# Routage Efficace pour les Réseaux Pair-à-Pair utilisant des Tables de Hachage Distribuées

Marguerite Fayçal


École
Doctorale
d'Informatique,
Télécommunications
et Électronique de Paris

**THÈSE**

présentée à

**TÉLÉCOM PARISTECH**

pour obtenir le grade de

**DOCTEUR de TÉLÉCOM PARISTECH**

Mention **"Informatique et Réseaux"**

par

**Marguerite FAYÇAL**

---

# ROUTAGE EFFICACE POUR LES RÉSEAUX PAIR-À-PAIR UTILISANT DES TABLES DE HACHAGE DISTRIBUÉES

---

Soutenue le 28 mai 2010 devant la Commission d'Examen composée de :

*Président :*

| | |
|---|---|
| Dominique GAÏTI, Pr. | Université de Technologie de Troyes |

*Rapporteurs :*

| | |
|---|---|
| Elena MUGELLINI, Pr. | École d'Ingénieurs et d'Architectes – Fribourg |
| Abdelmadjid BOUABDALLAH, Pr. | Université de Technologie de Compiègne |

*Examinateurs :*

| | |
|---|---|
| Patrick BELLOT, Pr. | Télécom ParisTech |
| Maroun CHAMOUN, Pr. | École Supérieure d'Ingénieurs de Beyrouth |

*Directeur de Thèse :*

| | |
|---|---|
| Ahmed SERHROUCHNI, MC. | Télécom ParisTech |

*À Paul-Éole, mon père, ami et protecteur,*
*À Reine, ma mère, pour sa patience bienveillante,*
*À Jessie, ma sœur et compagne de toujours.*

*« Lorsque nous bâtissons,*
*pensons que nous bâtissons pour toujours. »*

JOHN RUSKIN

# *REMERCIEMENTS*

*Mes travaux de recherche synthétisés dans ce mémoire ayant été menés dans le cadre d'une thèse CIFRE au sein des laboratoires d'Orange Labs (anciennement France Télécom R&D), je tiens à remercier Dr. Yvon GOURHANT pour ses conseils avisés et son accueil au sein de son ancienne équipe CORE/M2I/R2A.*

*Je remercie tout particulièrement le directeur académique de cette thèse, Dr. Ahmed SERHROUCHNI, de Télécom ParisTech (ex – ENST), qui a apprécié mes capacités et m'a encouragée à m'embarquer dans ce projet de longue haleine. Je lui exprime toute ma gratitude pour tous les efforts, directives et encadrements qu'il a entrepris et déployés pour m'aider à atteindre les objectifs, et pour son érudition accueillante qui a bien servi mes efforts et orientations.*

*Je tiens également à remercier les membres de la commission d'examen :*

- *les Professeurs M. Abdelmadjid BOUABDALLAH, de l'Université de Technologie de Compiègne, et Mme Élena MUGELLINI, de l'École d'Ingénieurs et d'Architectes de Fribourg (Suisse), pour l'intérêt qu'ils ont témoigné à ces travaux en assumant la tâche de rapporteur avec toute sa rigueur scientifique ;*
- *le Professeur Mme Dominique GAÏTI, de l'Université de Technologie de Troyes, pour l'honneur qu'elle m'accorde en présidant la commission ;*
- *MM. les Professeurs Patrick BELLOT, de Télécom ParisTech, et Maroun CHAMOUN, de l'École Supérieure d'Ingénieurs de Beyrouth (Liban), pour avoir bien accepté de siéger dans la commission.*

*Je réserve aussi une place à chaque personne – collègue, amie ou connaissance – qui, par un service rendu ou un encouragement, de manière isolée ou répétée, a facilité, de façon directe ou indirecte, l'aboutissement de ce projet.*

*Enfin, je ne pourrais terminer sans exprimer toute ma reconnaissance et ma gratitude affectueuse à mes parents et ma sœur pour leur présence, leur soutien, leur enthousiasme, leurs encouragements, leur patience et leur entrain.*

# ABSTRACT

This dissertation is a synthesis of our research at Orange Labs (formerly France Telecom R&D) to answer a problem identified by the aforesaid network operator and concerning peer-to-peer (P2P) streams.

Commonly likened to file sharing, P2P has many applications. It corresponds to a changing world of software, networks and equipment. Beyond sharing, we are facing a real power (in terms of CPU, bandwidth, storage, etc.) available in a distributed manner.

The rise of P2P requires new systems to meet the needs of users, but also those of ISPs and other network operators. The former seek permanently quite noticeable high quality of service (QoS). The latter aim to optimize the use of network resources (e.g. bandwidth) and to reduce various operations' and management costs (including those following the business agreements they signed with each others).

Hence the interest of this thesis, that aims to let a P2P network be aware of its underlying IP network in order to achieve a system with an efficient routing mechanism that leads to a win-win situation. Our research focuses on systems based on distributed hash tables (DHT), that we study and analyze first.

This dissertation begins with an analysis of the main protocols for discovery of dynamic resources in the different P2P architectures. The requirements for efficient P2P routing are then established. Afterwards, discovery techniques for generating and providing underlay network proximity information are presented, followed by techniques and main systems that exploit such information.

Our research led to the definition, design, specification and both individual and comparative analysis of two systems: CAP (*Context-Aware P2P system*) and NETPOPPS (*Network Provider Oriented P2P System*). The former introduces semantics in the identifiers of peers and objects and is context-aware. The latter simplifies the management of the different identifiers and is network operator

oriented: it enables cooperation between P2P traffic and the underlay's network operator (its policies and network topology).

**Key words:**

Peer-to-Peer (P2P); Distributed Hash Table (DHT); hash function; application routing; identifier management, network operator; Internet Service Provider (ISP); topology-awareness; overlay network; underlay network.

# RÉSUMÉ

Ce mémoire est une synthèse de nos travaux de recherche menés au sein des laboratoires d'Orange Labs (anciennement France Télécom R&D) pour répondre à une problématique identifiée par ledit opérateur et concernant les flux d'échanges en mode pair-à-pair (P2P).

Communément assimilé à un échange de fichiers, le P2P a de nombreuses applications. Il correspond à une évolution du monde du logiciel, des réseaux et des équipements. Au-delà du partage, nous sommes confrontés à une puissance disponible de façon distribuée (en termes de CPU, bande passante, stockage, etc.).

La montée en puissance du P2P exige de nouveaux systèmes pouvant satisfaire les besoins des usagers, mais aussi ceux des fournisseurs d'accès à Internet (FAI) et autres opérateurs de réseaux. Les premiers cherchent en permanence une bonne qualité de service (QoS) bien perceptible. Les seconds aspirent à l'optimisation de l'usage des ressources du réseau (notamment la bande-passante) et à la réduction des différents coûts d'opération et de maintenance (dont ceux découlant de leurs accords économiques inter-opérateurs).

D'où l'intérêt de nos travaux de recherche qui visent à sensibiliser un réseau P2P au réseau IP sous-jacent, afin d'aboutir à un système de routage P2P efficace, en phase avec les politiques des réseaux d'infrastructures sous-jacents. Ces travaux se focalisent sur les systèmes P2P utilisant des tables de hachage distribuées (DHT), après les avoir étudiées et analysées.

Ce mémoire commence par une analyse des principaux protocoles de découverte de ressources dynamiques dans les différentes architectures de réseaux P2P. Les exigences requises pour un routage P2P efficace sont par la suite établies. Il s'en suit une présentation des techniques de génération de l'information de proximité sous-jacente, ainsi que des techniques et principaux systèmes d'exploitation de cette information.

Nos travaux de recherche ont abouti à la définition, la conception, la spécification et l'analyse individuelle et comparative de deux systèmes : CAP

(*Context-Aware P2P system*) et NETPOPPS (*NETwork Provider Oriented P2P System*). Le premier système est sensible au contexte et introduit une sémantique dans les identifiants des pairs et des objets. Le second système est orienté opérateur de réseau, et adapte les flux P2P à la topologie sous-jacente et aux politiques de l'opérateur, tout en simplifiant la gestion des différents identifiants.

**Mots clés :**

Pair-à-pair (P2P) ; table de hachage distribuée (DHT) ; fonction de hachage ; routage applicatif ; gestion des identifiants ; opérateur de réseaux ; fournisseur d'accès à Internet (FAI) ; sensibilité à la topologie sous-jacente ; réseau de recouvrement ; réseau sous-jacent.

# TABLE DES MATIÈRES

*Ce mémoire est une synthèse de nos travaux de recherche menés au sein des laboratoires d'Orange Labs (anciennement France Télécom R&D) pour répondre à une problématique identifiée par ledit opérateur et concernant les flux d'échanges en mode pair-à-pair (P2P).*

# INTRODUCTION

*« Toute la question du monde, c'est :*
*comment combler le fossé*
*qui sépare les êtres humains. »*

JEAN VANIER

## Émergence des réseaux pair-à-pair

Le phénomène de Peer-to-Peer (P2P) est né en 1999 avec le lancement de Napster [Nap99]. Il s'agissait d'un service qui permettait aux utilisateurs « de partager » des fichiers de musique au format MP3.

Napster avait un index central pour localiser les fichiers. L'idée à l'origine de ce service était le stockage décentralisé des fichiers au niveau des points terminaux (plutôt qu'au niveau des serveurs). Cette idée, pourtant simple et à la base du concept de l'Internet, a rencontré un succès énorme.

Suite à la croissance spectaculaire de Napster, des systèmes similaires ont rapidement émergé, les uns après les autres. Mais cette nouvelle génération de systèmes de partage de fichiers a été complètement décentralisée en termes à la fois de stockage et de récupération des ressources.

Cette montée du P2P décentralisé tire sa force des limites du modèle client-serveur. Ce modèle, qui fait tourner Internet, avait pourtant rapidement gagné en popularité depuis le début des années 80. Mais la nature centralisée d'un serveur est victime de la diversification en effervescence continue du réseau mondial. Le modèle client-serveur est devenu sujet à des goulots d'étranglement, facilement attaquable, et difficilement exploitable pour la mise en œuvre de systèmes distribués ou d'applications de partage de fichiers. De plus, toute modification du modèle est excessivement difficile et coûteuse.

Cependant, avec la démocratisation du matériel informatique, de plus en plus puissant, et des connexions à haut débit, l'Internet continue son effervescence. Et les réseaux IP ne cessent de prendre de l'essor. Le nombre de

machines connectées sur les lieux de travail, mais aussi – et parfois en quasi-permanence – à domicile, continue de croître. Les nouvelles technologies sans fil et l'essor de la mobilité ont aussi un impact croissant. Tous ces facteurs rendent possible et favorisent alors l'utilisation des applications décentralisées de partage de fichiers. De manière générale, ce sont toutes les technologies connues sous le nom de P2P, qui se trouvent ainsi favorisées.

Du coup, le P2P n'est point une révolution en lui-même, mais bel et bien une conséquence d'une vraie révolution. Il tire ses racines des infrastructures IP, recouvre le réseau Internet, et a donc toujours existé d'une certaine manière.

**Quelques statistiques du pair-à-pair**

Concrètement, le flux P2P consiste en un échange de données grâce à une application, dite aussi «logiciel client ». Il existe une panoplie de logiciels P2P aux particularités et performances différentes. Certains logiciels sont des variantes permettant d'accéder à un même réseau P2P ; d'autres permettent d'accéder à plusieurs réseaux. Depuis quelques années, les réseaux les plus populaires [SM07] sont BitTorrent [Bit, CC03, Cha05] et eMule [eMu, KB05]. La figure 1 montre l'évolution des trafics sur chacun de ces deux réseaux depuis le second trimestre 2004. Les données y sont représentées proportionnellement au pic le plus haut.

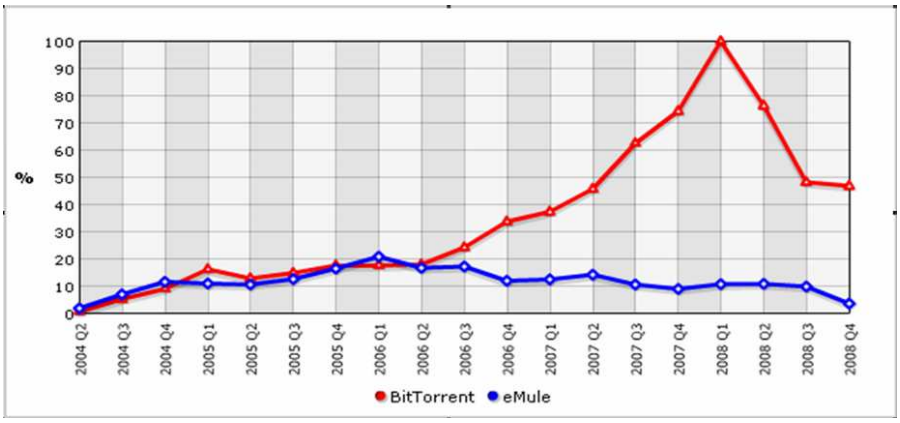


***Figure 1 :*** *Croissance relative des flux bitTorrent et eMule*[1]

[1] **In :** [Evi08]

Quant au type de fichiers les plus échangés au sein des différentes communautés P2P, la vidéo arrive en tête [SM07]. En 2004 déjà, 61,44% du volume global du trafic P2P était constitué de fichiers vidéo [Par04].

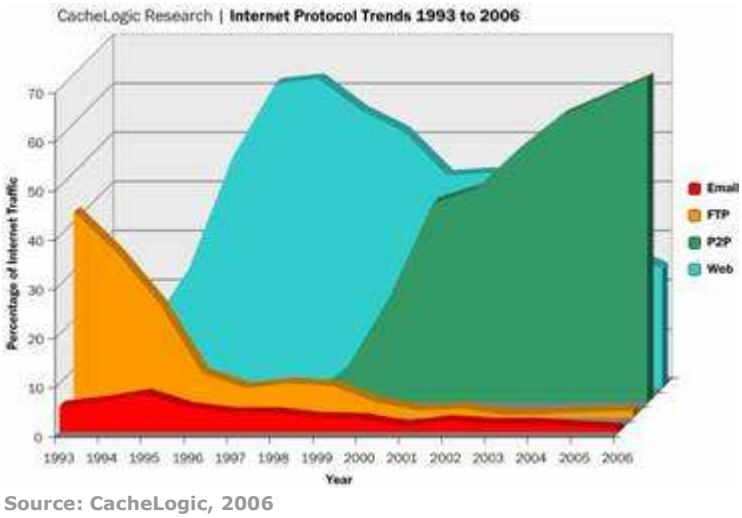


***Figure 2** : Évolution du trafic Internet*[1]

Depuis cinq ans, la popularité des réseaux P2P n'est plus à prouver (cf. fig. 2) : entre 43,5% et 83% du trafic mondial est constitué de flux P2P. Ces chiffres résument des résultats de deux études qui ont été menées par *ipoque* (pour l'Allemagne, le reste de l'Europe, le Moyen-Orient et l'Australie) [SM07] et par *Sandvine* (pour l'Amérique du Nord) [San08a]. Les figures 3 et 4 illustrent ces chiffres qui quantifient l'agrégation des flux P2P montants (*upstream*) et descendants (*downstream*). En effet, les deux types de flux ne représentent pas le même taux d'occupation de la bande passante. Ceci est notamment dû aux débits asymétriques des réseaux ADSL. Cependant les deux études (celle de *ipoque* et celle de *Sandvine*) montrent une nette dominance sur la voie montante du trafic P2P par rapport aux autres types de flux IP. Récemment, *Sandvine* a évalué à plus de 61% le taux journalier moyen d'occupation de la bande passante mondiale par le flux P2P sur la voie montante [San08b].

[1] **In :** [Fer06]

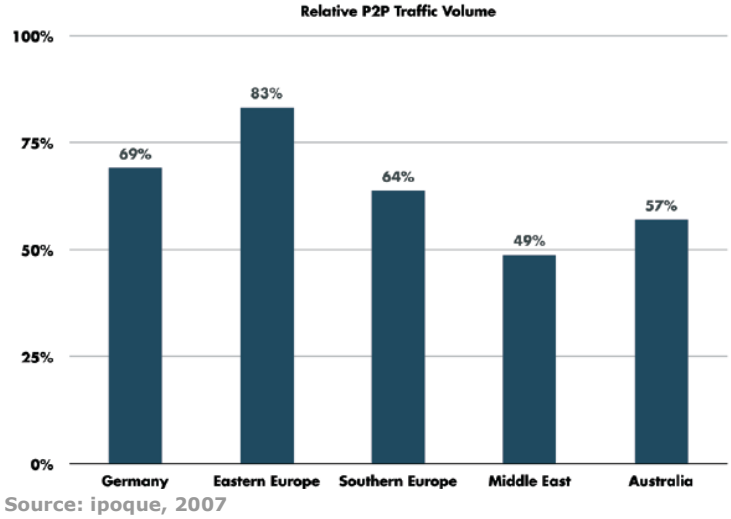


***Figure 3 :*** *Volumes relatifs du trafic P2P, par région*[1]

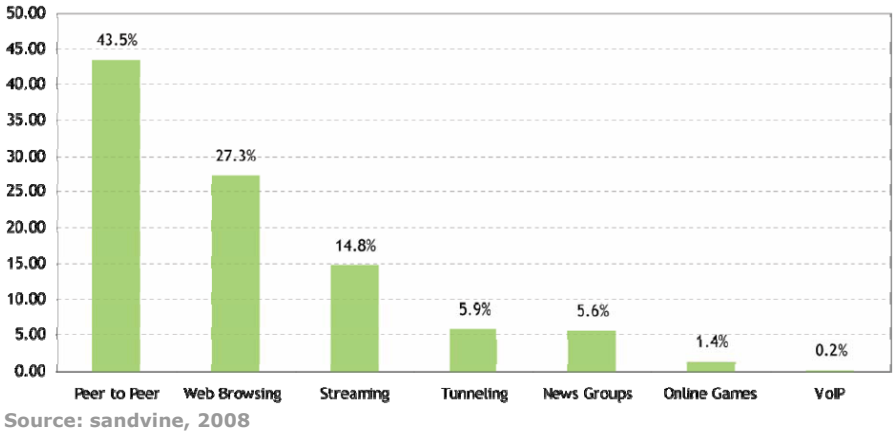


***Figure 4 :*** *Trafic Internet de l'Amérique du Nord représenté par catégorie*[2]

Bien qu'actuellement le trafic P2P domine nettement tous les autres flux Internet, les volumes échangés sur les réseaux P2P croîtraient de près de 400% sur les cinq ans à venir, avec les échanges légaux croissant dix fois plus vite que les illégaux [Dic08]. Cependant, la croissance du P2P est jusque là essentiellement due aux applications de partage de fichiers qui sont majoritairement de type vidéo. Par ailleurs, la vidéo sur Internet, un service de l'Internet indépendant du P2P, est aussi en train de gagner du terrain et devrait voir son flux approcher la moitié de la totalité des flux internet dans cinq ans (cf. fig. 5) [Cis08].

[1] **In :** [SM07]
[2] **In :** [San08a]

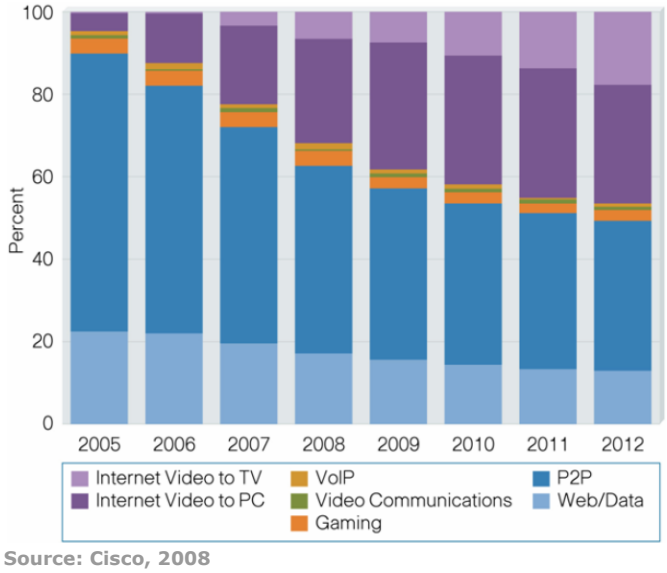


***Figure 5 :*** *Prévisions du trafic Internet global*[1]

En conclusion, sur les cinq années à venir le trafic P2P continuerait à croître en volumes échangés même s'il va perdre du terrain en pourcentage du trafic Internet global.

**Problématique de recherche**

Les réseaux P2P recouvrent l'infrastructure fixe d'Internet. Leur objectif consiste à répartir les ressources de tous les pairs entre eux, pour leur permettre l'accès à des services communs. Ainsi, l'élargissement d'un système P2P à grande échelle est constamment compensé par l'augmentation des ressources mises en partage. Les architectures P2P peuvent donc générer des économies gigantesques de déploiement et de gestion, et créer de la valeur en permettant le développement de toute une panoplie de nouveaux services.

Certes, la décentralisation des systèmes P2P à l'échelle du réseau mondial présente des faiblesses et des menaces. Outre la vaste question de la sécurité, les enjeux sont principalement éthiques et juridiques, tant au plan social qu'économique. Nous ne regarderons cependant pas le revers d'une médaille qui continue de séduire par ses atouts et ses potentiels.

[1] **In :** [Cis08]

Dans un réseau P2P, tous les pairs sont des sources potentielles de données, et les communications entre eux se développent de manière autonome et spontanée. L'avancement des techniques et du comportement des usagers aidant, un tel réseau – comme tout système auto-organisable – est donc indépendant du réseau sous-jacent.

Or, la rapidité dans les recherche et récupération de contenu (ou ressource), même dans un réseau *overlay*, va nécessairement dépendre du réseau sous-jacent. C'est pourquoi sensibiliser le réseau P2P au réseau sous-jacent se manifeste comme une problématique réelle. Cette sensibilisation doit consister à favoriser le dialogue entre pairs voisins dans le réseau sous-jacent (au sens de la proximité sous-jacente, en nombre de sauts IP).

Par ailleurs, la croissance continue du volume du trafic P2P semble intimider les opérateurs de réseaux. D'une part, cette croissance augmente leur nombre d'abonnés aux meilleures offres de débit. D'autre part, cette croissance constitue une menace pour eux. En fait, mis à part le coût de gestion et de maintenance, ceux-ci paient un coût réel pour la bande passante utilisée par leurs abonnés. Par conséquent, une coopération effective entre les deux parties constitue un vrai défi. Elle amènerait chaque pair à trouver et récupérer la ressource qu'il souhaite, selon les critères et politiques de l'opérateur du réseau sous-jacent. Nous nous retrouvons donc là face à une problématique de sensibilisation particulière au réseau sous-jacent du réseau P2P.

Enfin, la majorité des systèmes P2P émergeants, notamment les applications les plus populaires utilisent des tables de hachage distribuées (DHT). En bref, il s'agit d'algorithmes qui garantissent la localisation des ressources disponibles dans un système distribué. Sauf que les DHTs supposent les éléments du système (en l'occurrence, les pairs et les ressources) uniformément répartis ; ce qui est bien exceptionnellement le cas.

Notre problématique se résume alors à la sensibilisation efficace d'un réseau P2P utilisant une DHT à l'infrastructure du réseau sous-jacent, avec une coopération possible entre les flux P2P et les opérateurs de réseaux. Nos travaux se sont d'ailleurs déroulés à Orange Labs (anciennement France Télécom R&D), les laboratoires de recherche d'Orange, opérateur de réseaux.

**Contributions et structure du mémoire**

Après cette introduction, le mémoire est structuré en six chapitres, et se termine par une conclusion qui s'ouvre sur de nouvelles perspectives de recherche.

Le ***premier*** chapitre présente et analyse les réseaux pair-à-pair (P2P) qui sont des réseaux de recouvrement spécifiques. Il élucide leurs différentes architectures et leurs principaux protocoles. Il aborde plus particulièrement ceux à base des tables de hachage distribuées (DHT).

Le ***deuxième*** chapitre spécifie les exigences pour un routage efficace et explique la nécessité des systèmes P2P d'être sensible à la topologie du réseau sous-jacent. Il présente aussi les techniques de génération de l'information de proximité. Le terme 'proximité' désigne une distance proche dans le réseau sous-jacent, en termes de sauts IP.

Le ***troisième*** chapitre présente les techniques d'exploitation de l'information de proximité, et analyse les différentes solutions existantes à base de DHT pour la prise en compte du réseau sous-jacent.

Le ***quatrième*** chapitre définit, spécifie et analyse un système P2P sensible au contexte du réseau. Il s'agit d'un système générique et configurable, qui prend en compte un ou plusieurs paramètres relatifs au contexte du réseau. Cette solution, baptisée CAP (*Context-Aware P2P system*), entend ainsi introduire une sémantique dans les identifiants des pairs et des objets. Pour cela, elle construit la DHT avec une fonction de hachage particulière, à savoir HMAC, qui est présentée au début du chapitre.

Le ***cinquième*** chapitre définit, spécifie et analyse un système P2P orienté opérateur de réseau. Cette solution générique et simple, baptisée NETPOPPS (*NETwork Provider Oriented P2P System*), adapte les flux P2P à la topologie sous-jacente de l'opérateur tout en simplifiant la gestion des identifiants des différents pairs et objets. Elle utilise pour cela la technique de dérivation de clés, qui est présentée au début du chapitre.

Le ***sixième*** chapitre présente une analyse, applicative et comparative, des deux nouveaux systèmes, CAP et NETPOPPS.

Ce mémoire se termine par une ***conclusion*** qui récapitule l'enchaînement progressif des différents chapitres et aboutit aux perspectives de recherche qui s'ouvrent suite aux travaux présentés dans ce mémoire.

