Hoffman, John Snell, and Thomas Anderson. The end-to-end effects of internet path selection. In *Proceedings of ACM SIGCOMM'99*, Cambridge, Massachusetts, USA, September 1999.

[Sch01] Rüdiger Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *Proceedings of the IEEE 2001 International Conference on Peer-to-Peer Computing (P2P2001)*, pages 101–102, Linköping, Sweden, August 2001.

[SECHS02] Ajmani Sameer, Clarke Dwaine E., Moh Chuang-Hue, and Richman Steven. *ConChord : Cooperative SDSI certificate storage and name resolution*, pages 141–154. Springer-Verlag, 2002.

[SGMH04] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H.Zhang. The feasibility of supporting large-scale live streaming applications with dynamic application end-points. In *Proceedings of the ACM SIGCOMM*, 2004.

[SKL+02] Dejan S.Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu. Peer-to-peer computing. Technical report, HP Laboratories Palo Alto, March 2002.

[SMK+01] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord : A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM 2001*, pages 149–160, August 2001.

[SML06] Khaldoon Shami, Damien Magoni, and Pascal Lorenz. Autonomous, scalable, and resilient overlay infrastructure. *KICS/IEEE Journal of Communications and Networks*, 8(4) :378–390, 2006.

[SML07] Khaldoon Shami, Damien Magoni, and Pascal Lorenz. A scalable middleware for creating and managing autonomous overlays. In *Proceedings of the 2nd IEEE/Create-Net/ICST International Conference on Communication System Software and Middleware*, pages –, Bangalore, India, January 2007.

[SMLL06] Khaldoon Shami, Damien Magoni, Piotr Lipinski, and Pascal Lorenz. Scalable distributed k-resilient name to address binding system for overlays. In *Proceedings of the 5th International Conference on Networking*, pages –, Mauritius, April 2006.

[SMW+09] Khaldoon Shami, Damien Magoni, Wenjie Wang, Hyunseok Chang, and Sugih Jamin. Impacts of peer characteristics on p2ptv networks scalability. In *Submitted to the 28th Conference on Computer Communications (INFOCOM 2009)*, 2009.

[SMZ03] Kunwadee Sripanidkulchai, Bruce Maggs, and Hui Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *INFOCOM 2003 Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 2166–2176, Carnegie Mellon Univ., Pittsburgh, PA, USA, 2003.

[SMZ04] K. Sripanidkulchai, B. Maggs, and H. Zhang. An analysis of live streaming workloads on the internet. In *Proceedings of the ACM IMC*, pages 41–54, 2004.

[Sol00] Clip2 Distributed Search Solutions. *The gnutella protocol specification v0.4*. http ://www9.limewire.com/ developer/gnutella_protocol_0.4.pdf, 2000.

[Spa99] Gene Spafford. *Usenet Software : History and Sources*. ftp ://rtfm.mit.edu/pub/usenet-by-group/news.admin.misc/Usenet Software :

[SR02] Anurag Singla and Christopher Rohrs. Ultrapeers : Another step towards gnutella scalability. Technical report, Gnutella Developer Forum, Novembre 2002.

[SSBK04] Lakshminarayanan Subramanian, Ion Stoica, Hari Balakrishnan, and R. Katz. Overqos : An overlay based architecture for enhancing internet qos. In *Proceedings of the 1st Symposium on Networked Systems Design and Implementation NSDI*, pages 71–84, San Francisco, CA, March 2004.

[SSK99] R. Siamwalla, R. Sharma, and S. Keshav. Discovering internet topology. Technical report, Cornell University, New York, USA, March 1999.

[Sun01] Todd Sundsted. *The practice of peer-to-peer computing : Introduction and history*. http ://www-106.ibm.com/developerworks/java/library/j-p2p/, September 2001.

[TAC05] Su-Wei Tan, A.G.Waters, and John Crawford. Meshtree : reliable low delay degree-bounded multicast overlays. In *Parallel and Distributed Systems, Proceedings 11th International Conference*, volume 2, pages 565– 569, July 2005.

[TEW04] Joe Touch, Lars Eggert, and Yu-Shun Wang. Use of ipsec transport mode for dynamic routing. Request For Comments 3884, Internet Engineering Task Force, September 2004.

[TH98] Joe Touch and Steve Hotz. The x-bone. In *In Proc. Third Global Internet Mini-Conference in onjunction with Globecom '98*, pages 59–68, Sydney, Australia, November 1998.

[THD03] D. Tran, K. Hua, and T. Do. Zigzag : An efficient peer-to-peer scheme for media streaming. In *Proceedings of the IEEE Infocom*, 2003.

[TT04] David Taylor and Jonathan Turner. Towards a diversified internet. Technical report, Washington University in Saint Louis, 2004.

[VdaFR03] Aline Viana, Marcelo Dias de amorim, Serge Fdida, and José F. Rezende. Indirect routing using distributed location information. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 224–234, March 2003.

[VGLN07] Long Vu, Indranil Gupta, Jin Liang, and Klara Nahrstedt. Measurement and modeling of a large-scale overlay for multimedia streaming. In *The Fourth International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, 2007.

[WDY05] Jie Wuy, Fei Daiz, and Shuhui Yang. Iterative local solution for connected dominating set in ad hoc wireless networks. In *In Proceedings of the Second IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS 2005)*, page 8, Washington, DC, November 2005.

[WLZ07] C. Wu, B. Li, and S. Zhao. Magellan : Charting large-scale peer-to-peer live streaming topologies. In *Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS'07)*, page 62, 2007.