# CHAPITRE 1

## *ÉTUDE ET ANALYSE DES RÉSEAUX PAIR-À-PAIR*

*Ce chapitre présente et analyse les réseaux pair-à-pair (P2P) qui sont des réseaux de recouvrement spécifiques. Il élucide leurs différentes architectures et leurs principaux protocoles. Il aborde plus particulièrement ceux à base des tables de hachage distribuées (DHT).*

### 1.1 - Réseaux de recouvrement

Ce paragraphe définit les réseaux de recouvrement et en donne quelques exemples

#### *1.1.1 - Définition*

Un réseau de recouvrement ou *overlay* est un réseau virtuel s'appuyant sur un ou plusieurs réseaux physiques existants. Parmi ces réseaux existants, dits sous-jacents ou *underlay*, l'Internet (ou des sous-réseaux de l'Internet) est un bon exemple.

Dans la suite, ce sont les qualificatifs anglais : « overlay » pour un réseau de recouvrement et « underlay » pour un réseau sous-jacent, qui seront privilégiés.

Le réseau *overlay* est formé d'un sous-ensemble des nœuds du réseau *underlay* et d'un ensemble de liens logiques entre eux. Ces liens permettent une communication « directe » entre les nœuds concernés, en faisant abstraction de la topologie et des protocoles du réseau *underlay*. Nous désignons par nœud tout équipement du réseau, terminal ou intermédiaire, de quelque nature qu'il soit, jouant le rôle de routeur, client ou serveur.

Les réseaux *overlay*s ont vu le jour dès lors qu'un nouveau service, inexistant dans le réseau, devait y être implanté. Ainsi tout problème informatique peut se résoudre par une couche d'indirection ou plutôt une redirection vers un nouveau réseau virtuel qui implémente la solution (cf. fig. 1.1). Aussi peut-on concevoir une superposition de réseaux *overlay*s. Dans ce cas, les réseaux *overlay*s supérieurs s'appuient sur un ou plusieurs autres réseaux intermédiaires ayant l'Internet pour réseau *underlay*.

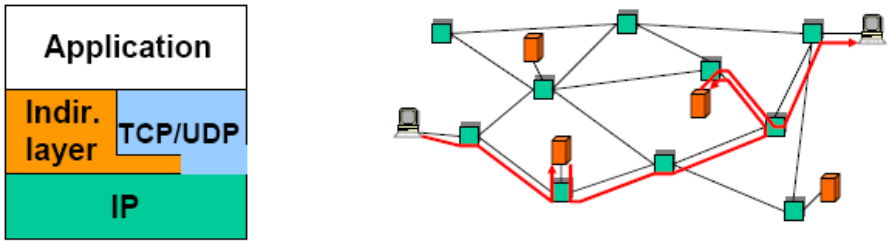


***Figure 1.1 :*** *Indirection et réseau overlay*[1]

### *1.1.2 - Quelques exemples*

Voici quelques exemples de réseaux *overlay*s : Resilient Overlay Network (RON) [ABK+01], Content Delivery Network (CDN) [Mor03] et Eternity [And96].

#### *1.2.4.1 - Resilient Overlay Network*

Resilient Overlay Network (RON) [ABK+01] est une architecture *overlay*, basée sur l'Internet. Elle surveille la qualité des liens Internet pour permettre la détection rapide de routes alternatives lors des interruptions de service des applications unicast distribuées. Elle assure ainsi le rétablissement d'une communication entre un groupe de nœuds en quelques dizaines de secondes au lieu de plusieurs minutes, comme c'est le cas avec BGP-4 [RFC1771]. Il a été, en effet, observé qu'il est parfois plus rapide d'envoyer

[1] **In :** [Sto04]

un message via un ou plusieurs intermédiaires, que de l'envoyer directement à sa destination.

#### *1.2.4.2 - Content Delivery Network*

Content Delivery Network (CDN) [Mor03] est un réseau de serveurs cache offrant un dispositif de gestion optimisée des flux de données grand consommateurs de bande passante, permettant leur transit dans de bonnes conditions et empêchant les goulots d'étranglements.

Les clients du CDN sont des fournisseurs d'accès au réseau ou des fournisseurs de contenu qui souhaitent fiabiliser leur disponibilité. Les principaux fournisseurs sont Akamai [Aka] et Cisco [Cis].

Il est à noter qu'un réseau de diffusion de contenu ne se limite pas au streaming (diffusion en temps réel de contenus audio/vidéo). Aussi est-il possible qu'un réseau pair-à-pair soit utilisé en tant que réseau de diffusion de contenu, moyennant les outils logiciels adéquats.

#### *1.2.4.3 - Eternity*

Eternity [And96] a été développé dans le but de construire un moyen de stockage résistant aux attaques par déni de service. C'est un modèle pair-à-pair de stockage anonyme qui assure la réplication des données et la redondance des chemins. Il n'a jamais été implémenté.

## 1.2 - Caractéristiques et pratiques des réseaux pair-à-pair

### *1.2.1 - Définition*

Le P2P est une contraction de *Peer to Peer,* en anglais, ou, *Pair à Pair*, en français. Le terme P2P sera utilisé par la suite.

Les définitions du P2P sont nombreuses et différentes. La plus générique est la suivante.

Le P2P fait référence à une classe de systèmes et d'applications qui utilisent des ressources distribuées (d'un réseau) pour réaliser une fonction critique de manière décentralisée [MKL+02][1]. Il est à noter que le terme fonction fait allusion à un service, un mécanisme (par exemple, de routage,…), etc.

La figure 1.2 illustre un modèle de réseau P2P.

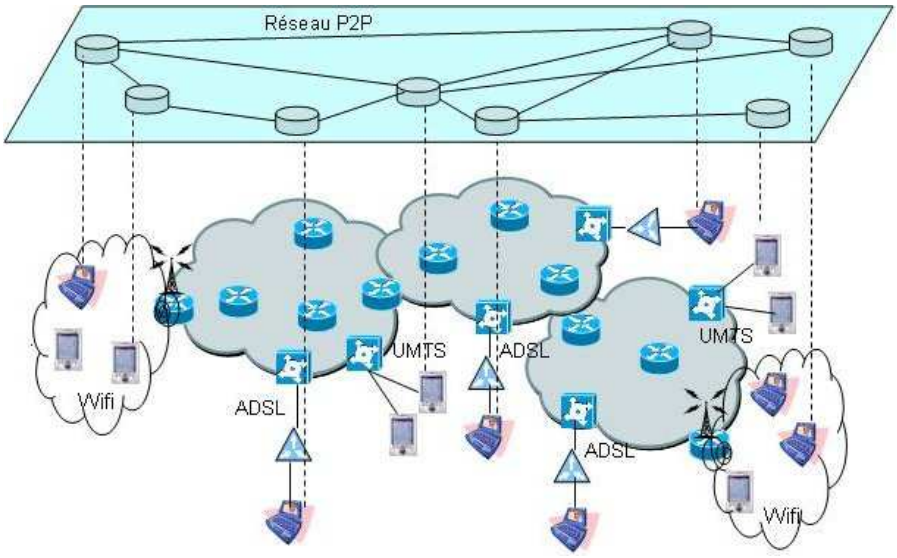


***Figure 1.2 :*** *Modèle de réseau P2P*

### *1.2.2 - Différentes approches*

Dans la multitude des définitions des réseaux P2P, nous distinguons différentes approches qui justifient cette diversité. Trois aspects des réseaux P2P peuvent être identifiés, superposés en niveaux distincts au dessus d'un réseau de communication. Ces niveaux sont – de bas en haut : les infrastructures, les applications et les communautés (cf. fig. 1.3) [SF03].

[1] *"P2P refers to a class of systems and applications that employ distributed resources to perform critical function in a decentralized manner."*

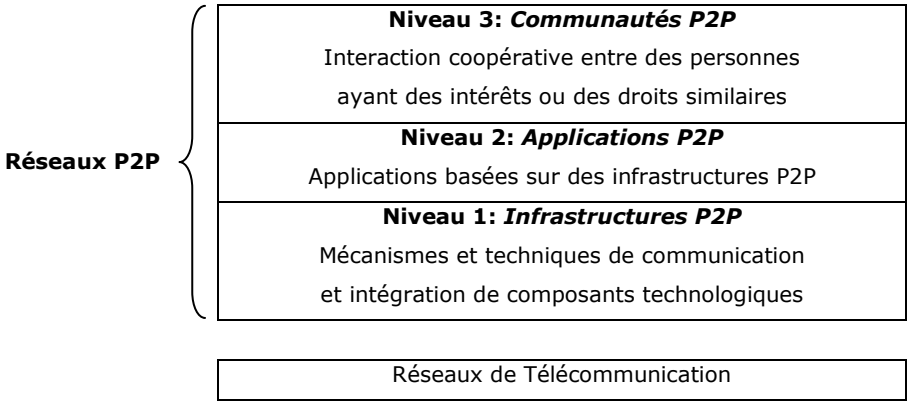


***Figure 1.3 :*** *Différents niveaux d'un réseau P2P*

*1.2.2.1 -* *Infrastructures pair-à-pair*

Les infrastructures P2P se positionnent au dessus de réseaux de communication existants, qui constituent alors la base de tous les niveaux P2P. Elles définissent la topologie et le routage et sont parfois gérées par des algorithmes spécifiques. Elles fournissent des mécanismes et techniques de communication et d'intégration des composants IT, et parfois même des mécanismes d'authentification pour une éventuelle sécurisation des services P2P offerts.

*1.2.2.2 -* *Applications pair-à-pair*

Les applications P2P constituent un niveau intermédiaire entre d'une part les communautés P2P, qui les utilisent, et d'autre part les infrastructures P2P dont elles utilisent les services. C'est à ce niveau que peuvent être gérées les différentes ressources : fichiers, bande passante, capacité de stockage, etc.

En permettant l'échange entre des communautés, les applications génèrent le trafic de données utiles recherchées, mais aussi un trafic – appelé par la suite « de signalisation » – formé par les requêtes et les différentes mises à jour.

À ce niveau, tout comme au niveau des infrastructures, le terme *peer*, ou pair, représente tout composant IT du réseau P2P.

*1.2.2.3 - Communautés pair-à-pair*

Les communautés P2P émergent de l'interaction coopérative entre des personnes ayant des intérêts ou des droits similaires.

Elles représentent le phénomène apparent au niveau social de l'émergence et du développement des réseaux P2P. Du coup, à ce niveau, le terme *peer*, ou pair, prend un sens non technique et représente toute personne utilisant un service P2P à travers des logiciels ou autres applications P2P.

### *1.2.3 - Caractéristiques générales*

Un réseau P2P est un réseau logique distribué dont le principe de base est un principe d'égalité entre des nœuds dynamiques. Ces nœuds sont donc dits *pairs* bien que, comme nous le verrons dans la suite, ce principe ne soit pas vérifié dans tous les types d'architecture P2P.

Les pairs peuvent être de nature différente (routeurs ou périphériques autonomes quelconques), mais sont fonctionnellement identiques, pouvant jouer à la fois le rôle de client, serveur ou routeur.

Le routage se fait au niveau applicatif, dans le sens où il est indépendant du routage dans le réseau sous-jacent – qui est généralement un routage IP. Le routage P2P est généralement basé sur l'usage de clés et consiste en un échange direct entre les pairs. Il utilise les ressources des différents pairs, éliminant ainsi les coûts d'infrastructure.

Chaque pair participant à un réseau P2P met à disposition une partie ou la totalité de ses ressources (données, bande passante, espace de stockage, mémoire cache, CPU, etc.), mais aussi des services (impression, etc.). Chaque pair n'a qu'une visibilité partielle du réseau.

Cette topologie virtuelle – qu'est un réseau P2P – peut avoir une distribution géographique aussi importante que celle du réseau Internet. Elle est aussi instable, vu que la durée de présence de chaque pair est assez différente et variable.

## 1.3 - Architectures types des réseaux pair-à-pair

Plusieurs modèles d'architecture P2P existent. D'ordinaire, c'est le degré de décentralisation qui est retenu pour leur classification, puisque c'est là l'idée de base du P2P. Ce degré de décentralisation de l'architecture caractérise aussi l'index de référence des ressources.

La première génération des applications P2P était à architecture centralisée. Une deuxième génération à architecture décentralisée non structurée a vite vu le jour, suivie par une troisième génération d'applications à architecture hybride.

Au niveau de l'infrastructure P2P, des protocoles et des architectures décentralisées structurées se sont développées en parallèle aux applications P2P, à partir de la deuxième génération.

Nous distinguons donc trois familles d'architectures P2P (cf. fig. 1.4) :

- l'architecture centralisée ;
- l'architecture décentralisée, qui peut être structurée ou non structurée ;
- et l'architecture hybride.

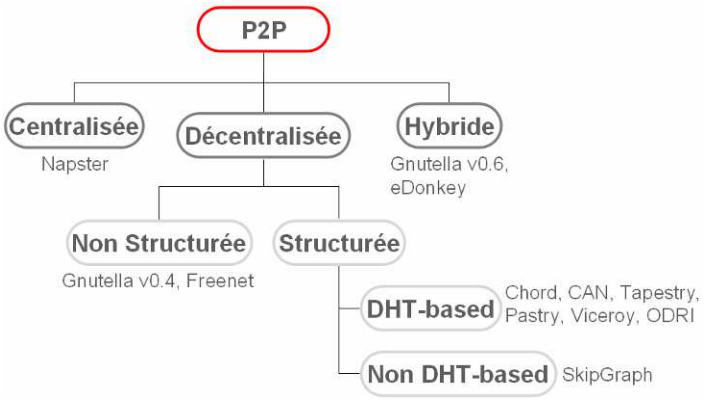


***Figure 1.4 :*** *Classification des systèmes P2P, avec quelques exemples*

### *1.3.1 - Architecture centralisée*

L'architecture centralisée est caractérisée par un index central pour repérer les ressources. Cet index reçoit tous les messages de contrôle et toutes les requêtes, mais par la suite, l'échange se fait directement entre les pairs concernés (cf. fig. 1.5). C'est ce qui distingue principalement cette architecture d'une architecture traditionnelle de type client-serveur.

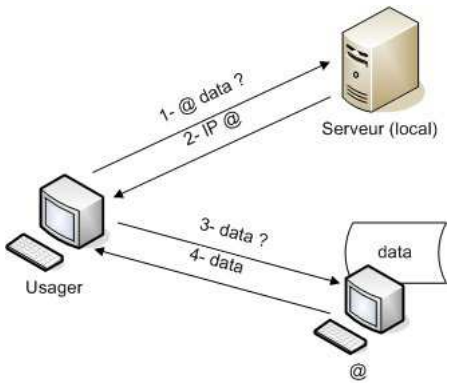


***Figure 1.5 :*** *Architecture P2P centralisée*

Comme toute architecture centralisée, cette architecture facilite la gestion des différentes ressources de l'index et offre une bonne performance de découverte des ressources (le coût de localisation des ressources est faible). Cependant la centralisation représente un goulot d'étranglement au niveau de l'index, tant par la charge de son

utilisation que de sa mise à jour. De plus, au niveau de la sécurité, l'index est le talon d'Achille de ce type d'architecture.

L'exemple type d'une telle architecture est Napster (1999-2002) [Nap99] qui a déclenché le *phénomène* P2P. Il a aussi été le modèle le plus spectaculaire de la réussite technologique du P2P. Avec Napster, les pairs se joignant au réseau informaient l'index des fichiers dont ils disposent, et le recontactaient pour obtenir les coordonnées (adresse IP et numéro de port) d'un autre pair possédant les fichiers recherchés.

BitTorrent [Bit, CC03, Cha05] est un autre exemple d'architecture P2P, qui était cité dans le cadre des architectures partiellement centralisées, à cause de ses différents *tracker*s (sorte d'index central pour la localisation d'une même ressource). Depuis l'introduction d'une couche réseau basée sur le protocole Kademlia [MM02] (cf. p. 54), qui a pour effet de distribuer les informations de chaque *tracker*, BitTorrent est déjà officiellement décentralisé [Cha05, Loe08].

### *1.3.2 - Architecture décentralisée*

#### *1.3.2.1 - Caractéristiques*

Dans l'architecture P2P décentralisée, tous les pairs sont fonctionnellement parfaitement équivalents. C'est un modèle P2P pur. (cf. fig. 1.6) Cette architecture est caractérisée par une décentralisation de l'index, qui devient local à chaque pair, faisant effet de table de routage. La décentralisation rend le système autonome et répartit la charge équitablement. L'architecture décentralisée est donc plus robuste que l'architecture centralisée, mais le temps de découverte des ressources est évidemment plus long.

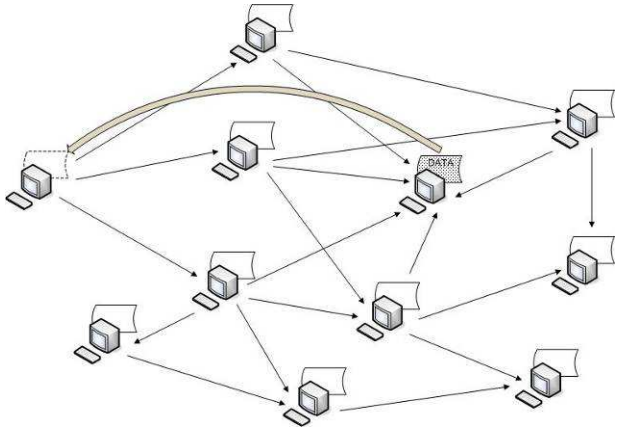


***Figure 1.6 :*** *Architecture P2P décentralisée*

On distingue dans ce type d'architecture deux sous-catégories :

- l'architecture décentralisée non structurée ;
- et l'architecture décentralisée structurée.

*1.3.2.2 -* *<u>Architecture décentralisée non structurée</u>*

Une architecture P2P décentralisée est dite non structurée lorsqu'aucune contrainte topologique n'existe. Un système P2P à architecture décentralisée est aussi dit : système P2P pur. Aucune connaissance de la topologie n'est donc disponible au préalable et chaque pair dans le réseau est autonome. La maintenance et la mise à jour des connexions se font par sondage périodique du voisinage, et la découverte des ressources se fait par une technique d'inondation associant un champ TTL (Time To Live) à chaque message de recherche envoyé, afin de comptabiliser le nombre de retransmissions restantes.

L'architecture décentralisée et non structurée permet de pallier les points faibles de la centralisation. Elle est simple et flexible, et ne requiert pas des requêtes exactes ou des demandes de recherches très précises. Néanmoins, cette architecture ne peut pas être gérée (par un opérateur de réseau ou fournisseur de services). Elle présente aussi

certains inconvénients résultant de l'utilisation de la technique d'inondation et du TTL :

- la génération d'un nombre important de messages, dont découlent :
  - une consommation importante de la bande passante – et donc une consommation des ressources du réseau ;
  - la visite des pairs par des messages (requêtes ou réponses) non sollicités ;
- l'absence de garantie de résultat et de connectivité ;
- un passage à l'échelle assez limité.

L'exemple type d'une architecture P2P décentralisée non structurée est Gnutella dans sa version initiale [Cli01, Rip01]. Le TTL y était typiquement fixé à 7. De ce fait, une mise hors réseau (panne ou déconnexion) d'un nombre aléatoire de pairs scinderait le réseau en deux.

FreeNet [CSW+00] est un autre exemple qui utilise un routage par inondation. Mais il s'agit déjà d'un pont vers les architectures décentralisées structurées. En effet, les données qui y sont stockées ont déjà un identifiant : une clé. De plus d'autres types de clés sont aussi utilisés, comme pour la signature des documents et leur vérification. FreeNet assure l'anonymat et la permanence des informations qui y résident, en insérant chaque clé sur tout chemin parcouru. Cependant, ce fonctionnement présente des risques de sécurité, tels une attaque de type *man-in-the-middle* ou de type *cheval de Troie*.

#### *1.3.2.3 - Architecture décentralisée structurée*

Une architecture P2P décentralisée est dite structurée lorsque la topologie logique est contrôlée par un algorithme et les données sont mises en réseau aussi suivant un

algorithme bien spécifié. Ceci permet à n'importe quel pair de se joindre au réseau ou de le quitter, sans pour autant mettre en cause la cohérence de ce dernier ; un tel réseau est dit « auto-organisé ».

Un autre avantage d'une architecture décentralisée structurée est la garantie qu'elle offre sur les résultats des requêtes.

Ce type d'architecture sera détaillé à la section 1.4 (cf. p. 40), qui lui est consacrée et où différents exemples seront présentés.

### *1.3.3 - Architecture hybride*

L'architecture hybride est une architecture décentralisée dont chaque nœud est l'élément central d'une architecture centralisée. Ces éléments centraux sont des super-pairs par rapport à l'architecture globale.

Pour une homogénéité entre les super-pairs, c'est souvent le critère de la bande passante – identifiée suivant le type de connexion (e.g. par modem ou haut débit) – qui est pris en compte. Mais d'autres critères peuvent aussi entrer en jeu, comme la puissance de calcul, l'espace de stockage, etc. Un nœud peut, avec le temps, à plusieurs reprises gagner et perdre le statut de super-pair.

L'architecture hybride tire avantage simultanément de l'architecture centralisée et de l'architecture décentralisée. Le nombre de pairs mis en jeu dans la découverte des ressources est aussi réduit, et avec lui, le trafic global entre les pairs. Ceci permet une économie de bande passante et facilite le passage à l'échelle.

Un super-pair a cependant les mêmes faiblesses que l'index d'une architecture P2P centralisée. Une amélioration possible est d'assurer une redondance en choisissant, pour un même sous-groupe, un ou plusieurs partenaires au super-pair [YGM03]. L'architecture sera dite à super-pairs redondants.

Des supers-pairs partenaires ont les mêmes connexions aux autres pairs et possèdent des index identiques de toutes leurs ressources. Pour une répartition de charge équitable entre les super-pairs partenaires, les pairs d'un sous-groupe envoient leurs requêtes selon le principe d'ordonnancement *round Robin*.

L'introduction dans l'architecture P2P hybride de super-pairs partenaires améliore la tolérance aux fautes. Et, avec *k* partenaires, la charge d'un partenaire est divisée par *k*. En revanche, le coût d'adhésion d'un nouveau pair au réseau est multiplié par *k* et le nombre de connexions ouvertes par $k^2$.

Voici quelques exemples d'applications basées sur un réseau P2P hybride : Gnutella (à partir de la version 0.6) [SR04], FastTrack [Har04], eDonkey/eMule [eMu, KB05], Skype [Sky, BS06a, GDJ06].

Il est à noter que les architectures hybrides et les architectures centralisées sont parfois regroupées au sein d'une même classe dite architecture à serveurs.

### *1.3.4 - JXTA*

JXTA [JXTA, Dal03] abréviation de "JuXTApose", par rapport au modèle de référence (Web ou client-serveur), est une plateforme associée à une suite de protocoles libres de droit. Elle a été développée par SUN Microsystems dans le but de supprimer les problèmes de non-interopérabilité et non-portabilité des différentes implémentations d'architectures P2P. Ainsi JXTA permet d'interconnecter n'importe quel système sur n'importe quel réseau, établissant des communications et des échanges au sein de réseaux fixes et mobiles, filaires et non filaires (PC, PDA, etc.), même en présence d'un pare-feu. Par conséquent, l'on a plusieurs points d'accès au réseau (IP, TCP, HTTP, etc.) et les liens sont établis dynamiquement. Les canaux de communications virtuels ainsi formés, et appelés « *pipe* », fonctionnent en mode asynchrone.

## 1.4 - Architectures et protocoles pair-à-pair décentralisés et structurés

La grande majorité des architectures décentralisées structurées est basée sur des tables de hachage distribuées : les DHT (*Distributed Hash Tables*).

Ce paragraphe étudie les DHT et explique les principales caractéristiques de plusieurs modèles où elles sont utilisées.

Il existe toutefois des architectures décentralisées structurées n'utilisant pas de DHT ; par exemple : les *skip graphs* ou graphes à enjambement (cf. paragraphe 1.4.9 page 58).

### *1.4.1 - Les tables de hachage distribuées*

Une fonction $f$ d'un ensemble $E$ vers un ensemble $F$ est dite injective si : $\forall (x,y) \in E^2, (x \neq y) \Rightarrow (f(x) \neq f(y))$ Ceci garantit donc que : $\forall (x,y) \in E^2, (f(x) = f(y)) \Rightarrow (x = y)$

Une fonction de hachage est une fonction injective d'un ensemble de clés vers un ensemble de valeurs, qui à une chaîne d'octets de longueur quelconque associe une valeur unique (de taille fixe).

Une table de hachage représente un tableau de correspondance entre des clés et des valeurs liées par une fonction de hachage, *h*.

Une table de hachage distribuée (ou DHT pour *Distributed Hash Table*) est une table de hachage partagée entre tous les éléments actifs d'un système réparti.

Appliquée à une majorité de réseau P2P, une DHT partitionne un index global de ressources (ou objet) et distribue ces parties d'index sur plusieurs entités différentes, elles-mêmes indexées. Ces entités sont les pairs, et sont aussi considérées comme des ressources (puisque chaque pair met à disposition une partie de ses propres ressources).

Par ailleurs, chaque pair reçoit un identifiant unique, dit *nodeId*. Il en est de même pour chaque ressource, dont l'identifiant est dit *objectId*. Lorsqu'on parle d'une requête, l'*objectId* recherché

est dit *key*. Dans la littérature, on dit souvent, par exemple, « *le pair 3* », au lieu de dire « *le pair d'identifiant (ou de nodeId) 3* ». Il en est de même pour l'identification des objets.

Les *nodeId* et *objectId* sont dans un même espace logique *K* qui correspond à l'index global de la DHT. Chaque pair est responsable d'une partie de cet espace. Cette partie sera dite tout court : la DHT du pair.

Dépendamment de l'implémentation de la DHT, l'espace de hachage correspondant (et donc la DHT) peut être visualisé(e) comme une grille, un cercle (ou anneau), ou une ligne. L'implémentation de la DHT est spécifique à l'algorithme de routage du protocole mis en œuvre. Cet algorithme va permettre de construire la DHT en associant chaque *objectId* au pair responsable de l'objet correspondant.

En d'autres termes, étant donné une fonction de hachage *h* et un pair ayant une adresse IP *@IP* et voulant partager sur le réseau une ressource *r*, on a :

$$h(r) = k \in K \text{ ; où } k \text{ est l'} objectId$$

$$\text{et } h(@IP) = i \in K \text{ ; où } i \text{ le } nodeId$$

La ressource *r*, ou un lien vers cette ressource, est alors stockée sur la DHT du pair d'adresse *@IP* tel que dist(*k, i*) est minimale ; dist(*k, i*) étant la distance entre *k* et *i*, telle que définie dans l'espace logique *K*. Autrement dit, dans la DHT, *r* est indexée par le pair d'identifiant *i*. De même, tout pair d'identifiant *j* quelconque connaît de par sa DHT tous les pairs d'identifiant *m* quelconque, tel que la distance entre *j* et *m* est faible. Si *m* n'est pas « proche » de *k*, le pair d'identifiant *j* a dans sa DHT au moins un pair *n* strictement plus proche de *k*. Ainsi, par itération, *j* trouve *k* et peut récupérer la ressource *r*. [Vie05]

La figure 1.7 donne un exemple de mécanisme de routage dans un réseau *overlay* basé sur une DHT. Le casier jaune représente pratiquement une ligne de la DHT du nœud pointé.

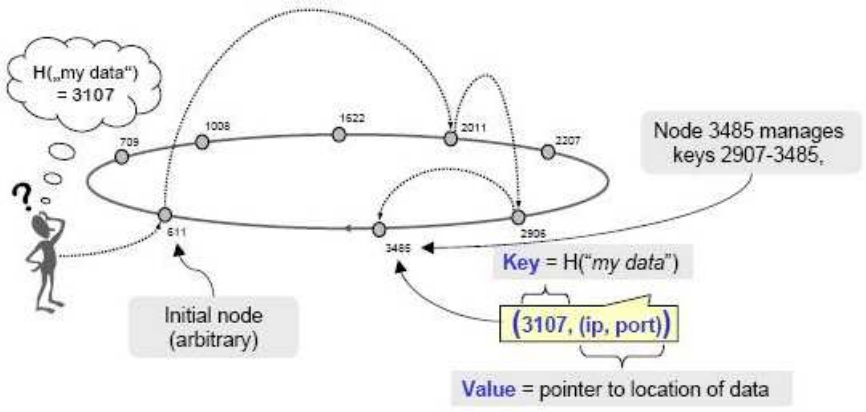


***Figure 1.7 :*** *DHT et mécanisme de routage overlay*[1]

En utilisant les services de sa DHT chaque pair peut donc à tout moment :

- sur la base du *nodeId*, établir une communication directe ou indirecte avec un autre pair ;
- et grâce à l'*objectId*, localiser une ressource.

D'ailleurs, chaque DHT implémente des fonctionnalités de recherche (*lookup*) et de récupération (*retrieval*), assurant deux fonctions principales [Rhe05] :

- une fonction *put(key, data)* qui remplit la DHT ;
- et une fonction *get(key)* qui permet de localiser un objet à partir de son *objectId* (qui est la clé de la requête) en retournant le *nodeId* du pair responsable de l'objet. Ce pair-ci va alors fournir l'objet soit directement soit indirectement en indiquant où il peut être trouvé.

Les DHTs garantissent donc l'unicité de la clé et l'aboutissement des requêtes en un nombre minimal de sauts. Il s'agit là de sauts logiques, par opposition aux sauts physiques que sont les sauts IP. Au pire des cas, la localisation d'un objet nécessitera l'échange d'un nombre de messages logarithmique du nombre de pairs actifs.

[1] **In :** [BS06b]

La particularité des DHTs utilisées dans les architectures P2P est la fonction de hachage, qui est dite consistante (*consistent hashing* [KLL+97]). Elle garantit avec une probabilité élevée une distribution uniforme des ressources sur l'ensemble des nœuds et un déplacement de *O(1/N)* objets lors de l'arrivée ou du départ d'un *N*ième pair du réseau. En effet, la dynamique des pairs induit l'échange de messages et le déplacement d'objets pour préserver la structure et la cohérence de la DHT. Les architectures P2P à base de DHTs représentent alors de faibles coûts de maintenance et d'utilisation.

L'objectif principal des DHTs est le passage à l'échelle. Appliquées aux réseaux P2P, elles garantissent en plus les quatre propriétés suivantes [RM06] :

- *un degré de connexion faible* :
  chaque pair garde en mémoire (dans une liste ou table) un nombre restreint de connexions actives ;
- *un diamètre petit* :
  le nombre maximal de sauts nécessaires pour atteindre n'importe quel autre pair du réseau est minimisé ;
- *un routage "gourmand"* :
  chaque pair calcule tout seul le plus court chemin pour atteindre la destination ;
- *la robustesse* :
  un chemin vers la destination peut être trouvé même en cas de rupture de liens ou disparition de pairs.

Dans la dynamique des réseaux P2P, les pairs vont constamment s'organiser pour former un graphe de communication conforme avec l'implémentation de la DHT et ayant un rapport diamètre/degré optimal.

Les protocoles de routage dans les réseaux P2P qui utilisent des DHTs, et souvent appelés DHTs tout court, ont vu le jour dans les milieux de recherche académique. Les quatre premiers protocoles

sont : Chord [SML+01], Pastry [RD01], CAN (*Content-Addressable Network*) [RFH+01, Rat02] et Tapestry [ZKJ01, ZHS+04]. Depuis, de nombreuses DHTs ont émergés. Parmi les plus connues, citons : Kademlia [MM02], Viceroy [MNR02], Bamboo [RGR+04], et D2B [FG06]. Et la liste continue à s'allonger sans cesse bien qu'assez peu nombreuses sont les DHTs publiées avec une implémentation robuste. Mais Kademlia est d'ores et déjà intégrée dans les deux applications P2P de partage de fichiers, les plus populaires [Loe08, eMu04] : BitTorrent et eMule.

### *1.4.2 - Architecture en anneau*

L'architecture en anneau est la plus simple qui puisse venir à l'esprit et la plus simple à gérer. Une DHT à géométrie circulaire permet la meilleure flexibilité et offre les meilleures performances de résilience et de proximité [GGG+03]. Dans une telle architecture l'espace d'identification est circulaire et la proximité est numérique. Chord [SML+01] en est un exemple. Il constitue aussi l'exemple type du routage P2P avec les DHTs.

Dans Chord, le réseau logique est un anneau orienté de $2^m$ pairs, où *m* est le nombre de bits de la fonction de hachage utilisée (souvent SHA-1 à 160 bits). Chaque pair est lié à un prédécesseur et plusieurs successeurs. Chaque objet est stocké sur le pair le plus proche dont le *nodeId* est supérieur ou égal à son *objectId*.

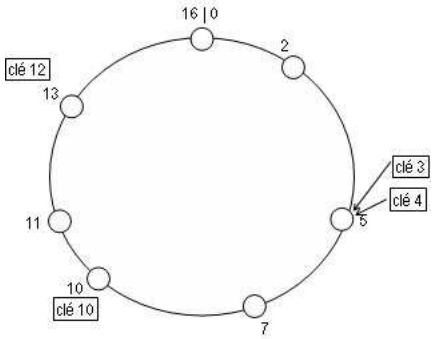


***Figure 1.8 :*** *Un exemple d'anneau Chord*

La figure 1.8 montre la répartition de quatre objets sur les sept pairs actifs d'un réseau avec $m = 4$. Chaque objet y est représenté par le mot 'clé' suivi de l'*objectId*, et chaque pair actif y est représenté par son *nodeId*.

Une réplique de chaque objet est insérée sur chacun des successeurs du pair où il est stocké. La liste des successeurs est de taille fixe. Si tous les successeurs d'un pair disparaissent du réseau simultanément, l'anneau se scinde. Ce qui est cependant assez peu probable, même avec un petit nombre de successeurs par pair. Enfin chaque pair lance périodiquement une fonction de mise à jour de ses listes.

Chaque pair maintient une *finger table* qui joue le rôle de table de routage. Le premier pair de cette table est le premier nœud qui suit sur le cercle, c'est aussi un successeur. La figure 1.9 représente une *finger table* avec les détails de sa construction.

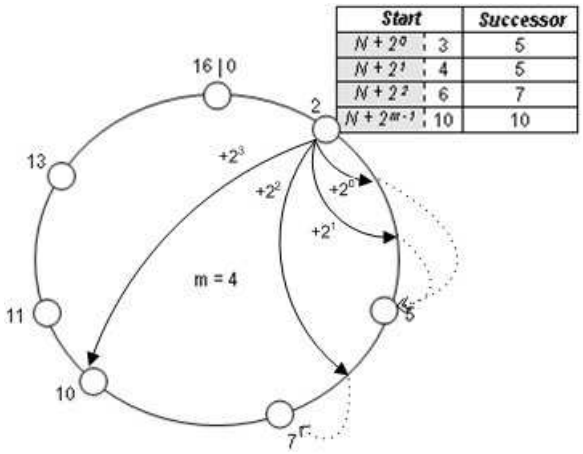


***Figure 1.9 :*** *Un exemple de 'finger table'*

La figure 1.10 représente les sauts effectués par une requête consistant à trouver la clé 12 (l'objet d'identifiant 12) à partir du pair 2. À chaque saut l'on est au moins à mi-distance entre le pair émetteur de la requête et le pair cible, responsable de l'objet recherché. L'opération de recherche dans Chord est donc une fonction logarithmique du nombre total de nœuds pouvant adhérer à l'anneau.

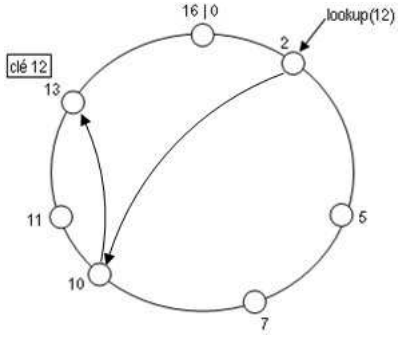


***Figure 1.10 :*** *Un exemple du parcours d'une requête dans Chord*

Les *finger table*s sont aussi consultées pour l'insertion d'un nouveau nœud afin de lui indiquer ses prédécesseur et premier successeur, et donc son emplacement logique dans le réseau. Suite à cela, les *finger table*s de tous les nœuds sont mises à jour. (Préalablement à toute insertion, le nouveau nœud de *nodeId n*, par un mécanisme externe à Chord, prend connaissance d'un pair à proximité de lui et actif dans le réseau P2P sous le *nodeId* $n_1$. *n* envoie alors une demande d'adhésion à $n_1$. L'adhésion se fait à travers la fonction $n.join(n_1)$.)

Pour se retirer, un nœud peut avertir les nœuds présents dans sa *finger table*, afin de faciliter les mises à jour, qui seront à défaut faites à l'expiration d'un temporisateur.

La simplicité de Chord continue à inspirer plusieurs nouvelles DHTs : DKS (*Distributed K-ary Search*) [EAB+02] qui améliore les performances de Chord ; une hiérarchisation d'anneaux Chord [AS04, GGG04] ; Chord on Demand [MJB05] ; etc.

Chord est déployé dans de nombreuses applications, dont :

- CFS (*Collaborative File System*) [DKK+01], un système de fichiers distribués à l'échelle de l'Internet, et où chaque fichier est partagé en blocs ;
- ConChord [ACM+02], une infrastructure distribuée qui utilise CFS et délivre des certificats de sécurité SDSI (*Simple Distributed Security Infrastructure*) ;

– Chord-based DNS [CMM02] qui propose une implémentation P2P du DNS (*Domain Name Service*).

### *1.4.3 - Architecture maillée de Plaxton*

L'algorithme de Plaxton [PRR97] a été développé pour les systèmes statiques distribués, formant une maille quelconque. Il peut donc localiser les objets en utilisant des tables de routage de taille fixe. C'est le premier algorithme applicable aux DHTs.

Les identifiants des nœuds et des objets sont numériques et indépendants de toute sémantique. Chaque nœud est une racine d'un arbre de recouvrement. Le routage en soi est de type *hyper-cube*, c'est-à-dire qu'il se fait par rapprochement successif d'un seul digit à la fois dans l'identifiant numérique jusqu'à atteindre l'objectif. Il s'agit d'un routage analogue au CIDR [RFC4632], le routage inter-domaine sans classe dans les réseaux IP. Il peut être basé soit sur le préfixe (comme dans Pastry [RD01]), soit sur le suffixe (comme dans Tapestry [ZKJ01, ZHS+04]).

#### *1.4.3.1 - Tapestry*

Dans Tapestry, l'espace d'identification est une maille quelconque et la fonction de hachage est fixée à 160 bits. L'identifiant de l'objet est désigné par GUID (*Global Unique IDentifier*) au lieu d'*objectId*. Des répliques de l'objet sont insérées avec des clés différentes le long du chemin menant du pair au sommet de l'arbre de recouvrement. Ce nœud racine identifie l'objet de manière unique. Le pair responsable de l'objet garde des pointeurs vers les pairs supports des répliques.

Le routage *hyper-cube* est basé sur le suffixe et la recherche est récursive. À défaut, le routage se fait vers le pair le plus proche numériquement dans le niveau hiérarchique supérieur ; on parle alors de s*urrogate routing*.

La figure 1.11 illustre les niveaux hiérarchiques dans Tapestry.

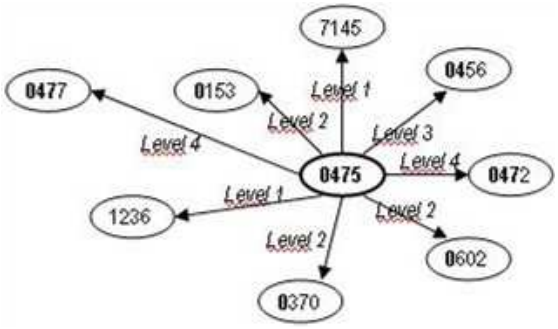


***Figure 1.11 :*** *Exemple des niveaux hiérarchiques dans Tapestry*

La table de routage dans Tapestry contient les pairs les plus proches numériquement. Elle est formée de $b.Log_bN$ lignes et $Log_bN$ colonnes, où $b$ est la base d'identification des clés. Chaque colonne représente un niveau de suffixe commun avec le pair. En fin de colonne, des *back pointers* pointent vers les pairs reconnaissant le pair courant comme un de leurs voisins. Dans chaque ligne, chaque pair pointe vers des pairs voisins de même suffixe. Ces pointeurs sont classés par ordre de proximité physique (délai réseau).

Tapestry a servi de réseau de base à plusieurs applications, dont :

- OceanStore [Oce, KBC+00], une application de stockage massif déployée sur PlanetLab [Pla, PAC+02] ;
- SpamWatch [ZZZ+03], un système collaboratif de filtrage du pourriel.

*1.4.3.2 - Pastry*

Dans Pastry [RD01], l'espace d'identification est circulaire non orienté (à la différence de Chord [SML+01]). Les recherches peuvent donc se faire dans un sens ou dans

l'autre. La fonction de hachage est fixée à 128 bits. Les *nodeId*s sont en base $2^b$, où $b$ vaut souvent 2 ou 4. Ils peuvent aussi résulter d'un hachage de la clé publique du pair. Pastry se base donc sur une proximité numérique, mais prend aussi en considération la proximité réseau pour gérer les arrivées et départs des pairs. Sa méthode de routage hyper-cube se base sur les préfixes (cf. fig. 1.12).

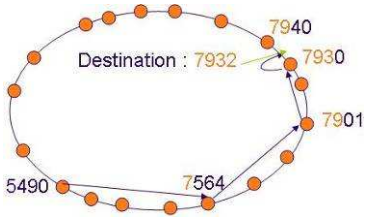


***Figure 1.12 :*** *Exemple de routage dans un réseau Pastry*

Dans Pastry, chaque pair maintient en plus de la table de routage, un *leaf set* et un *neighborhood set*. (cf. fig. 1.13)

*Exemple pour un nodeId 30230312*

| | | | | |
|---|---|---|---|---|
| 30230302 | 30230300 | 30230313 | 30230322 | Leaf set |
| 30230223 | 30230212 | 30230323 | 30230331 | |
| -0-1321312 | -1-1321300 | -2-0311230 | 3 | Table de routage |
| 0 | 3-1-302210 | 3-2-230312 | 3-3-021031 | |
| 30-0-01120 | 30-1-21001 | 2 | 30-3-32112 | |
| 302-0-3213 | 302-1-1320 | 302-2-2203 | 3 | |
| 0 | 3023-1-210 | 3023-2-321 | 3023-3-312 | |
| | 30230-1-32 | 30230-2-12 | 3 | |
| | 1 | 302303-2-2 | | |
| | | 2 | | |
| 32032312 | 01321312 | 32031302 | 11321300 | Neighborhood set |
| 21331201 | 13232010 | 20311230 | 31302210 | |

***Figure 1.13 :*** *Exemple de table de routage dans Pastry*

La table de routage, proprement dite, contient $log_2{}^b N$ lignes. La $i^{ème}$ ligne contient des pairs ayant un préfixe de $i$ digits communs avec le pair en question. Le nombre de colonnes contenant des *nodeId*s est de $2^b$-$1$. Il reste une colonne qui contient un digit du *nodeId* du pair courant et

correspondant au numéro de la colonne. Le format des *nodeId*s dans la table de routage est donc :

*préfixe commun – numéro de la colonne – suite du nodeId*.

Le *leaf set* contient les *L* pairs les plus proches numériquement du pair en question ; *L/2* ayant un *nodeId* inférieur et autant ayant un *nodeId* supérieur. Les répliques des objets sont insérées sur chacun de ces *L* éléments, sans aucun contrôle de la part du pair détenteur de l'objet. Pastry diffère donc de Tapestry dans sa gestion de la proximité et des réplications.

Quant au *neighborhood set*, il contient *M* pairs proches du pair en question, au sens de la proximité réseau. Ils servent au maintient des propriétés locales.

Pour le routage, le pair consulte d'abord son *leaf set*, s'il y trouve la clé, il route directement vers ce pair-là. Sinon, il consulte sa table de routage, proprement dite, pour opérer un routage par saut selon Plaxton. Si à une des étapes, il n'y a pas de pair avec le *nodeId* duquel la clé partage un préfixe d'un digit plus long que celui partagé avec le *nodeId* courant, le routage s'opère vers un pair du *leaf set* qui serait numériquement plus proche de la clé.

Pastry a été avantagé par le développement de FreePastry [Fre] en code source libre et a servi de réseau de base à de nombreuses applications, dont :

- SCRIBE [RKC+01, CDK+02], une application de diffusion multi-groupes, de grande taille chacun ;
- PAST [DR01], un utilitaire de stockage massif ;
- Pastiche [CMN02], un système de sauvegarde qui exploite les capacités disponibles des antémémoires réparties dans le réseau.

### *1.4.4 - Architecture torique*

Dans l'architecture en tore, l'espace d'identification est de dimension *d* à coordonnées cartésiennes. Il est uniformément partagé en zones ; chacune sous la responsabilité d'un pair. Les objets sont stockés en des points de l'espace. Chaque pair est responsable des objets de sa zone L'exemple type d'une DHT avec une structure torique est CAN (*Content-Addressable Network*) [RFH+01, Rat02].

Dans CAN, chaque zone peut être assimilée à un nœud d'un arbre binaire. La proximité des nœuds dans cet arbre est définie par la notion de voisins. Des zones voisines étant celles dont les coordonnées se chevauchent sur *d-1* dimensions et sont contiguës sur la dimension restante. Les pairs responsables de zones voisines sont alors des pairs voisins ; chacun connaissant l'adresse IP et les coordonnées de ses voisins.

Pour rechercher une clé, CAN hache cette clé suivant les différentes dimensions, localise le point résultat et repère le nœud responsable de la zone d'appartenance de ce point-là. Le routage s'effectue alors de proche en proche en passant par tous les voisins jusqu'à arriver au pair cible.

La figure 1.14 illustre un exemple de répartition et de routage dans un réseau CAN de dimension 2.

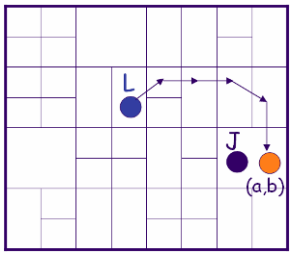


***Figure 1.14 :*** *Exemple de routage dans CAN en dimension 2*

Afin d'améliorer les performances, notamment la latence du routage applicatif, l'espace entier avec sa DHT peut être dupliqué. On parle alors de « réalités » et un même pair occupera des zones différentes d'une *réalité* à l'autre.

### *1.4.5 - Architecture en papillon*

L'architecture est dite papillon, ou *butterfly*, car les connexions établies entre les nœuds du réseau dessinent des formes en papillon. Ce type d'architecture a été initialement étudié pour les systèmes multiprocesseurs SIMD (*Single Instruction Multiple Data*) [Sie79].

Viceroy [MNR02] est un protocole de routage dans une architecture P2P *butterfly* (cf. fig. 1.15). En fait, Viceroy est une amélioration de Chord [SML+01] appliqué à une topologie annulaire à plusieurs niveaux, dite aussi multi-anneaux. De par l'architecture, chaque pair a une connectivité (ou degré) constante égale à 7. En effet, chaque pair maintient :

- *deux pointeurs généraux* : vers le nœud précédent et le nœud suivant sur le même niveau ;
- *deux pointeurs de niveau* : vers le prédécesseur et le successeur sur l'anneau de référence (généralement celui de niveau 1) ;
- *trois pointeurs 'butterfly'* : vers les nœuds à gauche et à droite sur le niveau d'indice supérieur et vers le nœud le plus proche sur le niveau d'indice inférieur. À noter que les indices des niveaux vont croissants vers le bas.

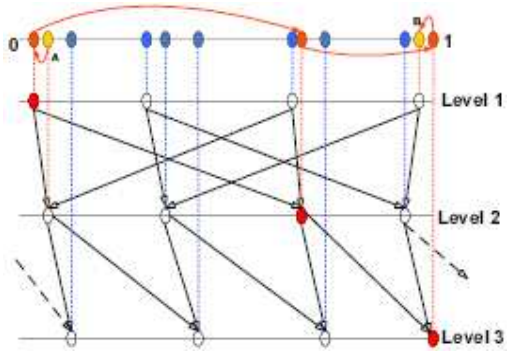


***Figure 1.15 :*** *Illustration simplifiée de l'architecture de Viceroy, sans représentation des liens supérieurs et de niveau*[1]

Avec Viceroy, le routage se fait en commençant par le niveau du nœud qui lance sa requête, puis en remontant vers le niveau

[1] **In :** [LLX+04]

d'indice supérieur à l'aide des pointeurs de niveau, qui permettent par la suite de redescendre aux niveaux d'indices inférieurs, et ce récursivement jusqu'à satisfaction de la requête.

Ulysses [KMX+03] est un autre protocole de routage P2P utilisant une architecture en papillon. Basé sur Viceroy, il définit des liens supplémentaires entre un ensemble de pairs de niveaux différents. Ces liens servent de raccourcis en cas de congestion au niveau de certains pairs, due à la dynamique du réseau. Ils garantissent un système sans congestion et avec un trafic réparti équitablement.

### *1.4.6 - Architecture en arborescence*

L'architecture en arborescence est une topologie plate où les nœuds sont les feuilles d'un arbre logique. Il n'y a pas d'hiérarchisation entre les nœuds. De par l'arborescence, l'espace d'identification est en base 2. Nous avons vu plus haut que CAN [RFH+01, Rat02] peut être transposé en une arborescence (cf. p. 51). Nous aborderons maintenant Kademlia [MM02] qui a déjà été introduit et fait ses preuves dans BitTorrent [Loe08] et dans eMule (sous le nom de Kad [eMu04, SR06]), les applications P2P les plus populaires pour le partage de fichiers.

À l'origine, Kademlia a été pensé comme une correction de certaines limites de Chord [SML+01]. L'espace d'identification de Kademlia est donc unidirectionnel, et la clé de hachage est sur 160 bits. Comme Pastry [RD01], Kademlia, utilise le routage hyper-cube basé sur le préfixe. Mais Kademlia a introduit une nouvelle métrique dans les réseaux P2P : le choix exclusif bit-à-bit (XOR) entre *n*, le *nodeId* courant, et *k*, la clé de l'objet à localiser. Plus *n* et *k* ont des bits en commun, plus *n XOR k* devient petit. À l'annulation, l'objet est localisé. Les requêtes contiennent le *nodeId* de l'initiateur. La figure 1.16 illustre un exemple de localisation de la clé1110, à partir du pair 0011.

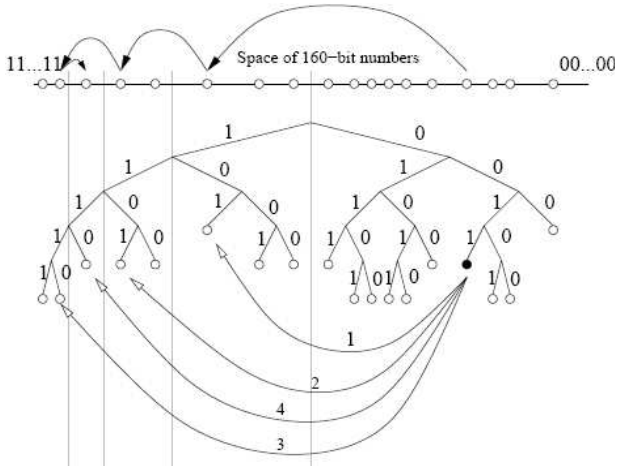


***Figure 1.16 :*** *Localisation dans Kademlia de 1110 à partir de 0011*[1]

Dans Kademlia, les pairs communiquent grâce à UDP (*User Datagram Protocol*) et chaque pair maintient des listes finie de triplets {adresse IP, numéro de port UDP, *nodeId*} relatifs aux derniers pairs contactés. Chaque liste est régie par le mode d'ordonnancement LRU (*Least Recently Used*) qui avance en premier les éléments les moins utilisés. Kademlia avantage ainsi les pairs les plus anciens dans le réseau. Ceci assure une résistance active à l'intrusion de nœuds malicieux et à certaines attaques de type déni de service (DoS – *Denial of Service*).