[XKL07] S. Xie, G.Y. Keung, and B. Li. A measurement of a large-scale peer-to-peer live video streaming system. In *Proceedings of the International Conference on Parallel Processing Workshops (ICPPW'07)*, page 57, 2007.

[YCMM02] Saito Yasushi, Karamanolis Christos, Karlsson Magnus, and Mahalingam Mallik. Taming aggressive replication in the pangaea wide-area file system. *SIGOPS Oper. Syst. Rev.*, 36(SI) :15–30, 2002.

[YHGSH02] Chu Yang-Hua, Rao S. G., Seshan S., and Zhang Hui. A case for end system multicast. *Selected Areas in Communications, IEEE Journal on*, 20(8) :1456–1471, October 2002.

[Zat] Zattoo inc., http ://zattoo.com/fr/. *ZATTOO*.

[ZHS+04] 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 (J-SAC)*, 22(1) :41–53, January 2004.

[ZKJ01] Ben Y. Zhao, John Kubiatowicz, and Anthony D. Joseph. Tapestry : An infrastructure for fault-tolerant widearea location and routing. Technical report, University of California at Berkeley, April 2001.

[ZLLY05] X. Zhang, J. Liu, B. Li, and T.-S. Yum. Coolstreaming/donet : A data-driven overlay network for live media streaming. In *Proceedings of the 24th IEEE Infocom*, pages 2102–2111, 2005.

Table des figures

Acronymes

ADSL : Asymmetric Digital Subscriber Line
AH : Authentication Header
ALMI : Application Level Multicast Infrastructure
AS : Autonomous System
BGP : Border Gateway Protocol
CAN : Content-Addressable Networks
CDF : Cumulative Distribution Fonction
CDN : Content Delivery/Distribution Network
CFS : Cooperative File System
DAG : Directed Acyclic Graph
DDNS : Dynamic DNS
DHT : Distributed Hash Table
DNS : Domain Name System
DoS : Denial of Service
DVMRP : Distance Vector Multicast Routing Protocol
ECDN : Enterprise Content Distribution/Delivery Network
EON : Experimental OSI-based Network
ESM : End System Multicast
ESP : Encapsulating Security Payload
IETF : Internet Engineering Task Force
INPL : IP Next Layer
INS : Intentional Naming System
IPsec : Internet Protocol security
ISP : Internet Service Provider
MBone : Multicast Backbone
MONET : Mobile Network
NAT : Network Address Translation
NEM : NEtwork Manipulator
OSI : Open Systems Interconnection
PIM : Protocol Independent Multicast
RIP : Routing Information Protocol
ROMA : Reliable Overlay Multicast Architecture
RON : Resilient Overlay Network
RTP : Real-Time Transport Protocol
RTT : Round-Trip Time
SA : Security Association
SDSI : Simple Distributed Security Infrastructure

SHA : Secure Hash Algorithm
SON : Semantic Overlay Network
SPI : Security Parameter Index
SSL : Secure Sockets Layer
TCP : Transmission Control Protocol
TTL : Time To Live
UDP : User Datagram Protocol
VPN : Virtual Private Network
YTMP : Yoid Tree Management Protocol

Résumé

Un nombre grandissant d'applications réseaux permettent de créer des réseaux recouvrants ou plus simplement des recouvrements (« overlays ») au dessus du réseau Internet. Les modes de communication avancés tels quel la diffusion multipoint n'ont pas été déployés avec succès au niveau de la couche réseau et sont actuellement implémentés au niveau de la couche application, ce qui entraîne de ce fait la création de réseaux recouvrants par des applications spécifiques. Les applications distribuées telles que les grilles (« grid ») et les technologies pair-à-pair (« peer-to-peer ») évoluent très rapidement, mais leur déploiement à travers des réseaux hétérogènes tels que les réseaux non adressables de manière globale pose des difficultés. Le manque d'adresses IP a entraîné une explosion des traducteurs d'adresses NAT et a donc créé une grande quantité d'hôtes non adressables globalement. Même si cela ne perturbe pas trop le comportement d'une communication client-serveur, cela n'est plus vrai dans une application distribuée construite sur un modèle grille ou P2P dans lequel n'importe quelle entité peut jouer les deux rôles et a donc besoin d'une adresse globale. Des outils de niveau applicatif formant des réseaux recouvrants (e.g., globus) sont proposés pour résoudre ce problème, mais ils sont souvent spécifiques à des applications distribuées données. Les réseaux recouvrants créés par ces applications nécessitent en interne une forme d'adressage et de routage. Habituellement leurs topologies sont les plus simples possibles (e.g., arbre, anneau) mais avec l'augmentation de leurs tailles, ces réseaux devront être capables de gérer des topologies bien plus complexes dans l'avenir.

L'objectif de cette thèse est de proposer une architecture d'adressage de niveau applicatif et un mécanisme de routage ayant pour but de surmonter les limitations précédemment présentées. Cette architecture sera conçue dans le but d'apporter une plus grande extensibilité et plus de flexibilité pour les communications entre membres d'une application distribuée quelconque formant une topologie non-triviale. L'unique hypothèse est que les réseaux recouvrants soient créés au dessus de l'Internet et que leurs topologies soient par conséquent contraintes par celle de l'Internet lui-même. L'intérêt de cette architecture est que les applications n'auront pas à mettre en place et à maintenir des topologies spécifiques. L'architecture devra aussi fournir un système de nommage permettant de séparer l'espace d'adressage variant avec la dynamicité du réseau par rapport à l'espace de nommage fournissant un référentiel constant aux applications.