### *1.4.7 - Graphes de 'de Bruijn'*

Les graphes de *de Bruijn* portent le nom du mathématicien qui les a introduits [deB46]. Il s'agit de graphes orientés $B(k,l)$ dont les sommets sont toutes les suites possibles de longueur $l$ formées des symboles de l'alphabet $A=\{0, 1, \ldots, k-1\}$ et dont les arcs joignent deux sommets $f$ et $g$, tel que $f=xh$ et $g=hy$, où $x$ et $y$ sont des symboles de $A$ et $h$ une suite quelconque de $l-1$ symboles de $A$.

Chaque nœud d'un graphe $B(k,l)$ de *de Bruijn* connaît $k$ autres nœuds, avec $k$ variable selon le degré de robustesse désiré. En passant d'un nœud à un autre, on parle de routage par décalage. Le

[1] **In :** [MM02]

décalage peut se faire vers la gauche ou la droite, selon l'orientation des arcs ; il est défini par l'algorithme du protocole.

La figure 1.17 illustre le graphe *B*(2,3) de *de Bruijn*.

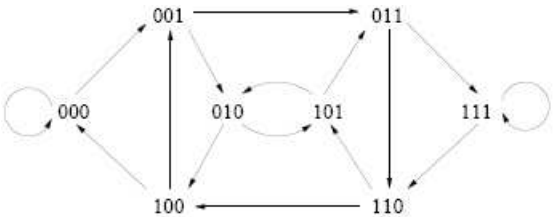


***Figure 1.17 :*** *Exemple d'un graphe* B(2,3) *de 'de Bruijn'*[1]

Ces dernières années plusieurs systèmes P2P basés sur les graphes de '*de Bruijn*' et utilisant les DHTs ont vu le jour. Nous citons :

- Koorde [KK03] qui étend Chord [SML+01] pour atteindre les performances d'un graphe *B*(2,*l*) de *de Bruijn*. C'est l'une des DHTs qui représente le taux de latence le plus faible, en l'absence de congestion ;
- Broose [GV04], qui adapte Kademlia [MM02] à la topologie de *de Bruijn* ;
- D2B [FG06] qui adapte CAN [RFH+01, Rat02] aux graphes de *de Bruijn* ;
- Optimal Diameter Routing Infrastructure (ODRI) [LKR+03] pour les réseaux à degré constant.

### *1.4.8 - Analyse comparative des principales architectures avec DHT*

Les performances des protocoles de routage utilisant des DHTs et dédiés à des architectures P2P décentralisées et structurées sont évaluées par le degré et le diamètre du protocole. Le tableau 1.1 donne les performances des principaux protocoles présentés, ainsi que celles intéressantes d'autres protocoles dont il a été fait mention plus haut dans ce chapitre.

[1] **In :** [FG06]

***Tableau 1.1 :*** *Comparaison des degré et diamètre de quelques protocoles de routage P2P utilisant des DHTs*

| ***Protocole*** | ***Degré*** | ***Diamètre*** |
|---|---|---|
| *Chord et Kademlia* | $O(\log N)$ | $O(\log N)$ |
| *DKS* | $O((k-1).\log_k N)$ | $O(\log_k N)$ |
| *Tapestry et Pastry* | $O(\log_2{}^b N)$ | $O(\log_2{}^b N)$ |
| *CAN* | $O(d)$ | $O(dN^{1/d})$ |
| *Viceroy* | $O(1)$ | $O(\log N)$ |
| *Ulysses* | $O(k)$ | $O(\log N / \log k)$ |
| *Les graphes de 'de Bruijn'* | $k$ (variable) | $O(\log_k N)$ |
| *Koorde et Ulysses* (avec $k \rightarrow \log N$) | $O(\log N)$ | $O(\log N / (\log (\log N)))$ |

Ci-suivent quelques remarques d'analyse comparative.

- Malgré des performances bien intéressantes, Ulysses [KMX+03] reste lourd à mettre en œuvre, notamment à grande échelle, et ce, vu les nombreux liens croisés à gérer.
- Les architectures en anneau et celles utilisant la métrique du choix exclusif (XOR) permettent la meilleure flexibilité quant à la disposition des pairs voisins et au choix de routes alternatives (résilience) [GGG+03].
- Les graphes de '*de Bruijn*' sont plus fiables qu'une architecture en anneau simple ou une architecture torique, mais les routes alternatives sont indépendantes entre elles [LKR+03]. Ces graphes présentent d'ailleurs une connectivité intéressante, mais leur structure est assez rigide : pour une distribution aléatoire des clés, ils ne peuvent pas assurer à la fois un bon équilibrage de charge et une recherche efficace des objets [DGA04].

#### *1.4.8.1 - Limites des tables de hachage distribuées*

Les DHTs garantissent aux architectures P2P le passage à l'échelle d'un routage performant (évalué par le diamètre), tout en maintenant pour chaque pair une connectivité raisonnable (évaluée par le degré). Cependant, ces structures distribuées établissent certaines limites aux systèmes P2P qui les utilisent.

- La répartition des éléments dans l'espace d'identification d'une DHT dépend à tout moment du nombre total de pairs présents dans le système. La dynamique continue des pairs amène donc un changement permanent dans la répartition et l'affectation des objets.
- Les DHTs ne peuvent pas traiter des requêtes complexes de type base de données ou intervalle. En effet, la fonction de hachage ne respecte ni la similitude ni l'ordonnancement des clés.
- Les identifiants référencés dans les DHTs (*nodeId*s et *objectId*s) n'ont aucune sémantique. En fait, la fonction de hachage broie toute sémantique et n'en crée aucune.
- Les DHTs font abstraction du réseau *underlay*, et leurs performances sont mesurées en sauts logiques. Elles supposent donc tous les pairs issus d'un même domaine administratif et les liens physiques les reliant homogènes et équitablement chargés.
- Les DHTs font aussi abstraction des différentes ressources réseaux disponibles (e.g. processeurs, capacités de stockage, etc.) et de leurs répartitions réelles (au niveau de chaque pair).
- Enfin, comme tout réseau, ceux à base de DHTs sont perméables aux différentes attaques et problèmes de sécurité (e.g. pair malicieux, faux-contenu si les objets

sont des fichiers, surcharge ciblée, messages parasites, etc.) [SM02].

### *1.4.9 - Graphes à enjambement*

Les graphes à enjambement, ou *skip graph*, sont des architectures P2P décentralisées et structurées n'utilisant pas de DHT. Ils ont été développés sur base des listes à enjambement, ou *skip list*. Dans la suite sont utilisés les termes anglais : *skip list* et *skip graph*.

#### *1.4.9.1 - Skip lists*

Les *skip list*s ont été inventées en 1990 par W. Pugh [Pug90]. Il s'agit d'une structure de données probabiliste, organisée en plusieurs niveaux parallèles de listes chaînées. Le niveau le plus bas est une liste chaînée standard ; les niveaux supérieurs sont des listes dérivées de la liste de base suivant une certaine probabilité, permettant ainsi un parcours rapide de la liste de base. (cf. fig. 1.18)

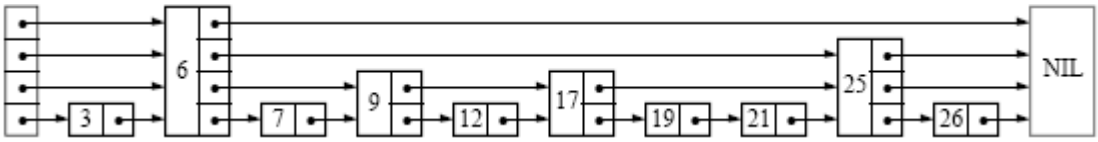


***Figure 1.18 :*** *Exemple d'une 'skip list'*[1]

Le principal atout des *skip list*s est qu'elles ne nécessitent aucune connaissance préalable du nombre total de ses éléments (d'ailleurs indéfini). L'insertion et la suppression se passent comme dans une liste chaînée, sauf que les éléments présents à plus d'un niveau doivent être insérés et supprimés de tous ses niveaux. La recherche d'une clé est aussi similaire au mécanisme de recherche dans une liste chaînée. Elle s'opère successivement dans

[1] **In :** [Pug90]

chacune des listes, en commençant par la liste supérieure en premier. Une liste inférieure n'est parcourue que si la requête n'a pas été satisfaite dans les listes supérieures. La figure 1.19 illustre les modalités de recherche pour l'insertion d'un nouvel élément dans la *skip list*.

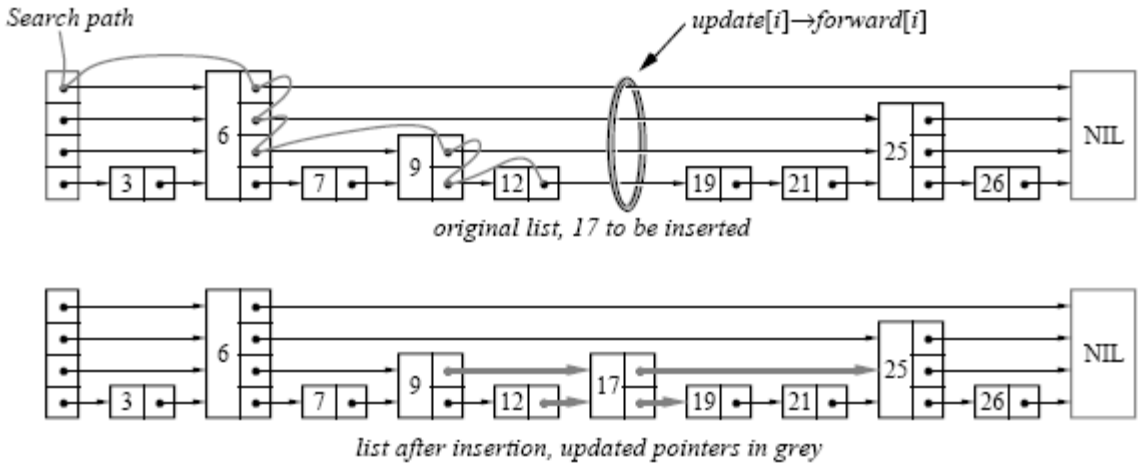


***Figure 1.19 :*** *Illustration du parcours d'une* skip list *et de l'insertion d'un élément dans celle-ci*[1]

*1.4.9.2 -* <u>*Skip graphs*</u>

À la différence d'une *skip list*, un *skip graph* présente plusieurs listes par niveau afin d'assurer une certaine redondance et chaque nœud est présent à tous les niveaux.

Dans les *skip graph*, les nœuds sont identifiés par une chaîne aléatoire de bits, en base deux, et de longueur *l-1* ; *l* étant le nombre de niveaux. Chaque nœud indexe ses propres objets et choisit ses voisins en fonction des identifiants des objets qu'ils stockent.

Un vecteur d'appartenance, ou *membership vector*, est lié à chaque nœud pour lui permettre de reconnaître les listes chaînées auxquelles il appartient. Chaque liste chaînée est étiquetée par un mot de taille finie. Le vecteur

[1] **<u>In</u> :** [Pug90]

d'appartenance est un mot aléatoire de longueur indéfinie mais composé à partir d'un certain alphabet de taille finie. Un nœud appartient à toutes les listes chaînées dont l'étiquette est un préfixe de son vecteur d'appartenance.

La figure 1.20 illustre un exemple de *skip graph* à trois niveaux avec six nœuds.

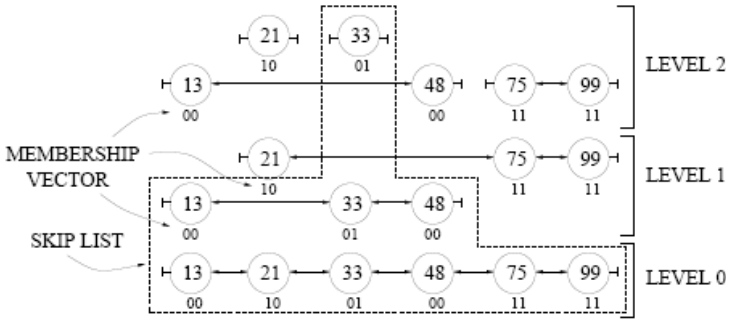


***Figure 1.20 :*** *Un 'skip graph' de six nœuds à trois niveaux*[1]

*1.4.9.3 -* <u>*SkipNet*</u>

Parallèlement aux *skip graph* [AS03], l'idée d'utiliser les *skip lists* pour le routage P2P structuré a abouti aussi à SkipNet [HJS+03]. SkipNet se présente comme un cas d'usage des *skip graph*, garantissant la localité des chemins et des contenus. La figure 1.21 illustre une infrastructure SkipNet à huit nœuds.

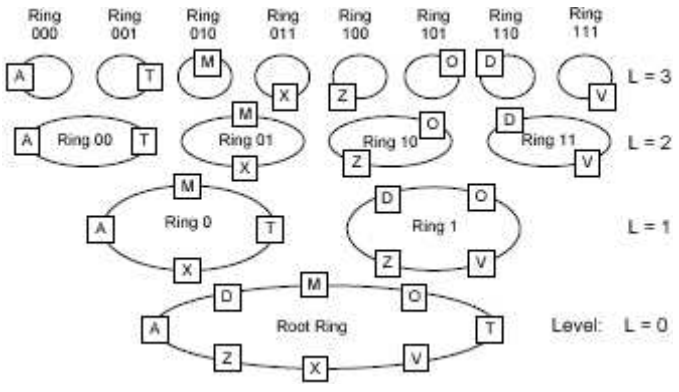


***Figure 1.21 :*** *Une infrastructure SkipNet à huit nœuds*[2]

[1] **<u>In</u> :** [AS03]
[2] **<u>In</u> :** [HJS+03]

Dans SkipNet les nœuds ont deux identifiants : un numérique (*numericId*) et un nominal (*nameId*). La recherche d'objets peut alors se faire sur la base soit de l'identifiant nominal, soit de l'identifiant numérique. La recherche en mode nominal est une recherche par domaine (comme la recherche DNS). Un exemple clair est donné dans [RM04] : pour trouver le document *docname* se trouvant sur le nœud *user.compagny.com*, la requête parcourt les listes à la recherche du préfixe *com.company.user*. Quant à la recherche en mode numérique, elle se fait en remontant les niveaux à partir du niveau le plus bas, de sorte à avancer d'un digit par niveau jusqu'à destination. Dans les deux cas, à défaut de pouvoir avancer, le routage vers l'élément suivant se fait de façon aléatoire mais en se rapprochant de la destination.

#### *1.4.9.4 - Analyse des graphes à enjambement*

Dans un *skip graph*, les objets sont référencés par leurs identifiants logiques, sans passer par le service d'une fonction de hachage. Similitudes et ordonnancements des clés sont donc conservés. Les requêtes complexes de type intervalle deviennent alors possibles, sans qu'elles soient fractionnées ou répétées. Elles s'exécutent dans une sous-liste.

L'absence de fonction de hachage fait que chaque pair est seul responsable de ses objets. Ceci accroît la sécurité du système et diminue le temps de latence entre la localisation d'un objet et sa récupération. La gestion des objets est aussi simplifiée et leur emplacement plus flexible que dans une DHT.

La gestion du système est cependant moins équitablement répartie entre les pairs. En fait, la charge de

chaque pair est fonction du nombre d'objets qu'il partage. Chaque pair physique est représenté par autant de pairs logiques que d'objets qu'il met en partage, et avec *k* clés, il y a *O*(*log k*) niveaux dans le réseau. Le degré d'un pair est donc aussi en *O*(*log N*). Chaque nœud physique gère donc un nombre de liens différent et bien plus important que dans une DHT. L'importance des trafics de maintenance et de mise à jour est tout autant conséquente. Le diamètre reste pourtant en *O*(*log N*).

Comme les DHTs, les *skip graph*s font abstraction des ressources réseaux disponibles. Quant à la localité structurelle du réseau *underlay* (e.g. domaines administratifs), elle est conservée dans SkipNet. Aussi SkipNet, permet de par l'identifiant nominal de garder une sémantique, pour les objets uniquement.

## CONCLUSION DU CHAPITRE –

*Ce chapitre, après une présentation des réseaux P2P et de leurs différentes classes d'architecture, s'est consacré aux principaux protocoles P2P. Il s'est particulièrement étendu sur ceux à base de DHT et s'est terminé par ceux utilisant les graphes à enjambement. Les deux types de systèmes ont été analysés : il en ressort que les derniers pallient certaines limites inhérentes aux premiers, mais non sans introduire des surcharges. Les DHTs ont toutefois déjà fait leurs preuves dans les applications P2P les plus populaires. D'où leur attractivité pour être améliorées en vue de meilleures performances.*

# CHAPITRE 2

## *SPÉCIFICATION DES EXIGENCES POUR UN ROUTAGE PAIR-À-PAIR EFFICACE*

*Ce chapitre examine notre problématique de recherche. Il spécifie les exigences pour un routage efficace et explique la nécessité des systèmes P2P d'être sensibles à la topologie du réseau sous-jacent. Il présente aussi les techniques de génération de l'information de proximité. Le terme 'proximité' désigne une distance proche dans le réseau sous-jacent, en termes de sauts IP.*

### 2.1 - Le routage efficace

Les systèmes P2P assurent un passage à l'échelle et supposent une communauté de pairs fortement dynamique et assez hétérogène (notamment en termes de capacités réseaux). Mais leurs performances globales dépendent largement des performances du protocole de routage mis en œuvre en arrière-plan.

Pour être efficace, un protocole de routage P2P doit pouvoir :

- être consistant, c.-à-d. garantir qu'un message relatif à une clé parvienne au nœud responsable de cette clé (et pas à un autre nœud) ;
- rester robuste même en cas de forte dynamique des pairs :
  - être fiable en toutes conditions réseaux (c.-à-d. garantir qu'aucun message ne se perde) ;
  - être stable (c.-à-d. garantir la mise à jour rapide des données du système, avec un minimum de messages à échanger) ;
  - garantir une bonne tolérance aux pannes (c.-à-d. qu'un chemin vers la destination soit trouvé même en cas de rupture de liens ou de disparition imprévue de pairs (e.g. panne, attaque, etc.)) ;

  - ne pas être affecté par les pairs peu fiables partageant des ressources ;
- être efficient :
  - assurer une convergence rapide (c.-à-d. garantir un temps de réponse court avec un temps de latence minimal) ;
  - transmettre les messages à moindre coût (le coût total est le nombre de sauts, chacun éventuellement combiné à son coût individuel) ;
  - optimiser voire minimiser l'usage des ressources du système ;
- garantir le passage à l'échelle : quand le nombre de processeurs actifs augmente, le système doit :
  - rester robuste et efficient (comme décrit ci-haut) ;
  - conserver des tables de routage de taille raisonnable ;
- assurer un bon équilibrage de charge (des pairs et des objets) dans l'espace d'identification *overlay*, ainsi qu'un bon équilibrage des différents flux de trafic dans le réseau ;
- s'adapter au réseau *underlay* (critère détaillé au paragraphe suivant 2.2) ;
- respecter la philosophie du pair-à-pair et viser le symétrique, notamment dans le cas des architectures hybrides. En ce sens qu'un super-pair ne doit pas garder un monopole indéfini sur les autres pairs de son clan et chaque nœud doit pouvoir être racine ;
- offrir la possibilité de sécuriser le système (les infrastructures, les applications et les pairs) pour :
  - prévenir et pallier les comportements malicieux (e.g. pair malicieux, faux-contenu si les objets sont des fichiers, surcharge ciblée, messages parasites, etc.) ;
  - éviter qu'un comportement malicieux ne puisse corrompre l'état ou le fonctionnement du système ;
  - pouvoir garder une traçabilité légale et protectrice des mouvements du système ;
- permettre le cloisonnement des ressources et garantir la localité du contenu et du chemin, afin de :
  - améliorer les performances (notamment les temps de réponse) ;

- renforcer la sécurité du système ;
- permettre un éventuel contrôle administratif pour la gestion des ressources et le contrôle d'accès.

## 2.2 - Adéquation du routage pair-à-pair au réseau sous-jacent

Nous avons vu au chapitre précédent que le routage dans les systèmes P2P dépend principalement de l'architecture du système, mais aussi du protocole qui lui est relatif. En tout cas, pour un même réseau, les topologies *overlay* et *underlay* sont souvent bien différentes :

– la topologie au niveau *overlay* reflète la disposition des liens entre les pairs ;

– et la topologie au niveau sous-jacent considère plutôt les distances « physiques » entre les pairs. Ces distances, dans un réseau IP, sont généralement mesurées en termes de RTT (*Round Trip Time*), qui estime le temps d'aller-retour d'un paquet entre deux nœuds.

Du coup, un saut entre deux pairs voisins au niveau *overlay* ne signifie pas nécessairement que les deux pairs sont proches au niveau *underlay* (cf. fig. 2.1). Ceci est d'autant plus vrai qu'un réseau P2P ne prend pas naturellement en compte les caractéristiques du réseau *underlay*.

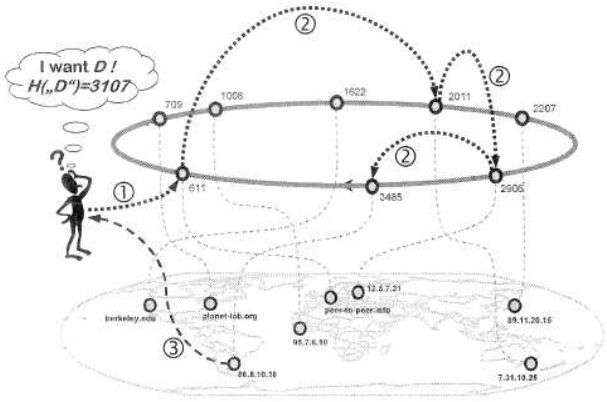


***Figure 2.1 :*** *Routage P2P utilisant une DHT au dessus de l'Internet*[1]

[1] **In :** [WGR05]

La figure 2.1 illustre le routage dans un système P2P au dessus de l'Internet. Elle représente plus particulièrement le cas d'une DHT. D'ailleurs, comme précisé et justifié dès l'introduction, nous nous focalisons dans la suite uniquement sur les réseaux P2P utilisant des DHTs. Mais, comme déjà relevé, les DHTs ignorent aussi la structure du réseau *underlay* (cf. paragraphe 1.4.8.1 page 57). Chaque saut dans la DHT est alors susceptible de générer plusieurs sauts dans le réseau IP. Et les DHTs ne prennent pas en compte la répartition hétérogène des ressources réseaux et la charge différente des liens physiques entre les pairs. Il en résulte que le chemin IP parcouru en conséquence d'un chemin P2P est loin d'être optimal.

Par ailleurs, le réseau Internet est composé de dizaines de milliers de systèmes autonomes (*AS – Autonomous System*) appartenant à des entités administratives différentes. Ces entités peuvent être des opérateurs de réseaux (ou *FAI* pour *Fournisseur d'Accès à Internet* – nous utiliserons le terme anglais *ISP* pour *Internet Service Provider*), des organismes quelconques, des compagnies, des universités, des administrations, etc. Au sein d'une même entité, un ou plusieurs ASs peuvent être regroupés en des domaines autonomes de routage (*ARD – Autonomous Routing Domain*). Mais chaque entité a des politiques de routage propres à chaque ARD. Les ARDs sont donc reliés entre eux suivant des accords économiques signés entre les entités qui les gèrent [Meu07]. Et les informations de routage sont échangées entre les entités via un protocole inter-passerelle extérieur (e.g. BGP [RFC1771]).

Un réseau P2P interconnecte donc des pairs de différents AS, alors que les requêtes et messages P2P sont distribués de façon autonome faisant abstraction de l'infrastructure sous-jacente. Le trafic conséquent au niveau sous-jacent traverse alors les frontières d'un AS sans le savoir, faisant même parfois plusieurs allers-retours [KRP05]. Ceci induit une redondance, des temps de latence, et une surcharge des liens IP, augmentant ainsi leur risque de congestion.

### *2.2.1 - Corrélation entre le flux pair-à-pair et le réseau Internet*

Les techniques d'ingénierie de trafic consistent en la prévision des moyens d'écoulement du trafic qui transite par un réseau. Celles actuellement utilisées par les ISPs sont inadéquates pour coopérer avec les services émergeants des réseaux *overlay*s, car ces réseaux-ci ne suivent pas les politiques de gestion et de routage des ISPs [KTC+04]. De plus, lorsque des pannes surviennent dans le système, un manque de coordination entre les deux réseaux *overlay* et *underlay* peut mener à une dégradation des performances des trafics à chacun des niveaux [KTC+04].

Des études sur l'interaction entre le routage *overlay* et l'ingénierie de trafic au sein d'un AS ont montré que le comportement individualiste de l'*overlay* peut causer une augmentation importante du coût de fonctionnement de tout le système [LZG+06]. Mais la mise au point des paramètres de routage au niveau *underlay* améliore les performances de l'*overlay*, notamment pour le réacheminement des messages en cas de rupture de liens [SA06].

Par conséquent, il doit y avoir coopération de part et d'autre entre le réseau pair-à-pair et le réseau *underlay*. Mais les réseaux *overlay*s les plus populaires sont les réseaux P2P [Fer06] (cf. fig. 2 p. 21). D'ailleurs, une bonne coopération sera perçue positivement à la fois au niveau financier par l'ISP, et au niveau de la qualité perçue par l'usager du réseau P2P.

Du côté de l'ISP, les accords inter-opérateurs définissent un type de contrat d'interconnexion, son prix et ses modalités [Meu07]. Ces accords sont négociés en fonction de plusieurs paramètres et prennent en compte les pré-requis de chaque opérateur en termes de politiques de routage et de sécurité.

Parmi les paramètres considérés, nous pouvons citer : les différents points géographiques d'interconnexion ; les capacités des liaisons (la bande passante maximale) ; les services proposés, leurs

qualités et leurs popularités ; les volumes échangés par les utilisateurs (les différentes destinations générant le plus de trafic en entrant et au sortant de chaque ARD) ; la taille estimée de l'opérateur (en nombre de clients) ; etc.

La relation économique d'interconnexion de deux opérateurs est soit de type *transit*, soit de type *peering*.

- Dans une relation de type *transit*, un opérateur va soit vendre ses capacités de réseaux, soit acheter les capacités d'un autre réseau. Entre deux ISPs, l'un se trouve alors réseau client, et l'autre réseau fournisseur. Pour chaque liaison, l'ARD client paye l'opérateur fournisseur conformément aux clauses du contrat signé en commun.
- Dans une relation de type *peering*, les deux opérateurs établissent un libre échange entre leurs clients. Chacun va partager gratuitement l'accès vers la totalité ou un sous-ensemble géographique de ses clients, mais sans garantie particulière.

Une relation de *peering* entre deux ARDs est convenue parfois pour des raisons de performance. Mais généralement, elle nécessite que les deux ARDs soient équivalents en volume de trafic (il s'agit d'un rapport de taille et de l'importance du contenu fourni) et que chacun dispose d'assez de clients pour être rémunéré correctement. Or, toutes ces conditions sont rarement réunies et les relations de *transit* sont majoritaires entre les ISPs.

Avec la dynamique et l'évolution des réseaux, des services, des contenus, mais aussi des politiques de routage et de sécurité spécifiques à chaque ARD, les ISPs sont amenés à (re-)définir et négocier en permanence leurs accords économiques. De plus les flux P2P ne les favorisent pas, vu l'importance variable de la part de bande passante consommée par ces flux, ainsi que la dynamique irrégulière des communautés P2P.

Par ailleurs, les allers-retours fréquents du trafic P2P entre un AS et un ou plusieurs autres [KRP05], combinés aux accords de

*peering* et de *transit*, amènent certains ISPs selon le cas à avoir leurs ASs servir de transit pour d'autres ISPs sans pouvoir rien y gagner financièrement, ou au contraire payer pour un trafic qui ne bénéficie au bout du chemin à aucun de leurs propres abonnés, mais à ceux de l'ISP fournisseur.

Par conséquent, définir et concevoir des systèmes P2P sensibles à la structure sous-jacente, qui favorisent les échanges en local et en adéquation avec le contexte, et garantissent une récupération des objets suivant ces mêmes critères, représente un intérêt réel pour les ISPs et autres opérateurs de réseaux.

Du côté de l'usager, de tels systèmes sont tout aussi intéressants, non seulement au niveau de l'amélioration des délais de réponse et de la diminution des temps de latence, mais aussi au niveau de la qualité de service perçue. En effet, la vidéo arrive en tête des types de fichiers les plus échangés au sein des différentes communautés P2P [SM07]. Et comparé à la majorité des pages web, un fichier vidéo est bien plus volumineux et consomme plus de bande passante. Or les volumes échangés et le coût de la bande passante entrent en jeu dans le coût financier d'une relation *transit* entre ASs d'opérateurs différents. Ce qui implique actuellement une transmission vidéo de moindre qualité (à défaut d'accords plus onéreux, d'autant plus que le trafic P2P consomme rapidement toute bande passante augmentée).

## 2.3 - Techniques de génération des informations de proximité

Concevoir un système P2P où le routage P2P sera en adéquation avec la structure du réseau *underlay* et les intérêts des opérateurs qui les gèrent nécessite d'exploiter efficacement les informations de proximité des pairs dans le réseau *underlay*. Ceci exige donc en premier de générer ces informations-là. Plusieurs techniques sont possibles. Les premiers travaux sur cette problématique en ont distingué trois [XTZ02] :

- *expanding ring search* est une technique d'inondation du réseau pour la mesure des RTTs entre un nœud et tous les autres dans un rayon prédéfini en nombre de sauts IP ;
- une amélioration *heuristique* consiste à ce que chaque nœud mesure les RTTs qui le sépare de ses voisins à proximité uniquement ;
- *landmark clustering* est une technique qui consiste à repérer des agglomérats de nœuds qui seront représentés par un des leurs. Les distances sont alors mesurées par rapport à ces nœuds-repères. Intuitivement, deux nœuds ayant des distances sensiblement pareilles à une même liste de nœuds-repères sont considérés à proximité physiquement l'un de l'autre. Cette technique a été utilisée pour la création d'une version de CAN [RFH+01, Rat02] sensible à la topologie [RHK+02].

Vivaldi [CDK+03] s'est passé des nœuds-repères en attribuant des coordonnées virtuelles à chaque nœud du réseau Internet. La distance euclidienne entre les coordonnées de deux nœuds prédit le temps de latence entre eux.

La majorité des travaux sur l'estimation des informations de proximité se focalise aussi sur la prédiction de la proximité réseau par le temps de latence. Mais pour nombre d'applications P2P (e.g. la vidéo), le débit est souvent une métrique de qualité plus importante que la latence.

Le service *iPlane* [MIP+06] vise alors à générer et maintenir un « atlas » de l'Internet en utilisant des mesures actives contenant des informations sur le temps de latence, la bande-passante disponible, la capacité des liens et le taux de perte entre deux nœuds quelconques du réseau Internet. Cependant, l'estimation de la bande-passante disponible et du taux de perte, à partir des hôtes terminaux, risque d'être quelque peu obscurcie par les goulots d'étranglement du dernier kilomètre.

La notion de tomographie [CCL+04] des réseaux résume les techniques de vérification active du réseau et de surveillance passive du trafic permettant de déduire des informations concernant la topologie du réseau et les caractéristiques des différents niveaux de liens.

Une fois l'information de proximité acquise, il faut l'exploiter et pouvoir l'utiliser lors de l'adhésion d'un nouveau pair au réseau P2P. Le chapitre suivant présente les solutions existantes pour l'exploitation de cette information et la construction de système sensible à la topologie sous-jacente, pour des réseaux P2P utilisant une DHT.

Il est à noter que dans les réseaux non structurés, il est plus aisé de se servir de la proximité physique, du moment où le réseau *overlay* n'impose pas de contraintes structurelles et que le routage P2P s'y fait par inondation. Cependant, la facilité à s'ajuster à la topologie physique, qu'ont les réseaux non structurés par rapport aux réseaux structurés, risque de compromettre la facilité d'accès aux contenus disponibles dans le réseau P2P.

Enfin, la sensibilisation des réseaux *overlay*s à la topologie physique sous-jacente peut être modélisée, impliquant une métrique mathématique pour le degré de correspondance des deux réseaux [RH07]. Le modèle est basé sur un problème d'optimisation. Il s'agit d'un problème NP-complexe. Ceci veut dire que l'on ne peut résoudre ce problème par une méthode déterministe en temps convenable (*N* pour *Non-determinist*) mais que des solutions, dont on peut prouver la validité, existent en temps polynomial (*P* pour *Polynomial*).

## CONCLUSION DU CHAPITRE –

*Ce chapitre a examiné notre problématique de recherche. Commençant par une spécification des exigences pour un routage P2P efficace indépendamment de tout type d'architecture, il s'est poursuivi par une élucidation de la nécessité de ces systèmes d'être sensibles à la topologie du réseau sous-jacent. Ceci a été particulièrement justifié dans le cadre des DHTs.*

*La nécessité élucidée requérant donc une connaissance de l'information de proximité sous-jacente, le chapitre s'est terminé par une présentation analytique de différentes techniques de génération d'une telle information. Le chapitre suivant abordera différentes possibilités d'exploiter cette information dans le cadre des réseaux P2P utilisant des DHTs.*

# CHAPITRE 3

## *ANALYSE DES SOLUTIONS EXISTANTES POUR UN ROUTAGE PAIR-À-PAIR EFFICACE*

*Ce chapitre analyse les différentes solutions existantes à base de DHT pour la prise en compte du réseau sous-jacent.*

*Le terme 'proximité' désigne une distance proche dans le réseau sous-jacent, en termes de sauts IP.*

### 3.1 - Techniques d'exploitation de la proximité

Les premières DHTs ont été développées indépendamment de la topologie sous-jacente. Toutefois, Pastry [RD01] utilise certaines heuristiques pour explorer la proximité des pairs dans le réseau sous-jacent afin de pouvoir construire ses tables de routage *overlay* (notamment le *neighborhood set*). Cependant les impacts sur le routage P2P sont assez peu importants puisqu'une décision de router vers un pair de cette table de proximité n'est prise qu'en dernier recours face à un choix de nœuds. La localité dans Pastry n'est donc pas exploitée pour une stratégie de prise en compte du réseau sous-jacent.

Les techniques existantes qui permettent d'exploiter les informations de proximité dans le réseau sous-jacent sont classées en trois catégories [CDH+02] :

- le routage de proximité (*proximity routing*) ;
- l'agencement géographique, qui sous-entend une affectation du *nodeId* en fonction de la topologie (*topology-based nodeId assignment*) ;
- et le choix du voisin à proximité (*proximity neighbor selection* ou *PNS*) [RSS02].

### *3.1.1 - Routage de proximité*

Le routage de proximité s'applique à un réseau *overlay* construit indépendamment du réseau sous-jacent. Durant le mécanisme de routage, si un pair a l'opportunité de choisir parmi *k* prochains sauts possibles, il choisit de router le message vers le nœud qui, parmi ces *k* nœuds, est le plus proche de lui dans le réseau sous-jacent. Une alternative serait de choisir de router vers le nœud qui représente le meilleur compromis entre la proximité et la progression dans l'espace d'identification *overlay*. En tout cas, les performances globales de cette technique dépendent de *k*, qui est proportionnel à la taille de la table de routage. Par ailleurs, des petits sauts risquent d'être contrecarrés par l'augmentation du nombre de sauts.

### *3.1.2 - Agencement géographique*

La technique de l'agencement géographique vise à reporter l'espace d'identification *overlay* sur la topologie sous-jacente, et biaise l'attribution des *nodeId*s. Ceci entraîne forcément une diminution des délais, mais les *nodeId*s ne sont plus uniformément distribués dans l'espace d'identification *overlay*. D'où des problèmes d'équilibrage de charge. Les nœuds voisins sont aussi susceptibles de subir des pannes ou attaques corrélées, ce qui affaiblit la robustesse et la sécurité du système. Malgré ces revers, cette technique a bien permis de créer une version de CAN [RFH$^+$01, Rat02] sensible à la topologie [RHK$^+$02]. Elle ne peut toutefois pas être appliquée à des espaces d'identification à une seule dimension, comme est le cas de la majorité des DHT (e.g. Chord [SML$^+$01], Pastry [RD01], etc.)

### *3.1.3 - Choix du voisin à proximité*

Comme la technique d'agencement géographique, la technique du choix du voisin à proximité (*proximity neighbor selection* ou *PNS*) [RSS02] intervient dans la construction d'un *overlay* conscient

de la topologie sous-jacente. Mais au lieu de biaiser l'attribution des *nodeId*s, cette technique choisit pour les entrées de sa table de routage des références vers les nœuds de son voisinage qui satisfont les contraintes algorithmiques du protocole mis en œuvre. La technique *PNS* n'est alors susceptible de satisfaire ses ambitions que si elle est appliquée à un algorithme qui permet une certaine liberté dans la construction des tables de routage sans affecter le diamètre du réseau. Elle peut donc être appliquée à la construction du *neighborhood set* de Pastry [RD01], par exemple, mais pas à la construction de DHTs CAN [RFH+01, Rat02] ou Chord [SML+01], où chaque entrée de la table de routage fait référence à un point bien précis de l'espace d'identification *overlay*. Mais quand elle est mise en œuvre, la technique *PNS* introduit un surcoût assez faible et facilite la gestion de l'antémémoire (puisque les nœuds voisins dans le réseau *overlay* sont alors susceptibles d'être voisins au niveau sous-jacent). Cette technique de choix du voisin à proximité s'adapte à des caractéristiques de réseau variables. Elle peut être considérée aussi comme un bon compromis qui diminue le diamètre du réseau et préserve l'équilibre de charge. Cependant, la découverte des voisins est étroitement liée au protocole.

## 3.2 - Solutions de prise en compte de la topologie du réseau sous-jacent

Les trois techniques d'exploitation de l'information de proximité présentées au paragraphe précédent ont des limites importantes, et parfois s'impliquent dans le protocole de routage *overlay*. Voilà pourquoi de nouvelles solutions structurelles ont émergé. Elles sont nombreuses et la liste continue à s'allonger. Dans ce paragraphe, nous présentons brièvement les principaux systèmes P2P utilisant des DHTs et connus pour prendre en compte la topologie du réseau sous-jacent, à savoir : Brocade [JZD+02], l'*expressway* (la voie expresse) [XMK03], Hieras [XMH03], Toplus [GER+03] et Plethora [FGJ04].

### *3.2.1 - Brocade*

Brocade [JZD+02] construit un réseau *overlay* secondaire au-dessus d'un réseau P2P utilisant une DHT. Ce réseau secondaire exploite les caractéristiques du réseau sous-jacent au réseau P2P initial. Il est constitué en fait des pairs ayant de grandes capacités réseau (en termes de bande passante, puissance du processeur, etc.) et situés près des points d'accès au réseau IP (e.g. passerelles, routeurs). Ces pairs particuliers sont dits des *super-nœuds* et agissent comme des points de repère (*landmarks*) pour chaque domaine du réseau IP. Chaque *super-nœud* gère un groupe de pairs locaux afin de réduire le trafic dans le réseau. En même temps, chaque *super-nœud* garde des informations de tous les pairs de son domaine ; ce qui peut les engorger.

Pour router un message dans Brocade, un pair se connecte au *super-nœud* le plus proche et utilise le réseau *overlay* secondaire comme raccourci vers la destination. L'usage de la bande passante est ainsi réduit, de même que le nombre total de sauts IP. Sauf que le réseau secondaire de Brocade utilise toujours un routage logique qui ne prend pas en compte la topologie sous-jacente. Du coup, Brocade ne résout le problème qu'à un certain degré et le ramène à un réseau secondaire plus petit (en diamètre et degré).

### *3.2.2 - Expressway*

L'*expressway* (la voie expresse) [XMK03] est un réseau auxiliaire à un réseau P2P existant utilisant les DHTs. Il est construit au-dessus de ce réseau dans le but d'accélérer le routage *overlay* en tirant profit de l'hétérogénéité inhérente au réseau IP sous-jacent. Cette hétérogénéité est assez diversifiée : en termes de connectivité des nœuds, leur proximité physique relative, leur disponibilité, leur capacité de transmission, etc.

L'*expressway* est formée par des *expressway nodes*. Ce sont des pairs de grandes capacités et situés près des passerelles ou près

des routeurs. Pour la construction de cette voie expresse, il existe deux approches génériques.

- La première approche utilise la topologie des niveaux des ASs, dérivée des tables BGP [RFC1771]. Pour router un message, un pair commence par contacter l'*expressway node* qui se trouve dans son AS. Cette approche nécessite de chaque pair de l'*expressway* de connaître tous les nœuds de son AS. De plus, les revers de cette approche sont similaires à ceux de Brocade [JZD+02].
- La deuxième approche utilise une technique de numérotage de points de repère par groupement de nœuds et exploite la proximité sous-jacente par la technique *PNS* [RSS02], discutée ci-dessus (paragraphe 3.1.3). Pour router un message, un pair commence par contacter l'*expressway node* qui se trouve dans son groupement. Le routage se poursuit par une technique de distance vectorielle.

### *3.2.3 - Hieras*

Hieras [XMH03] est un système hiérarchique multi-niveau destiné à remédier à un routage *overlay* indifférent aux temps de latence dans le réseau sous-jacent. Chaque pair appartient simultanément à tous les niveaux et gère autant de listes ou tables nécessaires à son appartenance à chaque niveau.

Au niveau supérieur de l'architecture, tous les pairs actifs sont regroupés dans un même ensemble, appelé *ring*. C'est le niveau du réseau P2P existant ; les autres niveaux s'intercalent donc au-dessus du réseau *underlay* initial. À chacun de ces niveaux *overlay*s intermédiaires, les pairs adjacents (au sens topologique) sont groupés en plusieurs *ring*s disjoints. Chacun de ces *ring*s constitue un sous-groupe du réseau P2P global. L'algorithme de routage dans chaque *ring* est le même que celui mis en œuvre au niveau global.

Les différents *ring*s des niveaux intermédiaires sont construits de sorte que le temps de latence moyen entre deux quelconques de leurs pairs décroît en passant d'un niveau intermédiaire à un autre qui lui est inférieur. Ce temps de latence sera donc minimal au niveau *overlay* le plus bas, et les *ring*s y sont le plus petit.

Afin de pouvoir estimer une telle proximité pour chaque *ring*, Hieras emploie une technique de bacs distribués (ou *distributed binning*) [RHK+02] qui nécessite l'existence de nœuds de marquage bien définis.

Le mécanisme de routage s'exécute en premier au niveau le plus bas, dans le *ring* de l'initiateur de la requête. En cas d'échec, le routage reprend dans le *ring* du niveau supérieur, et ainsi de suite jusqu'à satisfaction de la requête. Généralement, la majorité des requêtes sont satisfaites à un niveau intermédiaire, réduisant ainsi les temps de latence puisque les requêtes n'atteignent donc plus le réseau global au niveau le plus haut.

### *3.2.4 - Toplus*

Toplus [GER+03] est un service hiérarchique de recherche de données dans les réseaux P2P utilisant des DHTs. Il regroupe les pairs selon le préfixe IP de leur réseau et organise les groupes hiérarchiquement en de nouveaux groupes suivant la topologie hiérarchique de l'Internet récupérée des tables BGP [RFC1771]. Ainsi un AS est-il subdivisé en une hiérarchie IP.

Le mécanisme de routage de Toplus est basé sur une généralisation de la règle du plus long préfixe correspondant. Comme Kademlia [MM02] (cf. p. 54), il utilise la métrique du choix exclusif, XOR.

La structure et les performances de Toplus suivent bien celles du réseau Internet. Cependant, ce système présente un certain nombre d'inconvénients. Évidemment, il est susceptible de subir des pannes de nœuds corrélées qui peuvent faire tomber tout un groupe

de nœuds dont les adresses IP sont reliées. Par ailleurs, le nombre de pairs actifs dans un (sous-)groupe n'est pas nécessairement proportionnel au nombre d'adresses IP qu'il couvre. La population de l'espace d'identification n'est donc pas uniforme, ce qui conduit à un déséquilibre de charge entre les pairs.

### *3.2.5 - Plethora*

Plethora [FGJ04] est un référentiel de données à large échelle, structuré en deux niveaux. Au niveau inférieur, le réseau *overlay* global contient tous les pairs. Au-dessus, le cœur de routage de Plethora organise les pairs en plusieurs *overlay*s locaux selon leur proximité relative et en conformité avec les AS. Les *overlay*s locaux sont tous à un même niveau et servent d'antémémoires de la localité pour l'*overlay* global afin d'améliorer les temps d'accès aux éléments de données. Les requêtes sont alors routées en premier dans l'*overlay* local. Cependant, la taille des *overlay*s locaux impacte les performances du système. Pour cela, elle est définie par des paramètres de système, constamment contrôlée et régulée au besoin par deux algorithmes distribués :

- les paramètres systèmes fixent un minimum et un maximum de nombre de pairs par *overlay* local ;
- dans chaque *overlay* local, le pair à l'identifiant le plus petit se charge de contrôler sa taille ;
- au besoin, un algorithme se charge d'unifier deux *overlay*s locaux devenus petits ;
- et si besoin y est, un autre algorithme se charge de partager un *overlay* local en garantissant avec une grande probabilité que les pairs issus d'un même AS restent dans un même *overlay* local.

Plethora suppose des pairs ayant une bonne connectivité Internet en termes de bande passante, et étant partiellement statiques.

## 3.3 - Solutions de coopération entre flux pair-à-pair et opérateur de réseaux

De nombreux systèmes ont émergé visant à améliorer les performances d'un système P2P utilisant une DHT, en exploitant l'information de proximité sous-jacente. Deux architectures proposent (indépendamment des DHTs) une coopération inter-couches entre le trafic P2P et les politiques d'un opérateur de réseaux. L'opérateur de réseau considéré dans ces deux propositions est un ISP.

### *3.3.1 - Solution de service intermédiaire*

Une interaction active entre le réseau P2P et l'infrastructure sous-jacente d'un ISP est possible en passant par le service d'un serveur intermédiaire [AFS07]. Ce serveur, appelé *oracle*, est hébergé par l'ISP et aident les usagers P2P à choisir des pairs de façon optimale, notamment dans le cas d'une application P2P de téléchargement de fichiers. Étant géré par l'ISP, l'*oracle* a un accès direct à toutes les informations nécessaires (e.g. celles relative à la topologie physique réelle).

Avant d'entrer en lien avec un autre pair, chaque pair envoie à l'*oracle* une liste des pairs potentiellement voisins. L'*oracle* ordonne alors la liste selon un certain nombre de critères que chaque ISP est libre de spécifier. Parmi ces critères, l'on peut citer : la proximité ou le taux de bande passante disponible par lien (entre un pair de la liste et le pair qui a construit la liste), les politiques de routage, les termes des accords signés avec d'autres ISPs, etc.

L'*oracle* agit donc à la fois comme une couche de routage abstraite sous-jacente au réseau P2P et comme un service offert par l'ISP. Il peut être implémenté de façon distribuée afin de pallier les risques d'engorgement.

### *3.3.2 - Provider Portal for Peer-to-Peer*

P4P (*Provider Portal for P2P*) [XKS+07] est une architecture légère permettant une communication explicite entre le trafic P2P et

les ISPs dans le cadre des applications à partage de fichiers. Elle vise à réduire le trafic dans la dorsale (*backbone*) et diminuer les coûts d'opérations. Elle s'appuie sur le fait qu'un ISP est le mieux placé pour déterminer la localité et router les messages non seulement vers des pairs voisins mais aussi vers des pairs accessibles par des liens à meilleur débit et peu chargés.

La structure de P4P consiste en un plan de données et un plan de contrôle avec une sorte d'entité centrale appelée *iTracker*. Cet *iTracker* fournit trois types d'informations concernant l'ISP :

– la topologie et le statut du réseau ;
– les politiques et directives de l'ISP ;
– et les capacités du réseau.

Avec son *iTracker*, P4P semble être une sorte d'architecture centralisée appliquant une architecture CDN (cf. paragraphe 1.2.4.2 p. 29) à un réseau P2P de partage de fichiers.

CONCLUSION DU CHAPITRE –

*Ce chapitre a analysé les techniques existantes d'exploitation de l'information de proximité sous-jacente, ainsi que les principales solutions de prise en compte de la topologie sous-jacente dans les systèmes P2P à base de DHT. Ces techniques et solutions ne permettent cependant pas une coopération entre le flux P2P et les opérateurs de réseaux. Le chapitre se poursuit donc par l'analyse de deux solutions émergentes expressément dédiées à une telle coopération. Or il s'avère que l'une nécessite une étude systématique de chaque requête, et l'autre nécessite une infrastructure particulière et ne respecte pas la parité pure recherchée entre les nœuds du système pour un routage P2P efficace.*

*Les chapitres suivants sont dédiés à nos contributions en vue d'une sensibilisation efficace d'un réseau P2P utilisant une DHT à l'infrastructure du réseau sous-jacent, avec une coopération possible entre les flux P2P et les opérateurs de réseaux.*

# CHAPITRE 4

## *DÉFINITION, CONCEPTION, SPÉCIFICATION ET ANALYSE D'UNE SOLUTION PAIR-À-PAIR SENSIBLE AU CONTEXTE*

*CAP : A CONTEXT-AWARE PEER-TO-PEER SYSTEM*

*Ce chapitre définit, spécifie et analyse un système P2P sensible au contexte du réseau. Cette solution, baptisée CAP (Context-Aware P2P system), entend introduire une sémantique dans les identifiants des pairs et des objets. Pour cela, elle construit la DHT avec une fonction de hachage particulière, à savoir HMAC (keyed-Hash Message Authentication Code).*

*Le chapitre commence par un rappel sur la fonction HMAC.*

### 4.1 - Keyed-Hash Message Authentication Code

HMAC (*keyed-Hash Message Authentication Code*) [RFC2104] est une fonction à trois paramètres : un message, une fonction de hachage et une clé. Elle est définie comme suit :

$$\text{HMAC}(h, k, m) = h(k \oplus opad \,||\, h(k \oplus ipad \,||\, m))$$

- *h* une fonction de hachage cryptographique, comme MD5 ou SHA-1 ;
- *m* est le message ;
- *k* est la clé de sécurité partagée uniquement par les entités impliquées dans l'échange du message. Si *k* est plus longue qu'*opad* et *ipad*, elle est remplacée par *h*(*k*). Si *k* est plus courte que ces constantes, elle est complétée par des zéros ;
- *opad* et *ipad* sont respectivement les octets 0x36 et 0x5C répétés chacun B fois (avec souvent B=64).
- || symbolise la concaténation ;

– $\oplus$ est l'opérateur de choix exclusif bit à bit (XOR). Il est prioritaire par rapport à la concaténation ;

HMAC est donc une fonction de hachage dont le paramètre, calculé de façon spécifique sur base d'opérations XOR et de concaténation, est en partie déjà haché. La fonction HMAC appliquée à *m* donne alors un résultat complètement différent de *h*(*m*).

HMAC a été initialement définie dans le domaine de la sécurité, et la clé sert la sécurité. En effet, le résultat de la fonction HMAC est un condensat, appelé MAC (*Message Authentication Code*), qui permet de vérifier l'intégrité du message et l'authenticité de sa source. L'émetteur et le récepteur partageant une même clé *k*, l'émetteur envoie au destinataire le message *m* et son condensat HMAC. Le récepteur applique alors la fonction HMAC au message reçu et compare le résultat ainsi obtenu avec la valeur du condensat HMAC reçu.

Afin de construire un système P2P sensible au contexte, nous proposons d'utiliser un routage contextuel. Pour cela, nous proposons de remplacer la fonction de hachage de la DHT par une fonction HMAC. Dans ce cas, la signification des paramètres en entrée de la fonction est adaptée :

– *h* devient la fonction de hachage initialement utilisée par la DHT ;
– *m* représentera selon le cas, soit l'adresse IP du pair, soit la référence de l'objet, ceux-là même qui permettent de construire les *nodeId* et *objectId* respectivement ;
– *k* sera une clé particulière, que nous appellerons *Hkey* (comme étant la clé de la fonction HMAC, différente de la clé de recherche qui est un *objectId*). Cette clé va représenter le contexte ; elle va prendre un sens et nous allons lui définir une sémantique (au paragraphe suivant). Elle sera nécessairement connue de tous les pairs d'une même DHT ; autrement aucune recherche ne pourra être effectuée dans l'espace d'identification de la DHT.

## 4.2 - Définition sémantique d'une Hkey

Une *Hkey* peut être simple ou composée. Elle peut être aussi dérivée et dérivable.

### *4.2.1 - Hkey simple*

Nous disons qu'une *Hkey* est simple, si elle représente un seul critère ou un seul paramètre caractéristique.

Voici quelques exemples de ce que peut être une *Hkey* simple : une clé secrète, le numéro de l'AS ou le nom du domaine administratif d'appartenance, le nom ou l'identifiant d'un groupe de communication (sécurisé ou pas), un paramètre de qualité de service (e.g. la bande passante minimale disponible, la vitesse minimale de fonctionnement du processeur, la puissance minimale s'il s'agit d'un système mobile, la capacité de stockage minimale, etc.), un paramètre de localisation (e.g. proximité réseau ou distance métrique, localisation GPS (*Global Positioning System*), pays d'appartenance, etc.), la langue ou le type des fichiers partagés (qui peut être basé sur des métadonnées), leur taille, un mot clé, etc.

### *4.2.2 - Hkey composée*

Nous disons qu'une *Hkey* est composée, si elle représente plusieurs critères ou paramètres. Elle est donc composée de plusieurs *Hkey*s simples. Elle est le résultat du hachage de la concaténation de ses composantes. Par exemple, si nous nous intéressons à trois métriques $m_1$, $m_2$, et $m_3$, nous définissons une *Hkey* composée, ayant pour valeur $h(m_1||m_2||m_3)$.

### *4.2.3 - Hkey dérivée*

Une *Hkey* composée peut être décomposée en des *Hkey*s dérivées, tel que chacune soit formée d'un ou plusieurs des éléments simples de la forme composée. Ceci revient à former une *Hkey* composée en concaténant graduellement et partiellement plusieurs

*Hkey*s simples en *Hkey*s intermédiaires, qu'on dira dérivées de la composée.

Des *Hkey*s dérivées peuvent être aussi définies à partir d'une *Hkey* simple. Par exemple, si une certaine *Hkey* simple peut prendre quatre valeurs différentes pour un même pair (e.g. si le pair réside dans quatre zones), des *Hkey*s dérivées peuvent être définies par la combinaison de deux ou plusieurs de ces valeurs.

## 4.3 - Architecture de CAP

HMAC étant une fonction de hachage, le condensat est de même nature et de même longueur que le résultat de cette même fonction de hachage. L'utilisation de HMAC pour la construction d'une DHT n'affecte donc ni l'espace d'identification, ni le principe des DHTs.

Avec HMAC, chaque *nodeId* ou *objectId* n'est plus le résultat d'un $h(m)$, mais d'un $HMAC(h, k, m)$.

Si $k$ (c.-à-d. *Hkey*) était une constante, il est évident que, les mécanismes de routage (pour la localisation d'une clé) et de maintenance (pour la (dé)connexion d'un pair) resteraient inchangés.

Maintenant que *Hkey* représente le contexte du réseau, il s'agit d'un paramètre variable. Le réseau P2P devient alors un réseau logique hétérogène. En d'autres termes, il s'agit d'une agglomération *overlay* de réseaux homogènes distincts, pour chacun desquels *Hkey* a une valeur constante. Chaque réseau homogène est appelé *zone*. Chaque zone est caractérisée par une *Hkey* et est identifiée par $h(Hkey)$ afin de respecter l'espace d'identification. Une *Hkey* sert alors d'étiquette pour l'espace d'identification de la zone concernée. Elle est donc utilisée pour le calcul (avec HMAC) de tous les *nodeId*s et *objectId*s de la zone. En d'autres termes, tous les *nodeId*s et *objectId*s d'une même zone connaissent une seule et même *Hkey*. Or ces identifiants se trouvent dans une même DHT. La DHT d'une zone est donc aussi étiquetée par cette même *Hkey*. Cette

nouvelle DHT est virtuelle par rapport à la DHT de base utilisant la fonction *h*. Elle sera appelée VDHT (pour *Virtual DHT*).

Le type des *Hkey*s sera défini dans un profil global du système. Ce profil pourrait être disponible, par exemple, en distribué ou sur un nœud d'amorce géré par le fournisseur de service ou de réseau. La manière dont il est chargé dans le système n'affecte cependant ni le routage P2P ni les opérations des différents pairs. Il ne sera donc pas détaillé dans la suite.

Dans le cas de l'existence de *Hkey*s dérivées, ce profil fournira aussi leurs structures (paramètres ou valeurs), ainsi qu'un ordre de priorité pour leurs usages dans les procédures de routage et de recherche de ressources.

Le mécanisme de détermination des *Hkey*s est toutefois extérieur à CAP. Il ferait partie d'un mécanisme de gestion de politiques du système.

Le mécanisme de l'évaluation individuelle des critères et paramètres par chaque pair est aussi extérieur à CAP. Il faudra pour cela sans doute se servir des bases d'information du système (e.g. des tables BGP [RFC1771] pour avoir les numéros des AS d'appartenance).

Ces deux types de mécanismes ne sont pas discutés dans la suite. CAP s'intéresse principalement aux mécanismes de routage et aux mouvements des pairs.

Le réseau P2P est donc une agglomération hétérogène de zones virtuelles homogènes chacune. Il peut alors se décomposer en plusieurs couches *overlay* secondaires disposées au-dessus du réseau global et caractérisée chacune par une *Hkey* précise. Ces *overlay*s correspondent aux différentes zones et sont dits locaux. C'est l'algorithme de routage (e.g. Chord [SML+01], Pastry [RD01], etc.) mis en œuvre au niveau global qui sera repris dans chacun de ces *overlay*s locaux.

Chaque pair reste actif dans le plan P2P global. Il peut en plus participer à un, plusieurs ou aucun de ces *overlay*s secondaires. Ceci dépend de la correspondance entre ses propriétés et la sémantique des différentes *Hkey*s du système, ainsi que du nombre des différentes *Hkeys* et de celui des éventuelles *Hkey*s dérivées.

Pour des raisons de simplicité, dans la suite, et sauf mention contraire explicite, nous n'aborderons pas les *Hkey*s dérivées et nous ne distinguerons pas entre une *Hkey* simple et une *Hkey* composée.

Dans le cas où le profil du système définit une seule *Hkey* (qu'elle soit donc simple ou composée, mais pas de *Hkey* dérivée), tous les *overlay*s locaux seront à un même niveau au-dessus du plan global. Et chaque pair participera à au plus deux *overlay*s : le global et un local.

La figure 4.1 illustre le cas d'un tel système. On y voit, au niveau des *overlay*s locaux, trois zones distinctes P1, P2 et P3, chacune caractérisée par une *Hkey* différente. Les pairs de chacune de ces zones se trouvent au niveau P2P global, où d'autres pairs, n'appartenant à aucune de ces zones, sont aussi actifs. Ces pairs peuvent soit faire partie d'autres zones que celles représentées, soit ne faire partie d'aucune zone s'ils ne satisfont pas les caractéristiques représentées par les *Hkey*s définies dans le profil du système. Au niveau IP, sont représentés aussi des nœuds n'appartenant pas à un réseau P2P, ou du moins pas à celui considéré dans l'exemple.

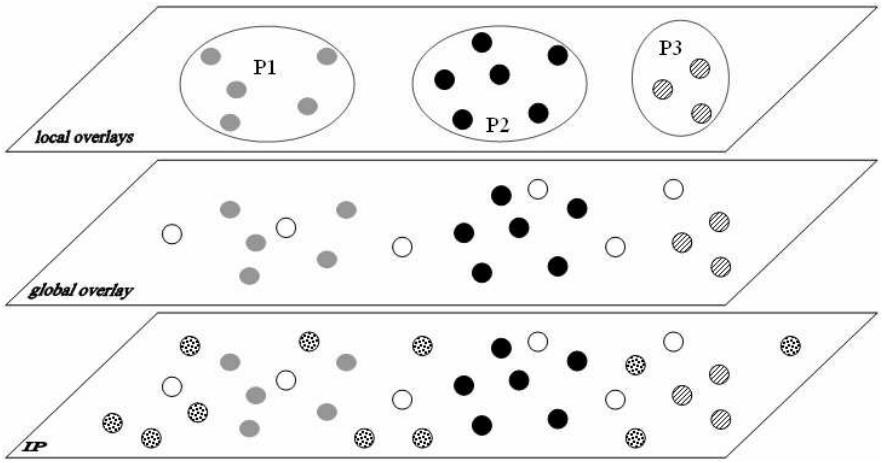


***Figure 4.1 :*** *Vue d'ensemble du système sans Hkeys dérivées*

Chaque pair a un *nodeId* dans chaque *overlay* où il existe, et donc autant de *nodeId*s que le nombre total d'*overlay*s auxquels il participe. Ceci est analogue à un terminal ayant plusieurs interfaces réseaux et donc autant d'adresses IP : une par réseau. Chaque *nodeId* est le condensat d'une fonction HMAC utilisant la *Hkey* de l'*overlay* concerné. Dans un souci de

simplicité, nous n'exigeons pas d'unicité entre les différents espaces d'identification.

Quant au niveau global, où les pairs sont hétérogènes, le calcul des identifiants avec HMAC est générateur de conflits entre les *nodeId*s. Pour cela, deux cas sont possibles :

– soit cette nouvelle méthode est appliquée uniquement au niveau des *overlay*s locaux ;
– soit une valeur est fixée pour la *Hkey* au niveau global (e.g. 0x00 répété 64 fois) et définie dans le profil du système. Ce cas s'appliquera aux nouveaux pairs se connectant au système, et se généralisera au rythme de la dynamique du réseau. Cependant, un risque de collision dans la phase intermédiaire n'étant pas à exclure, la préférence va au premier choix qui ne modifie pas la génération des *nodeId*s au niveau global.

Comme un pair participe à plusieurs *overlay*s, il partage alors ses ressources dans chacun de ces *overlays*. Une même ressource aura donc autant d'identifiants (*objectId*s) que le pair qui la partage en a. Ceci revient à donner à un objet plusieurs pseudonymes. De même que pour les *nodeId*s nous n'exigeons pas d'unicité entre les différents espaces d'identification.

## 4.4 - Mécanisme de routage dans CAP

Le mécanisme de routage est le même dans chaque *overlay* local. Il s'agit de celui mis en œuvre au niveau global (e.g. Chord [SML+01], cf. paragraphe 1.4.2 p. 44). Une différence s'installe toutefois d'un *overlay* à l'autre : la *Hkey* caractéristique et du coup, les *nodeId*s et *objectId*s utilisés, ainsi que la DHT (ou VDHT) consultée.

Quand un pair veut rechercher un objet, il commence par calculer sa clé en utilisant la *Hkey* de l'*overlay* local où il réside. Puis il lance sa requête dans cet *overlay* en s'identifiant par son *nodeId* correspondant et en utilisant la VDHT correspondante. Ainsi la requête n'atteint que des pairs résidant dans le même *overlay* local.

Si la requête n'aboutit pas, elle est alors relancée dans l'*overlay* global avec les *objectId*, *nodeId* et DHT correspondants.

Quand des *Hkey*s dérivées sont définies dans le système, et que le pair réside dans plusieurs *overlay*s locaux, la requête est lancée dans chacun de ces *overlay*s. Ceci se fait à tour de rôle conformément à l'ordre défini dans le profil du système, et jusqu'à satisfaction de la requête. La requête peut aussi se lancer simultanément dans les *overlay*s locaux concernés, mais ceci génèrerait un trafic supplémentaire pas nécessairement utile. En tout cas, ce n'est qu'en dernier lieu, qu'une requête est lancée dans l'*overlay* global, si elle n'a pas encore abouti.

Il est à noter que seul le pair initiateur d'une requête peut relancer (à un niveau inférieur) une requête non aboutie. Aussi, un pair reconnaît la VDHT qu'il doit utiliser grâce au *nodeId* par lequel il a été sollicité.

Quand une ressource n'est trouvée que dans l'*overlay* global, le pair qui la récupère la remet en partage dans son *overlay* local, après lui avoir calculé son nouvel identifiant basé sur la *Hkey* correspondante, tout comme il y partagerait une nouvelle ressource.

## 4.5 - Mouvement des pairs dans CAP

Les mouvements d'arrivée et de départ d'un pair sont gérés de façon identique dans tous les *overlay*s du système et conformément au protocole de routage mis en œuvre dans l'*overlay* global. Cependant après s'être joint à l'*overlay* global et avant de pouvoir se joindre à un *overlay* local, un pair *n* doit vérifier l'existence d'au moins un autre pair $n_1$ actif dans cet *overlay* local. Pour cela, il doit effectuer certaines opérations, qui vont lui permettre de reconnaître un tel pair $n_1$, et à défaut, d'initier l'*overlay* en question.

### *4.5.1 - La 'Zone Table'*

L'identifiant d'un *overlay* local est sauvegardée dans une '*zone table*' (ou *ZT*). Il s'agit d'une table de données relatives à une même zone. Pour chaque zone, il existe deux ZT : une locale,

stockée dans l'*overlay* local de la zone, et une autre globale, stockée dans l'*overlay* global.

***Tableau 4.1 :*** *Structure de la 'zone table'*

| *zoneId* | *zone label* | *largest nodeId* | *2nd largest nodeId* | *2nd smallest nodeId* | *smallest nodeId* |
|---|---|---|---|---|---|

Les deux ZTs ont une structure identique en six champs (cf. tableau 4.1) :

- le *zoneId* vaut *h*(*Hkey*) et identifie la table ;
- le '*zone label*' est la valeur de la *Hkey* en chaîne de caractères. Il est particulièrement utile pour éviter d'éventuels conflits dans le cas des *Hkey*s dérivées ;
- les *nodeId*s des deux pairs aux plus grands *nodeId*s dans l'*overlay* local, et des deux autres aux plus petits *nodeId*s dans l'*overlay* local. Il s'agit des *nodeId*s dans l'espace d'identification local pour la ZT locale, et des *nodeId*s dans l'espace d'identification global pour la ZT sauvegardée au niveau global. Ce sont les pairs ayant ces *nodeId*s qui assurent la cohérence entre les deux ZTs. S'il y a moins de quatre pairs dans l'*overlay* de la zone, les champs restants sont mis à NULL.

Chaque ZT est donc gérée comme un objet identifié par le *zoneId*. Dans chaque *overlay*, la ZT est stockée sur un pair dit « nœud de rendez-vous » (ou RP – *Rendezvous Point*). Il s'agit du pair dont le *nodeId* est le plus proche numériquement du *zoneId*, conformément à la DHT mise en œuvre dans le réseau. Au niveau global, un même RP peut donc stocker plusieurs ZTs, relatives chacune à une zone différente.

La ZT est mise à jour quand un des pairs qui y sont référencés disparaît de l'*overlay* local (départ volontaire ou panne), ou bien quand un nouveau pair arrive dans cet *overlay* avec un *nodeId* tel qu'il doit être référencé dans la ZT. La mise à jour de la table se faisant donc suite à une arrivée ou une disparition de pair, nous la

présentons convenablement dans chacun des paragraphes suivants (4.5.2 et 4.5.3).

Si un RP vient à quitter le réseau ou tomber en panne, la prise en charge de la ZT est gérée comme celle des autres objets que le RP stockait. Par souci de robustesse, la ZT peut être aussi dupliquée dans l'overlay où elle se trouve. En particulier, elle pourrait l'être sur le pair ayant le plus petit identifiant dans l'*overlay*, soit le pair référencé dans le dernier champ de la ZT.

### *4.5.2 - Arrivée d'un pair*

(Il est à noter que si des *Hkey*s dérivées sont définies dans le système et concernent le nouveau pair, celui-ci réitère le processus d'adhésion décrit dans ce paragraphe pour chaque *zone* à laquelle il veut appartenir, et ce avec la *Hkey* correspondante.)

Une fois inséré dans le réseau global, le nouveau pair prend connaissance du profil du système (qui peut lui être aussi communiqué). Il connaît alors les différentes *Hkey*s. Par un mécanisme externe au réseau P2P, le nouveau pair évalue et détermine les *Hkey*s qui lui correspondent. Il calcule alors l'identifiant *h*(*Hkey*) de chaque réseau local auquel il doit appartenir. Cet identifiant étant aussi celui de la ZT de l'*overlay* local, le nouveau pair initie une requête *zone_table_request( )* au niveau global pour récupérer la ZT. Deux cas sont alors possibles suivant que l'*overlay* local en question est déjà initialisé ou pas.

#### *4.5.2.1 - Overlay local initialisé*

Dans le premier cas, l'*overlay* local est déjà initialisé. Il a une ZT stockée sur un RP au niveau global. Le nouveau pair peut donc la récupérer. Il va ainsi connaître jusqu'à quatre pairs résidants dans la zone qu'il doit rejoindre. Il en choisit un et lui envoie dans l'*overlay* global un message de demande d'adhésion. Dans ce message, le nouveau pair

envoie la *Hkey* de la zone et *n*, son propre *nodeId* calculé avec cette *Hkey*. Le pair contacté envoie alors au nouveau pair, son *nodeId* local $n_1$. L'insertion du nouveau pair dans l'*overlay* local se poursuit alors conformément au protocole de routage mis en œuvre (e.g. à travers la fonction $n.join(n_1)$, si le protocole est Chord [SML+01] ; cf. paragraphe 1.4.2 p. 46).

Une fois le nouveau pair correctement inséré dans l'*overlay* local, il cherche dans celui-ci la ZT locale. Quand il la récupère, il compare son *nodeId* local, *n*, aux *nodeId*s qui s'y trouvent. Si *n* est supérieur au deuxième *nodeId* le plus grand ou inférieur au deuxième *nodeId* le plus petit, le nouveau pair met à jour la ZT locale avec *n*, et se charge de mettre à jour la ZT globale de la zone. Les champs des deux ZTs se succédant dans le même ordre, le nouveau pair remplace par son *nodeId* global le champ de la ZT globale ayant le même numéro d'ordre que le champ qu'il a modifié dans la ZT locale. Par la suite, il en informe le RP qui lui avait fournit la ZT au niveau global, afin que la mise à jour soit cohérente. Le nouveau pair peut aussi directement charger le RP global de la mise à jour de la ZT globale.

#### *4.5.2.2 - Overlay local inexistant*

Dans le second cas, l'*overlay* local n'est pas encore initié. Le nouveau pair est le premier à vouloir adhérer à cet *overlay*. Il n'y a pas de ZTs correspondantes. La requête lancée dans l'*overlay* global pour récupérer la table n'aboutit pas : soit il y a un retour de message (de la part du supposé RP) informant de l'inexistence de l'objet recherché, soit l'initiateur de la requête déduit ceci suite à l'expiration d'un jeu de temporisateurs. C'est alors au nouveau pair de créer la ZT globale de la zone qu'il voulait

rejoindre. Il le fait avec les *zoneId* et *zone label* correspondants et son propre *nodeId* global, et met les trois autres champs à NULL. Une fois la ZT globale créée, le nouveau pair la partage (elle sera stockée sur le RP) et initie l'*overlay* de sa zone. Ensuite, il crée la ZT locale, copie de la ZT globale où il remplace son *nodeId* global par son *nodeId* local calculé avec la *Hkey*. Les trois nouveaux pairs suivants dans la zone modifieront les champs initiés à NULL dans les ZTs concernées.

Dans un réseau P2P bien peuplé, ce dernier cas est peu probable, notamment avec des *Hkey*s communes (e.g. si la zone correspond à un AS, si la *Hkey* désigne un type populaire de fichiers, si la *Hkey* est la bande passante typique d'une majorité de pairs, etc.)

### *4.5.3 - Disparition d'un pair*

La disparition d'un pair du système (départ explicite, départ silencieux, ou panne) est généralement remarquée quasi-simultanément dans tous les *overlay*s auxquels il appartenait. Elle est traitée de façon identique mais indépendante dans chaque *overlay*, et conformément au protocole mis en œuvre au niveau global.

S'il s'agit d'un des pairs référencés dans les ZTs relatives à la zone affectée par la disparition, celles-ci seront mises à jour. En effet, le RP local vérifie périodiquement l'état des pairs référencés dans la ZT (locale) dont il a la charge. Lorsqu'un d'eux ne répond plus, le RP le remplace et réordonne les *nodeId*s de la ZT. Il le remplace par le pair suivant au *nodeId* strictement plus grand ou strictement plus petit. En fait, le RP local garde en antémémoire une liste courte des pairs de la zone aux plus petits ou aux plus grands *nodeId*s. Il est peu probable que tous tombent en même temps, mais si le cas se présente, le RT met le champ restant à NULL et attend une nouvelle adhésion.

Quand le RP local a mis à jour la ZT locale, il envoie localement un message au pair qu'il vient d'ajouter pour l'informer de la table où il est référencé. Ce pair se charge alors de la mise à jour de la ZT globale, exactement comme ceci se fait dans le cas de l'adhésion d'un nouveau pair à un *overlay* local déjà initié (cf. paragraphe 4.5.2.1, p. 92).

## 4.6 - Analyse de CAP

L'utilisation de HMAC pour la génération des identifiants (des pairs et des objets) permet à un système P2P utilisant les DHTs d'être sensible au contexte du réseau. Ceci est réalisable à travers la clé de HMAC, baptisée *Hkey* pour ce système. L'algorithme de la DHT en soi n'est pas affecté, même si des opérations complémentaires viennent s'y greffer pour la cohésion du système.

Ce changement de fonction de hachage donne une sémantique aux identifiants. Il permet aussi de structurer un réseau P2P initialement hétérogène en des couches secondaires virtuelles homogènes chacune, avec une granularité définie par la *Hkey*. A chacune de ces couches secondaires, il y a donc une DHT virtuelle, dite VDHT, qui permet une gestion et une recherche contextuelles des données.

En effet, la recherche d'une ressource ou d'un élément de données dans CAP commence toujours dans un *overlay* local en utilisant les services de la VDHT correspondante. Or, tous les identifiants référencés dans une VDHT sont calculés avec une fonction HMAC utilisant la même *Hkey* qui représente le contexte de l'*overlay* et caractérise donc la VDHT. Ainsi un *objectId* ne peut être référencé dans une VDHT que si l'objet est bien dans l'*overlay* (local) correspondant.

Par ailleurs, toute requête qui aboutit dans l'*overlay* où elle a été initiée est garante de l'existence de l'objet dans cet *overlay*. Il en découle donc que le pair initiateur de la requête peut récupérer l'objet d'un pair partageant le même contexte et à travers un chemin conforme à ce contexte.

En garantissant la récupération d'un objet de la zone où il a été trouvé, CAP offre un cloisonnement des ressources, contextuel et efficace, qui réduit le temps de latence des messages et améliore le temps de récupération des données. De plus, la mise en partage dans un *overlay* local d'une ressource récupérée au niveau global favorise la popularisation des objets dans le système.

## CONCLUSION DU CHAPITRE –

*Ce chapitre a défini, spécifié et analysé CAP (Context-Aware P2P system), un système P2P utilisant des DHTs et sensible au contexte du réseau. Ce contexte est caractérisé par un ou plusieurs paramètres du système, et ne se restreint pas à la seule topologie sous-jacente. Pour spécifier cette sensibilité, CAP a introduit une sémantique dans les identifiants des pairs et des objets en construisant la DHT avec une fonction de hachage particulière, HMAC, présentée au début du chapitre.*

*CAP est un système générique et configurable, mais où chaque élément (pair ou objet) a plusieurs identifiants indépendants. Le chapitre suivant présentera une nouvelle solution de sensibilité au contexte, où la gestion des identifiants sera simplifiée. Le contexte sera celui d'un opérateur de réseaux.*

# CHAPITRE 5

## *DÉFINITION, CONCEPTION, SPÉCIFICATION ET ANALYSE D'UNE SOLUTION PAIR-À-PAIR ORIENTÉE OPÉRATEUR DE RÉSEAUX NETPOPPS : A NETWORK PROVIDER ORIENTED PEER-TO-PEER SYSTEM*

*Ce chapitre définit, spécifie et analyse un système P2P orienté opérateur de réseau. Cette solution, baptisée NETPOPPS (NETwork Provider Oriented P2P System), adapte les flux P2P à la topologie réseau tout en simplifiant la gestion des identifiants des différents pairs et objets. Elle utilise pour cela la technique de dérivation de clés.*

*Le chapitre commence par la présentation du principe de la dérivation de clés.*

### 5.1 - Principe de la dérivation de clés

La figure 5.1 illustre le principe de la dérivation de clés. La fonction *h* est une fonction injective quelconque. Dans le cas d'un système P2P, *h* sera la fonction de hachage utilisée par la DHT du protocole P2P mis en œuvre au niveau global (non représenté sur la figure).

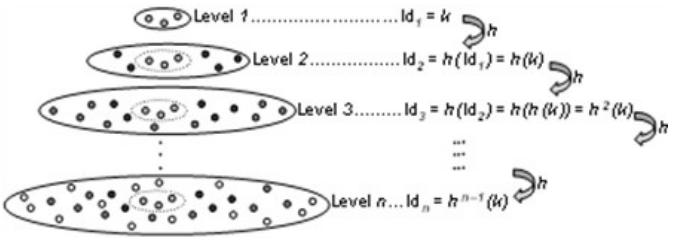


***Figure 5.1 :*** *Principe de la dérivation de clé pour la gestion de clés*

Le principe de la dérivation permet d'obtenir à partir d'une même clé plusieurs clés en relation mathématique simple entre elles et avec la clé d'origine. Dans l'exemple représenté à la figure 5.1, ce mécanisme

s'applique à chaque niveau de la hiérarchie, à partir du niveau supérieur. Au niveau 1 (*Level 1*) chaque élément a un identifiant $Id_1 = k$. C'est la clé. Elle est dérivée (par application de $h$) pour donner l'identifiant du même élément au niveau suivant. Ainsi cet élément est identifié au niveau 2 (*Level 2*) par $Id_2 = h(Id_1)$, au niveau 3 (*Level 3*) par $Id_3 = h(Id_2) = h(h(Id_1)) = h^2(Id_1)$, et ainsi de suite. Cette relation mathématique peut être vérifiée entre n'importe quels identifiants d'un même élément à deux niveaux strictement successifs. En d'autres termes, à tout niveau $i$ (*Level i*), un élément est identifié par $Id_i = h(Id_{i-1}) = h^{i-1}(Id_1)$. Et ainsi, tout élément peut déduire son identifiant $Id_j$ au niveau $j>i$ en dérivant ($j$-$i$) fois son identifiant $Id_i$ du niveau $i$, c.-à-d. $Id_j = h^{j-i}(Id_i)$.

Le principe de la dérivation de clés a déjà été utilisé dans le cadre de la sécurisation des communications privées de groupes hiérarchiques nécessitant une confidentialité particulière dans chacun des sens ascendant et descendant de la hiérarchie [HBB+05]. Nous l'utilisons ici à des fins de routage efficace transparent, indépendamment de toute exigence de sécurité.

## 5.2 - Architecture de NETPOPPS

Le réseau Internet est composé de dizaines de milliers d'ASs reliés entre eux suivant des accords économiques signés entre les opérateurs qui les gèrent [Meu07]. Un réseau P2P interconnecte donc des pairs de différents ASs. Il peut alors se décomposer en plusieurs couches *overlay*s secondaires, disposées au-dessus du réseau global tel que chacune soit conforme à un relevé précis (c.-à-d. une politique, une priorité, une métrique) des accords inter-opérateurs. Pour la présentation de l'architecture de NETPOPPS, nous considérons la métrique distance, exprimée en nombre de sauts d'un AS à l'autre. La priorité donnée à un AS parmi d'autres pour y aller au prochain saut résulte des accords inter-opérateurs négociés par l'opérateur gérant l'AS de départ.

NETPOPPS est cependant une architecture générale non spécifique à un opérateur de réseau donné, tel un FAI. Elle peut être adoptée par exemple

par des universités ou des compagnies multi-sites ayant des politiques dotées de priorités pour la gestion des flux entre leurs différents sites, notamment dans le cadre de certains services P2P.

Pour cela, dans la suite, nous désignons l'opérateur de réseau par NP (pour *Network Provider*) et nous remplaçons le terme AS par « entité de base » ou BE (pour *Basic Entity*). Ainsi un NP peut gérer une ou plusieurs BEs. Dans ce cas, il pourra y avoir priorités et politiques intra-NP à l'égard de ses BEs ; mais dans la suite, nous ne parlerons que d'accords inter-opérateurs.

La fusion de deux BEs est possible virtuellement, dès lors que les accords signés entre leurs NPs autorisent des échanges entre un pair de l'une et un pair de l'autre. La nouvelle entité virtuelle est dite une entité intermédiaire (entre la BE et l'ensemble de tous les pairs actifs) ou IE (pour *Intermediate Entity*). Deux IEs peuvent à nouveau fusionner en une nouvelle IE, et ainsi de suite, tant que la nouvelle entité ne s'est pas confondue avec l'*overlay* P2P global existant. Chaque IE étant la fusion de plusieurs BEs, elle sera gérée indépendamment par autant de NPs. La concordance de ces gestions indépendantes est garantie par la concordance des accords inter-opérateurs.

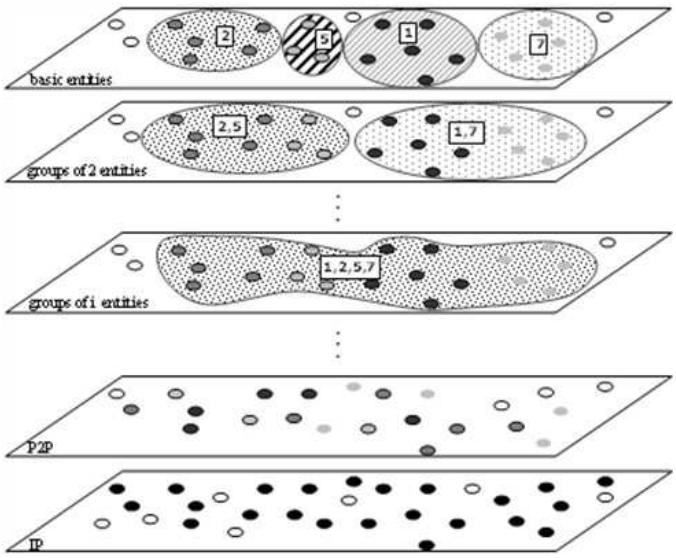


***Figure 5.2 :*** *Vue d'ensemble d'un système NETPOPPS*

La définition d'entités (BE ou IE) garantit que le chemin de connexion P2P, émergeant d'un pair quelconque de l'entité et aboutissant à un autre quelconques de ses pairs, reste entièrement dans la même entité.

La figure 5.2 illustre une vue d'ensemble du système. Au-dessus de l'*overlay* P2P initial, *l* couches *overlay*s secondaires sont organisées comme suit. Au niveau le plus élevé, se trouve les *overlay*s formés chacun par les pairs d'une même BE. C'est le niveau local. Les niveaux qui le séparent du niveau P2P global sont dits intermédiaires et regroupent des *overlay*s formés chacun par les pairs d'une même IE. Chaque niveau intermédiaire est caractérisé par le diamètre de ses IEs. Il s'agit du nombre de BEs qui forment les IEs en conformité avec les accords inter-opérateurs (ou politique des NPs). Ce nombre va croissant en passant d'un niveau secondaire à l'autre, allant du niveau le plus élevé au niveau le plus bas (l'*overlay* initial global). C'est l'algorithme de routage (e.g. Chord [SML+01], Pastry [RD01], etc.) mis en œuvre au niveau global qui sera repris dans chacun des *overlay*s intermédiaires.

Chaque pair est présent à tous les niveaux. Et chaque NP est responsable, à chaque niveau, de toutes les IEs basées sur les BEs qu'il gère. Une IE peut donc être gérée indépendamment par deux NPs différents. La cohérence des accords entre NPs, sur la base desquels les IEs sont formées, garantit la cohérence parfaite de la gestion assurée par chaque NP.

### *5.2.1 - Identification des entités*

Il est essentiel que des pairs provenant de deux BEs différentes et se retrouvant dans une même IE reconnaissent qu'il s'agit là de la même entité d'appartenance. Pour cela, nous supposons que chaque entité a un identifiant global unique, que nous appelons « vecteur identifiant » ou tout simplement : vecteur.

L'unicité globale des identifiants de chaque BE est tout à fait justifiée. Dans le cas où une BE se confond avec un AS, le vecteur de la BE sera l'ASN (*AS Number*).

Quant à l'identifiant d'une IE, il se présente sous la forme d'un vecteur (d'où la désignation généralisée) qui liste par ordre croissant tous les identifiants des différentes BEs qui forment la IE.

La figure 5.2 affiche le vecteur de chaque entité représentée. L'exemple illustré se focalise sur quatre BEs identifiées par les vecteurs "1", "2", "5", et "7". Il suppose les déclarations suivantes émanant de l'ensemble des accords signés par les différents NPs gérants de ces BEs :

- les connections P2P sortantes de la BE "1" doivent essayer de se satisfaire dans la BE "7" en première priorité (et vice-versa), avant d'essayer d'aboutir à des pairs des BEs "2" ou "5" en priorité *i* ;
- les connections P2P sortantes de la BE "2" doivent essayer de se satisfaire dans la BE "5" en première priorité (et vice-versa), avant d'essayer d'aboutir à des pairs des BEs "1" ou "7" en priorité *i*.

Par conséquent, au premier plan suivant le niveau local, les BEs de vecteur "1" et "7" fusionnent virtuellement en une même IE de vecteur "1,7", et les BEs de vecteur "2" et "5" fusionnent virtuellement en une même IE de vecteur "2,5". De même, au niveau intermédiaire de rang *i* à partir du plan local, les quatre BEs représentées fusionnent virtuellement en une même IE de vecteur "1,2,5,7".

Une BE peut donner aussi des priorités égales à deux autres BEs qui cependant ne se donnent pas la même priorité. Ceci peut notamment advenir dans le cas où les NPs sont des FAIs. En voici un exemple : les BEs "2", "5" et "6" (non représenté sur la figure) qui se regroupent en une IE "2,5" et une IE "5,6" au premier niveau intermédiaire et en la IE "2,5,6" au niveau suivant. Les pairs de la BE "5" seront donc représentés au premier niveau intermédiaire dans deux IEs différentes, mais fusionnent à nouveau en une

représentation unique au niveau qui suit. Dans la suite, nous référençons ce cas par la dénomination « représentation double ».

Chaque vecteur caractérise aussi le réseau *overlay* formé par les pairs d'une entité, ainsi que la DHT de l'espace d'identification correspondant. Cette nouvelle DHT référencie donc une partie des éléments (pairs et objets) de l'*overlay* global mais regroupés différemment. Elle est alors virtuelle par rapport à la DHT de base du plan P2P global. Elle sera appelée VDHT (pour *Virtual DHT*).

Afin de normaliser les vecteurs, leur taille peut être fixée à la longueur maximale possible. Dans ce cas, les vecteurs de longueur inférieure peuvent être compléter par des zéros à gauche. Par exemple, le vecteur "5" se transformera en un vecteur "0,…,0,5".

Par ailleurs, pour des soucis d'homogénéisation du système, le réseau P2P global peut se voir affecté un vecteur nul (c.-à-d. "0", ou "0,…,0" selon le cas).

### *5.2.2 - Identification des pairs*

Chaque pair a un *nodeId* dans chaque *overlay* où il existe, et donc autant de *nodeId*s que le nombre total d'*overlay*s auxquels il participe. Ceci est analogue à un terminal ayant plusieurs interfaces réseaux et donc autant d'adresses IP : une par réseau. Dans un souci de simplicité, nous n'exigeons pas d'unicité entre les différents espaces d'identification.

Par ailleurs, en observant de près l'architecture de NETPOPPS, nous remarquons que chaque BE forme avec les IEs où ses pairs se trouvent un sous-système hiérarchique géré par le NP propriétaire de la BE en question (cf. fig. 5.2 p. 99). (Dans le cas d'une « représentation double », la BE chapeaute deux sous-systèmes hiérarchiques.)

Les niveaux *overlay*s du système constituent alors un ensemble de plusieurs sous-systèmes hiérarchiques gérés par les NPs. Chaque NP gère autant de sous-systèmes qu'il gère de BEs. Chaque sous-

système est formé d'une BE au sommet d'autant d'IEs qu'il y a de niveaux intermédiaires dans le système.

Donc pour permettre une coopération effective du trafic P2P avec les politiques des NPs en matière de gestion du trafic, un mécanisme d'autogestion du trafic P2P serait nécessaire à l'intérieur de chaque sous-hiérarchie du système.

Par souci d'indépendance par rapport au protocole P2P, et comme nous abordons uniquement les systèmes utilisant des DHTs, il est utile d'introduire un mécanisme de gestion des DHTs. Par ailleurs, une hiérarchie isolée ressemble bien à l'illustration donnée par la figure 5.1 (cf. p. 97), où *n* correspondrait au nombre d'*overlay*s secondaires au dessus de l'*overlay* P2P global dans NETPOPPS.

Pour cela, le mécanisme de gestion des DHTs qui est introduit dans le système, dans chaque sous-hiérarchie, est basé sur le principe de dérivation de clés.

Par conséquent, les différents *nodeId*s de chaque pair seront en relation mathématique simple, comme suit : $Id_i = h(Id_{i-1}) = h^{i-1}(Id_1)$ et $Id_j = h^{j-i}(Id_i)$, où *j* et *i* sont des niveaux intermédiaires quelconques avec $j>i$ et $Id_1$ est le *nodeId* du pair dans l'*overlay* du BE (cf. paragraphe 5.1 p. 97). Cependant, comme NETPOPPS s'intègre à un système P2P existant, chaque pair va entrer dans le système à partir de l'*overlay* global, où il va alors acquérir son premier *nodeId*. Soit *k* ce *nodeId* que nous désignerons par « *nodeId* primaire ». Pour lier *k* à $Id_1$ à la manière des autres *nodeId*s du pair, une dérivation supplémentaire est insérée dans le processus, tel que $Id_1 = h(k)$. Il en découle pour tout niveau intermédiaire *i*, $Id_{,i} = h^i(k)$. (On a toujours $Id_j = h^{j-i}(Id_i)$ pour $j>i$) Cette variante du principe de la dérivation de clés est illustrée à la figure 5.3.

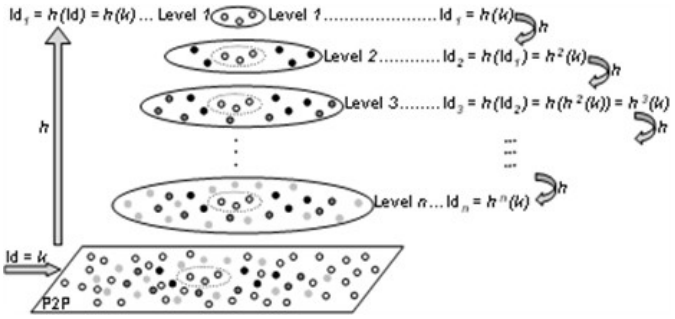


***Figure 5.3 :*** *Calcul des différents identifiants d'un même nœud*

Dans le cas d'une « représentation double », un pair aura à un niveau donné, le même *nodeId* dans chacune des hiérarchies.

### *5.2.3 - Identification des objets*

Comme un pair participe à plusieurs *overlay*s, il partage alors ses ressources dans chacun de ces *overlays*.

#### *5.2.3.1 - Approche générale*

Une même ressource partagée dans différents *overlay*s va avoir autant d'identifiants (*objectId*s) que le pair qui la partage. Ceci revient à donner plusieurs pseudonymes à un même objet. De même que pour les *nodeId*s, nous n'exigeons pas d'unicité entre les différents espaces d'identification.

Les différents *objectId*s sont en relation mathématique conforme au principe de la dérivation de clés adapté (tout comme les *nodeId*s). Un objet acquiert donc son premier identifiant dans l'*overlay* global : c'est l'« *objectId* primaire ». Et pour chaque objet, on a à tout niveau *i* intermédiaire : $Id_i = h(Id_{i-1}) = h^{i-1}(Id_1) = h^i(Id_{primaire})$.

Cependant avec NETPOPPS, un pair ne peut partager ses ressources qu'une fois qu'il aura adhéré complètement au système, c.-à-d. lorsqu'il se sera inséré dans chacun des différents *overlay*s auxquels il doit appartenir.

#### *5.2.3.2 - Approche particulière*

Au cas où tous les espaces d'identification du système sont de même taille, et comme ils sont de même nature sans aucune caractéristique distinctive qui impacte le calcul des différents identifiants, un objet peut être partagé avec le même *objectId* dans chaque *overlay*, d'autant plus que les objets sont passifs et gérés par les pairs.

Cette alternative permet à tout objet d'avoir un *objectId* unique dans tout le système. Ceci revient à utiliser pour tout le système (y compris dans l'*overlay* global) une DHT utilisant une fonction de hachage idempotente, c.-à-d. qui appliquée une, deux, ou même *n* fois à elle-même donne toujours le même résultat.

Ceci dit, une fonction de hachage idempotente pourrait bien aussi être utilisée uniquement pour les *overlay*s secondaires. Ce qui donnera à chaque objet un *objectId* primaire dans l'*overlay* initial (global) et un seul et même autre *objectId* à tous les autres niveaux.

## 5.3 - Mécanisme de routage dans NETPOPPS

Le mécanisme de routage est le même dans chaque *overlay*. Il s'agit de celui mis en œuvre au niveau P2P global (e.g. Chord [SML+01], cf. paragraphe 1.4.2 p. 44). Une différence s'installe toutefois d'un *overlay* à l'autre : les *nodeId*s utilisés, les *objectId*s (s'ils sont différents), ainsi que la DHT (ou VDHT) consultée.

Quand un pair veut rechercher un objet, il commence par calculer son *objectId* primaire qu'il garde en antémémoire le temps de la requête pour le cas où il aura à chercher dans l'*overlay* global. Puis il dérive l'*objectId* primaire et utilise cette nouvelle clé dans sa requête qu'il lance dans l'*overlay* local où il réside en s'identifiant par son *nodeId* correspondant et

en utilisant la VDHT correspondante. Ainsi la requête n'atteint que des pairs de son BE.

Si la requête n'aboutit pas, elle est alors relancée dans l'*overlay* de l'IE suivant au niveau strictement inférieur de la hiérarchie, avec les DHT, *nodeId* et éventuellement *objectId* correspondants. Et ainsi de suite, jusqu'à satisfaction de la requête.

Ce n'est qu'en dernier lieu, qu'une requête est lancée dans l'*overlay* global, si elle n'a pas encore abouti. À ce niveau, ce sont les services de la DHT globale qui sont utilisés par le *nodeId* primaire pour trouver l'objet identifié, selon le cas (approche particulière ou générale des espaces d'identification), par le même *objectId* ou par l'*objectId* primaire gardé en antémémoire. Il est toutefois rare d'atteindre l'*overlay* global si le système est bien peuplé et la charge y est bien équilibrée.

Il est à noter que seul le pair initiateur d'une requête peut relancer (à un niveau inférieur) une requête non aboutie. Aussi, un pair reconnaît la VDHT qu'il doit utiliser grâce au *nodeId* par lequel il a été sollicité.

Dans le cas d'une « représentation double » le pair va se retrouver avec deux VDHTs à un même niveau intermédiaire. Alors lorsqu'il sera sollicité dans l'un des *overlay*s où il se trouve, et afin d'éviter tout conflit, il va demander à celui qui le sollicite de lui donner le vecteur de son *overlay* (que l'autre reconnaît de par son *nodeId* ou la VDHT qu'il utilise). Si par contre, un des pairs de la BE intégrée dans deux IEs à un même niveau initie une requête, le mécanisme de routage se poursuit à l'identique et simultanément dans les deux hiérarchies que la BE chapeaute.

Enfin, quand une ressource est trouvée dans un *overlay* non local (c.-à-d. global ou intermédiaire), le pair qui la récupère la remet en partage dans son propre *overlay* local, tout comme il y partagerait une nouvelle ressource.

## 5.4 - Mouvement des pairs dans NETPOPPS

Les mouvements d'arrivée et de départ d'un pair sont gérés de façon identique dans tous les *overlay*s du système et conformément au protocole

de routage mis en œuvre dans l'*overlay* global. Cependant après s'être joint à l'*overlay* global et avant de pouvoir se joindre à n'importe lequel des *overlay*s secondaires, un pair *n* doit effectuer certaines opérations. En effet, il doit identifier ces *overlay*s et connaître au moins un autre pair $n_1$ actif dans chacun d'eux. Ce qui l'amènerait à devoir connaître autant d'autres pairs distincts que d'*overlay*s secondaires où il cherche à s'insérer. Mais étant orienté opérateur de réseaux, NETPOPPS propose de ramener tous ces différents pairs à un seul nœud géré par le NP. Ce nœud sera un nœud de référence pour les nouveaux pairs lors de leur adhésion au système. Il sera appelé nœud de contrôle (ou *CN* pour Control Node).

### *5.4.1 - Le nœud de contrôle*

Chaque NP insère dans le système un CN par BE qu'il gère. Le CN se retrouve alors dans toutes les autres entités (IE) de la sous-hiérarchie chapeautée par la BE. Par souci de robustesse, un nœud miroir peut aussi exister pour chaque CN.

Chaque NP est libre de définir les fonctionnalités qu'il souhaite déployer sur chacun des CNs qu'il gère. Un CN peut donc avoir plusieurs rôles ; en particulier, il détient un profil sauvegardé et géré par le NP. Chaque NP est aussi libre de définir et sauvegarder dans ce profil les données qui l'intéressent. En particulier, le profil mémorise les identifiants des entités (c.-à-d. les vecteurs de la BE et des IEs) dans lesquelles le CN réside. L'amplitude et diversification des autres fonctionnalités, ainsi que les détails du restant des données du profil n'affectent ni le mécanisme de routage P2P ni les opérations des pairs. Nous ne les abordons donc pas.

Le CN est présent dans chaque *overlay* correspondant à une entité où il réside. Il a donc plusieurs identifiants. Comme tout pair, il entre dans le système à partir de l'*overlay* global, où il acquiert son *nodeId* primaire, $Id_{CN,ref}$. Et ses différents identifiants sont aussi en relation mathématique conforme au principe de la dérivation de clé modifié (cf. figure 5.3 p. 103). Quand le niveau de rang 1 est celui de

la BE, on a donc pour tous niveaux intermédiaires de rang *i* et de rang *j>i*, $Id_{CN,i} = h^{i}(Id_{CN,ref})$ et $Id_{CN,j} = h^{j-i}(Id_{CN,i})$.

Ainsi lorsque le CN a acquis son $Id_{CN,ref}$ il calcule ses différents identifiants. Et à chaque nouvel identifiant calculé, il initie l'*overlay* correspondant conformément au profil. Quand le CN a réussi les initialisations des différents *overlay*s le concernant, il génère une table de référence (ou *RT* pour *Reference Table*).

### *5.4.2 - La table de référence*

La RT est une table qui récapitule tous les vecteurs des différentes entités d'appartenance du CN. Dans le cas d'une « représentation double », la RT sera une structure matricielle dont chaque ligne sera la RT relative à l'une des hiérarchies concernées.

La RT permet donc aux nœuds du même BE que le CN qui arrivent dans le système de connaître les différents IEs de la hiérarchie. Quand le CN crée la RT, il la met donc en partage dans l'*overlay* global. Elle y est gérée comme un objet, et par souci de robustesse, peut être répliquée dans ce même *overlay*, en particulier, elle reste stockée sur le CN.

Comme chaque BE a un vecteur unique, qu'il y a un CN par BE et une RT par CN, la RT est unique pour chaque BE. Chaque RT peut donc être identifiée de façon unique dans le système par *h(vector_of_BE)*. Il y a donc dans le système autant de RTs que de CNs et que de BEs.

La structure de la RT diffère cependant selon le degré d'interaction entre le NP et les pairs se joignant au système. Comme la RT est générée par le CN qui est géré par le NP, le choix de la mise en œuvre de l'une ou l'autre des structures de la RT revient au NP. On distingue entre un système passif et un système actif. Le choix de cette dénomination s'éclaircira à la présentation du mécanisme d'adhésion d'un nouveau pair au système (cf. paragraphe 5.4.3 p. 110).

#### *5.4.2.1- Système passif*

La structure de la RT en mode dit passif est donnée par le tableau 5.1. C'est une table à deux entrées et (*N+1*) colonnes, où *N* est le nombre de niveaux intermédiaires dans le système, et donc de IEs dans la hiérarchie considérée.

***Tableau 5.1 :*** *Structure de la RT en mode passif*

| *vector of the BE* | *vector of the 1st IE* | … | *vector of the Nth IE* |
|---|---|---|---|
| *local ID of CN* | *CN's ID in the 1st IE* | … | *CN's ID in the Nth IE* |

Les différents champs de cette RT sont les suivants :

- à la première ligne de la première colonne se trouve le vecteur de la BE du CN (*vector of the BE*) ;
- à la deuxième ligne de la première colonne se trouve l'identifiant local du CN, c.-à-d. le *nodeId* du CN dans la BE (*local ID of CN* = $Id_{CN,1}$) ;
- pareillement, les autres colonnes donnent à la première ligne le vecteur d'une IE et à la deuxième ligne le *nodeId* du CN dans cette IE. Les colonnes se suivent dans l'ordre de superposition des IEs, en allant de la première IE strictement en dessous de la BE, pour la deuxième colonne, à la IE strictement au dessus du plan P2P global, pour la dernière colonne.

#### *5.4.2.2- Système actif*

La structure de la RT en mode dit actif est donnée par le tableau 5.2. C'est une table à une seule entrée et (*N+2*) colonnes, où *N* est le nombre de niveaux intermédiaires dans le système, et donc de IEs dans la hiérarchie considérée.

***Tableau 5.2 :*** *Structure de la RT en mode actif*

| *Vector of the BE* | *primary ID of the CN* | *vector of the 1st IE* | … | *Vector of the Nth IE* |
|---|---|---|---|---|

Les différents champs de cette RT sont les suivants :

- le premier champ donne le vecteur de la BE du CN ;
- le deuxième champ donne le *nodeId* primaire du CN, $Id_{CN,ref}$ ;
- les autres champs contiennent successivement les vecteurs de toutes les IEs, en allant de la première IE strictement en dessous de la BE, pour la troisième colonne, à la IE strictement au dessus du plan P2P global, pour la dernière colonne.

### *5.4.3 - Arrivée d'un pair*

Un nouveau pair arrive dans le système en se joignant à l'*overlay* global, où il acquiert son *nodeId* primaire. Pour pouvoir se joindre correctement aux bons *overlay*s secondaires, il doit maintenant connaître les différents identifiants du CN de sa BE d'appartenance. Pour cela il doit retrouver la RT qui lui donnera les informations utiles.

Par un mécanisme externe à NETPOPPS, chaque pair peut connaître sa BE d'appartenance. Il lui faudra pour cela se servir des bases d'information du système (e.g. des tables BGP [RFC1771] pour avoir les numéros des ASs d'appartenance). Ce type de mécanisme n'est pas discuté dans la suite. NETPOPPS s'intéresse principalement aux mécanismes de routage et aux mouvements des pairs.

Quand le nouveau pair reconnaît sa BE d'appartenance, il connaît le vecteur de celle-ci. Il calcule alors l'identifiant de la BE dans l'*overlay* global, c.-à-d. *h(vector_of_BE)*.Cet identifiant étant aussi celui de la RT (dans l'*overlay* global), le nouveau pair initie une requête *reference_table_request( )* au niveau global pour

récupérer la RT. Deux cas sont alors possibles suivant le degré d'interaction entre le NP et les pairs se joignant au système.

(Dans le cas d'une « représentation double », la RT récupérée sera une matrice dont chaque ligne est une RT pour l'une des hiérarchies. Le mécanisme d'adhésion se poursuit alors simultanément dans chaque hiérarchie comme décrit ci-dessous)

#### *5.4.3.1 - Système passif*

Dans le cas passif, lorsque le nouveau pair récupère la RT, il connaît tous les vecteurs des différentes entités où il réside logiquement, ainsi que les différents *nodeId*s du CN de sa BE dans chacun de ces *overlay*s.

Le nouveau pair calcule alors ses différents identifiants selon le principe adapté de la dérivation de clé (cf. paragraphe 5.2.2 p. 102). Et à chaque nouvel identifiant calculé, il fait la correspondance avec les données fournies par la RT en avançant d'une colonne. Ainsi le nouvel identifiant du nouveau pair correspond à la nouvelle entité identifiée. Et grâce au *nodeId* correspondant du CN, le nouveau pair se joint à l'*overlay* correspondant.

Par exemple, si le protocole mis en œuvre au niveau global est Chord ([SML+01], cf. paragraphe 1.4.2 p. 44), et que le *nodeId* primaire du nouveau pair est *n* et celui du CN de sa BE $Id_{CN}$, l'opération d'adhésion à un niveau intermédiaire *i* s'exécute à travers la fonction $Id_i.join(Id_{CN,i}) = [h^i(n)].join[h^i(Id_{CN})]$.

#### *5.4.3.2 - Système actif*

Dans le cas actif, lorsque le nouveau pair récupère la RT, il prend connaissance de tous les vecteurs des différentes entités où il réside logiquement, ainsi que de $Id_{CN,ref}$, le *nodeId* primaire du CN de sa BE.

Il y a là deux mises en œuvre possibles de NETPOPPS selon que le NP ait ou pas un plan d'action à opérer sur les nœuds se joignant aux différents *overlay*s qu'il gère.

(Dans le cas d'une « représentation double » la mise en œuvre sera pareille pour toutes les hiérarchies issues de la BE.)

Dans les deux cas, une fois que le nouveau pair connaît les différents *nodeId*s du CN de sa BE dans chacun des *overlay*s secondaires, le mécanisme de son adhésion à ces différents *overlay*s se poursuit comme dans le cas d'un système passif.

*a) Le NP a défini un plan d'action*

Dans l'affirmative, le nouveau pair envoie au CN dans le plan global une requête d'intention d'adhésion à chacun des *overlay*s secondaires dont il a pris connaissance du vecteur. Le CN exécute alors le plan d'action prévu par le NP. Il peut par exemple opérer un contrôle d'accès en vérifiant les politiques d'accès. Il peut aussi appliquer un mécanisme propriétaire quelconque (e.g. pour un service payant, pour une récupération de données à des fins statistiques, etc.). Une fois l'opération réussie, le CN envoie son *nodeId* dans l'*overlay* secondaire en question.

L'intention d'adhésion est exprimée par des requêtes indépendantes pour chaque *overlay*. Mais elle peut bien être exprimée aussi par une seule requête incluant tous les vecteurs. Selon le cas, le CN renverra ses *nodeId*s secondaires individuellement, en réponse à chaque requête concernée, ou en bloc, si la requête est globale, mais dans le même ordre de classement des vecteurs dans la RT.

*b) Le NP n'a pas défini de plan d'action*

Si le NP n'entend pas appliquer de mécanisme particulier à ses nœuds entrant dans le système P2P, il peut toujours mettre en œuvre NETPOPPS de façon à ce que le nouveau pair calcule lui-même les différents identifiants du CN par dérivations successives de $Id_{CN,ref}$ conformément au principe de la dérivation de clé.

### *5.4.4 - Disparition d'un pair*

La disparition d'un pair du système (départ explicite, départ silencieux, ou panne) est généralement remarquée quasi-simultanément dans tous les *overlay*s auxquels il appartenait. Elle est traitée de façon identique mais indépendante dans chaque *overlay*, et conformément au protocole mis en œuvre au niveau global.

## 5.5 - Analyse de NETPOPPS

L'application du principe de la dérivation de clé aux identifiants d'un réseau P2P (principalement les *nodeId*s) permet la conception d'un système orienté opérateur de réseaux où le trafic P2P coopère avec les accords économiques de routage inter-opérateurs.

Pour la dérivation des identifiants, NETPOPPS peut utiliser des fonctions injectives différentes de la fonction de hachage utilisée au niveau global. Il peut s'agir d'une même fonction à tous les niveaux secondaires ; mais il peut aussi s'agir de plusieurs fonctions à condition qu'au niveau strictement supérieur au niveau global toutes les BEs d'une même IE utilisent la même fonction. Dans les deux cas, c'est le CN qui communique (par le profil) la fonction de dérivation à tout nouveau pair qui le contacte lors de son arrivée au niveau global.

Ce mécanisme de gestion des identifiants (par le principe de la dérivation de clé) permet de structurer un réseau P2P en des couches secondaires virtuelles conformes à un critère ou l'autre des accords inter-

opérateurs. Ceci permet un cloisonnement de ressources conforme à des critères économiques.

Le réseau P2P initial étant formé de pairs d'appartenance assez variée se transforme ainsi en une collection de sous-systèmes hiérarchiques aux niveaux uniformes, tel que chaque hiérarchie est gérée par un opérateur.

Chaque pair est ainsi logiquement présent à chaque niveau à travers des *nodeId*s différents, donc des pairs virtuels. De même, chaque objet est présent dans chaque *overlay* où réside le pair qui le stocke. Cette présence se manifeste à travers des *objectId*s différents. Néanmoins dans certains cas, un objet peut garder un même *objectId* dans tout le système puisque les objets sont passifs. Ce qui, en toute évidence, allège le système et contribue à diminuer les temps de latence et accélérer les procédures de recherche et de récupération des objets.

Chaque pair (ou objet) acquiert son premier identifiant (l'identifiant primaire) au niveau global. Comme la fonction de dérivation est injective, on ne trouve donc pas dans le plan local le même *nodeId* (ou *objectId*) dans deux BEs distinctes. De même, il n'y a pas deux *nodeId*s (ou *objectId*s) identiques dans deux IEs distinctes d'un même plan, à moins que le pair soit représenté deux fois.

Ainsi, si dans une des IEs, un *nodeId* (ou *objectId*) existe déjà, c'est qu'il s'agit du même pair (ou objet). Les représentations virtuelles fusionnent alors (avec le regroupement des objets à la charge de chaque représentation sur le même pair). C'est le cas des représentations doubles.

Chacune des couches secondaires créées virtuellement (par le principe de la dérivation de clé) a son propre espace d'identification et donc une DHT correspondante qui est virtuelle et dite VDHT. Celle-ci permet une gestion et une recherche circonscrites des données.

En effet, la recherche d'une ressource ou d'un élément de données dans NETPOPPS commence toujours dans l'*overlay* local (de la BE) en utilisant les services de la VDHT correspondante. Et en cas d'échec, la requête est relancée dans l'entité (IE) strictement suivante hiérarchiquement. Or, chaque entité est définie conformément à des critères d'intérêt pour le

NP et de sorte qu'un chemin entre deux de ses pairs ne passe pas par l'extérieur. De plus, chacune a un vecteur unique qui caractérise aussi sa DHT. Et chaque VDHT ne référencie que les identifiants des nœuds et objets dans l'entité. D'ailleurs chaque pair a un identifiant pour chaque *overlay* où il réside. (Le routage dans le cas d'une « représentation double » a été présenté au paragraphe 5.3 p. 104.) L'identifiant de chaque objet est aussi unique dans chaque *overlay* : si ceci est clair dans l'approche générale, il l'est tout aussi dans l'approche particulière qui peut être un cas particulier idempotent du cas général (cf. paragraphe 5.2.3.1 p. 104). En tout cas, un *objectId* ne peut être référencé dans une VDHT que si l'objet est bien dans l'*overlay* correspondant, mais aussi le pair qui a la charge de l'objet.

Ainsi, toute requête qui aboutit dans l'*overlay* où elle a été initiée est garante de l'existence de l'objet dans cet *overlay*. Il en découle donc que le pair initiateur de la requête peut récupérer l'objet d'un pair répondant à des critères inter-opérateurs identiques et ce, à travers un chemin conforme à ce(s) critère(s) ne passant pas par une entité (IE) plus large.

Quant au CN, c'est un nœud de contrôle appartenant au NP. Donc à moins qu'il y ait une panne, il ne quittera pas le système : c'est un nœud quasi-statique.

Le CN n'est pas un super-pair : il ne joue pas de rôle central par rapport aux pairs de son entité, et il n'a aucune priorité dans le routage. Il sert de pont entre le NP et ceux des nœuds du même BE joignant le système P2P global :

– d'une part, il permet au NP de cadrer le trafic P2P et de le libérer graduellement (d'où les sous-systèmes hiérarchiques) conformément aux accords inter-opérateurs. Le CN permet aussi au NP d'exercer un contrôle sur ses pairs ou de leur offrir l'accès à un service propriétaire (cf. paragraphe 5.4.3.2-a p. 112) ;
– d'autre part, le CN permet aux pairs d'une même BE de se former correctement en réseaux *overlay*s secondaires correspondants à la BE et à chacune des IEs englobant cette BE. Pour cela, il se manifeste auprès

des pairs se joignant à l'*overlay* global, à travers la RT qu'il génère et met à disposition dans le réseau P2P initial (c.-à-d. global).

Comme chaque IE est formée de plusieurs BEs et peut être gérée par plusieurs NPs, alors il y réside autant de CNs différents que ses BEs de base, et ces CNs appartiennent à au plus autant de NPs différents. Tous ces CNs offrent alors les mêmes fonctionnalités et contiennent les mêmes données vitales à l'unité de l'IE. Ces exactitudes proviennent de la concordance des accords inter-opérateurs. Toutefois, cela n'empêche pas chaque CN d'offrir des fonctionnalités à seule destination des propres pairs de sa BE, ni de stocker des données concernant uniquement ces pairs-ci.

Par ailleurs, la définition et la négociation des accords inter-opérateurs peuvent être une charge quasi-permanente pour un NP, et le sont particulièrement si le NP est un FAI. Si les termes de ces accords viennent donc à être modifiés, la mise à jour de certaines données du profil d'un CN peut s'avérer nécessaire. C'est le cas où une modification des accords inter-opérateurs porterait sur la modification de la composition des IEs ou la création de nouvelles IEs. Dans les deux cas, il s'agit d'une redistribution des associations entre les différentes BEs au niveau hiérarchique des IEs concernées. Mais une modification de IEs existantes ne concernera pas toutes les BEs qui la composent. Tous les pairs de cette IE ne seront donc pas concernés par ce remaniement. En tout cas, la mise à jour du profil sur le CN est à la charge du NP. Et dès qu'elle est effectuée, le CN génère une nouvelle RT qui remplace la RT existante.

Comme la modification des accords inter-opérateurs ne modifie pas le vecteur d'une BE existante, l'identifiant de la RT reste donc inchangé et la RT reste trouvable. Elle sera donc mise à jour en gardant le même identifiant. Les différents identifiants du CN restent aussi inchangés puisqu'ils sont tous dérivés à partir de son *nodeId* primaire. Il en est de même pour les différents *nodeId*s d'un pair qui n'a pas quitté le système, tout comme des différents *objectId*s d'un même objet encore disponible.

Quand le CN a donc mis à jour sa RT, il va en informer les pairs de sa BE par un message de diffusion dans l'overlay de la BE. Ce message va

permettre aux pairs de se déconnecter de leurs anciennes IEs d'appartenance et de se joindre aux nouvelles. Mais dans un système dynamique, tous les pairs vont à un moment ou l'autre quitter le système avant de peut-être le rejoindre. Alors cette nouvelle adhésion se fait évidemment conformément aux mises à jour effectuées, puisque le nouveau pair en récupérant la RT y trouvera les informations mises à jour. Cette opération de remaniement (partiel) pourrait rendre certains *objet*s rares momentanément indisponibles à certains niveaux secondaires.

Au cas où la modification des accords inter-NPs est due à l'émergence d'une nouvelle BE, cette émergence se manifestera par l'insertion dans le système d'un nouveau CN. Celui-ci initiera le sous-système hiérarchique correspondant et génèrera une RT correspondante qu'il mettra à disposition dans l'*overlay* global. Par ailleurs, comme chaque nœud se trouve dans une seule BE à la fois, alors si les nœuds de la nouvelle BE sont initialement dans une autre BE existante, ils ne le seront plus à présent et auront donc forcément déjà quittés le système P2P. Et s'ils veulent le rejoindre, ils le feront à partir de leur nouvelle BE d'appartenance (et ils vont donc chercher la nouvelle RT correspondante).

En garantissant la récupération d'un objet de l'entité où il a été trouvé, NETPOPPS offre un cloisonnement des ressources efficace et conforme aux intérêts d'un opérateur avec une gestion optimale des différents identifiants. Le cloisonnement réduit le temps de latence des messages et améliore le temps de récupération des données. Ces performances sont plus remarquables avec la possibilité qu'a un objet de garder un identifiant unique dans tout le système. De plus, la mise en partage dans l'*overlay* d'une entité locale d'une ressource récupérée à un autre niveau (intermédiaire ou global) plus large favorise la popularisation des objets dans le système.

Enfin, NETPOPPS offre à un opérateur la possibilité d'opérer un contrôle d'accès sur les pairs utilisant ses réseaux ou de leur offrir ses services propriétaires (qui peuvent être par exemple des services privilégiés et/ou payant). Ceci est techniquement réalisable grâce à la structure

minimale de la RT dans le cas d'un système actif. Outre les vecteurs des différentes entités d'appartenance, cette structure ne fournit que le *nodeId* primaire du CN.

CONCLUSION DU CHAPITRE –

*Ce chapitre a défini, spécifié et analysé NETPOPPS (NETwork Provider Oriented P2P System), un système P2P utilisant des DHTs et orienté opérateur de réseau. Ce système permet une coopération entre, d'une part, les flux P2P et, d'autre part, la topologie sous-jacente et les politiques de l'opérateur. NETPOPPS veille aussi à simplifier la gestion des différents identifiants d'un même élément (pair ou objet), en créant entre eux une relation mathématique simple, grâce à la technique de dérivation de clés, qui a été présentée au début du chapitre.*

*Le chapitre suivant établit une comparaison analytique et applicative de NETPOPPS avec CAP.*

# CHAPITRE 6

## *COMPARAISON ANALYTIQUE DE CAP ET NETPOPPS*

*Ce chapitre établit une comparaison analytique de nos deux contributions, présentés et analysés individuellement aux chapitres précédents. Il s'agit de CAP (Context-Aware P2p system) et NETPOPPS (NETwork Provider Oriented P2P System)*

### 6.1 Analyse comparative et applicative des contributions

Les deux contributions principales de ce mémoire, CAP et NETPOPPS ont été analysé individuellement, dans chacun des chapitres qui leur ont été respectivement dédiés. Dans ce paragraphe, nous abordons une analyse comparative et applicative des deux systèmes.

Nous commençons par présenter les points communs aux deux propositions. En fait, toutes les deux :

- traitent les identifiants (des pairs et des objets), en tenant compte de caractéristiques du réseau sous-jacent, et assurent un équilibre gagnant-gagnant entre les usagers du P2P et les opérateurs du réseau ;
- s'appliquent à toute DHT mise en œuvre dans un réseau P2P existant ;
- exploitent la fonction injective de hachage de la DHT existante ;
- aboutissent à un système multi-niveaux, en définissant des couches secondaires regroupant les pairs en sous-ensembles suivant une ou plusieurs caractéristiques du réseau sous-jacent (ces sous-ensembles sont dits '*zone*' dans CAP et '*entité*' dans NETPOPPS), et dans l'overlay correspondant à chaque sous-ensemble, il y a donc une DHT virtuelle (ou VDHT) qui se forme ;

- garantissent la récupération d'un objet du sous-ensemble ('*zone*' ou '*entité*') où il a été trouvé et que le chemin de connexion reste entièrement dans le même sous-ensemble ;
- offrent un cloisonnement des ressources, qui réduit le temps de latence des messages et améliore le temps de récupération des données ;
- favorisent la popularisation des données, en les repartageant dans le sous-ensemble local du pair initiateur de la requête, si la requête a été satisfaite dans un autre ensemble (intermédiaire ou global) ;
- s'adaptent à l'utilisation qui en est faite, de par leur généricité.

Cependant, des différences d'optique sont claires entre les deux systèmes : CAP a une orientation utilisateur, alors que NETPOPPS est bien plus conforme aux intérêts, même économiques, de l'opérateur du réseau.

En effet, CAP est configurable et extensible. Le propriétaire d'une ressource peut définir une politique de partage de cette ressource. Chaque pair n'appartient pas nécessairement à un autre *overlay* que l'*overlay* global, et a fortiori n'appartient pas à toutes les zones du système. Chaque zone est indépendante et tout pair peut créer et initier une nouvelle zone conformément à la métrique de performance qu'il souhaite. La garantie de la découverte d'une ressource en un nombre de sauts minimal est donnée par la DHT. Mais la garantie de la sensibilité contextuelle du routage est donnée par le cloisonnement contextuel des ressources et la sémantique conséquente de l'*objectId* à travers la *Hkey*. CAP peut donc être vu comme un système cœur de réseau au-dessus duquel diverses applications peuvent être développées offrant des services à l'usager via une interface conviviale. Par exemple, l'usager pourra choisir et modifier explicitement la valeur de *Hkey*. Celle-ci peut aussi permettre la création explicite de groupes de communication sécurisés ou avec une qualité de service privilégiée. Ces groupes seront alors autogérés sans qu'il y ait besoin d'une entité de contrôle pour en vérifier l'accès. Aussi grâce à la *Hkey*, CAP peut être aussi adapté pour router des requêtes complexes qui s'exécuteront en parallèle dans plusieurs zones.

Cependant, dans CAP, les différents identifiants d'un même élément (pair ou objet) sont indépendants, et un même élément doit gérer autant d'identifiants que de zones où il réside.

Quant à NETPOPPS, il simplifie considérablement la gestion des identifiants. Chaque élément a ses identifiants en relation mathématique simple entre eux. Et un objet peut même garder un même et seul identifiant dans tout le système, si les différents espaces d'identification du système sont de même taille. Ce qui améliore encore plus les performances (diminue les temps de latence et recherche d'un objet, et donc de réponse).

Mais dans NETPOPPS, le cloisonnement des ressources est conforme aux intérêts de l'opérateur du réseau. Un usager ne peut pas décider de la création d'un *overlay* secondaire, donc non plus d'un groupe de communication, surtout avec des usagers d'autres BEs. La structure du système est fixée par l'opérateur qui gère le réseau de façon quasi-rigide, que seule la renégociation des accords inter-opérateur peut modifier (ou la politique intra-opérateur dans le cas où un opérateur gère plusieurs BEs).

De plus, à travers le nœud de contrôle, NETPOPPS offre à l'opérateur du réseau la possibilité d'exercer un contrôle d'accès sur les pairs utilisant ses réseaux ou de leur offrir ses services propriétaires (e.g. des services privilégiés et/ou payant). La porte est ainsi large ouverte à des services commerciaux, des mesures statistiques, etc.

Ainsi, NETPOPPS permet activement de profiter des importantes économies que peuvent générer les architectures P2P dans la gestion des réseaux. Alors que dans CAP, ceci reste une conséquence.

Enfin, les deux systèmes permettent de développer toute une panoplie de nouveaux services, mais dans NETPOPPS, ceci reste le produit de l'opérateur, selon sa volonté et sous son contrôle.

NETPOPPS, comme la solution de service intermédiaire (*oracle* service) [AFS07] et comme P4P [XKS$^+$07], met en avant le rôle qu'un opérateur de réseau peut prendre dans l'adéquation d'un réseau P2P à la topologie sous-jacente, l'optimisation du trafic P2P et des différents coûts, et la performance des applications. Cependant, les opérateurs de réseaux,

notamment les ISPs considèrent généralement la topologie des réseaux qu'ils contrôlent comme étant des informations confidentielles [MG08]. Par conséquent, pour réussir à être adopté à très large échelle, une solution de sensibilisation du trafic P2P à la topologie sous-jacente doit fournir une méthode qui aide les applications dans la sélection des pairs sans qu'il y ait besoin de divulguer explicitement la topologie du réseau sous-jacent. Dans cette perspective, CAP est prometteur, en tant que système générique, flexible et orienté utilisateur.

Toutefois, comme dans tout domaine de recherche applicative, une large adoption à très grande échelle d'un système P2P sensible à la structure sous-jacente ne verra probablement pas le jour sans un accord entre les principaux acteurs sur une solution commune basée sur des standards libres de droit [MG08].

## 6.2 Exemple de l'incidence positive des contributions sur l'efficacité du routage

Nos deux contributions (CAP et NETPOPPS) proposent des mécanismes de regroupement des pairs, et donc de formation d'entités, suivant un modèle économique donné, pour adapter le flux P2P au modèle économique considéré.

Nous avons aussi dans nos deux contributions qu'un pair physique A est identifié par un *nodeId* $n_1$ au niveau global, et par un *nodeId* $n_2$ à un niveau local. Les deux nœuds d'identifiants respectifs $n_1$ et $n_2$ sont des représentations logiques du nœud physique A. Il en est de même pour tout autre pair physique. Cela signifie que le nœud $n_1$ stocke deux DHTs :

- une DHT au niveau global : celle d'un pair physique A ;
- et une DHT au niveau local : celle d'un pair physique B.

Il en découle que quand un pair A consulte sa VDHT (et donc change pour cela de *nodeId*), c'est comme s'il voyait la DHT à ce nouveau niveau d'un autre pair B se trouvant dans la même zone/entité locale que lui.

C'est le profil du système qui nous dira que (par exemple) le pair physique A va se retrouver au plan local dans la même zone/entité que le pair physique B. Mais c'est la construction des identifiants qui nous donnera les différents *nodeId*s d'un pair physique A : nous saurons alors si le pair A – qui, au plan global, aura par exemple le *nodeId* 0 – aura au plan local le *nodeId* 0 ou 2 ou autre.

Par souci de clarté, nous nous positionnons face à un système à un seul niveau secondaire au-dessus du niveau P2P global (cf. fig. 6.1) :

- avec CAP, ceci revient à nous positionner dans le cas où le profil du système définit une seule *Hkey* (qu'elle soit donc simple ou composée, mais pas de *Hkey* dérivée) (cf. p. 88) ;
- avec NETPOPPS, ceci revient à nous positionner dans un cas où il n'y a pas de IEs (cf. p. 99).

Et par souci de simplicité, nous nous limiterons au niveau local à deux entités (ou *zone*, selon qu'il s'agisse de NETPOPPS ou CAP).

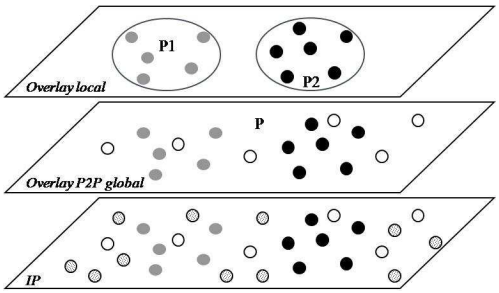


***Figure 6.1 :*** *Vue d'ensemble du système analysé*

Comme répété plus haut, c'est la méthode de construction des identifiants secondaires qui nous donne les différents *nodeId*s d'un pair physique quelconque. Cette méthode diffère entre nos deux contributions :

- dans NETPOPPS, elle utilise la méthode de dérivation de clés. Et l'on y a distingué deux cas selon qu'un même objet possède un seul ou plusieurs identifiants dans le système, c.-à-d. selon que la fonction injective utilisée pour les dérivations de l'*objectId* primaire est idempotente ou pas (cf. paragraphe 5.2.3 page 104) ;

– dans CAP, la méthode de construction des identifiants secondaires utilise la fonction HMAC (avec une clé *Hkey* et la fonction de hachage du protocole mis en œuvre), et les identifiants d'un même élément (pair ou objet) sont tous différents et indépendants les uns des autres.

Dans la suite, et par souci de simplicité et de clarté, nous considérons le système analysé comme mettant en œuvre NETPOPPS.

Par ailleurs, nous nous limiterons dans la suite à un espace d'identification de huit éléments, avec sept pairs physiques actifs dans le système, et dont les *nodeId*s forment l'ensemble {0,1,2,3,4,5,7}. Tous ces nœuds sont donc présents dans l'*overlay* global. Nous considérons que parmi ces nœuds :

– ceux dont l'identifiant fait partie de l'ensemble {0,2,3} sont groupés dans l'entité P1 ;
– et ceux dont l'identifiant fait partie de l'ensemble {1,4,7} sont groupés dans l'entité P2.

Aussi considérons-nous que notre système met en œuvre le protocole Chord ([SML+01], cf. paragraphe 1.4.2 p. 44).

Dans le cas où les pairs et objets gardent le même identifiant à tous les niveaux (c.-à-d. que la dérivation des identifiants primaires s'applique avec une fonction injective idempotente), la représentation logique de notre système est donc celle représentée par la figure 6.2.

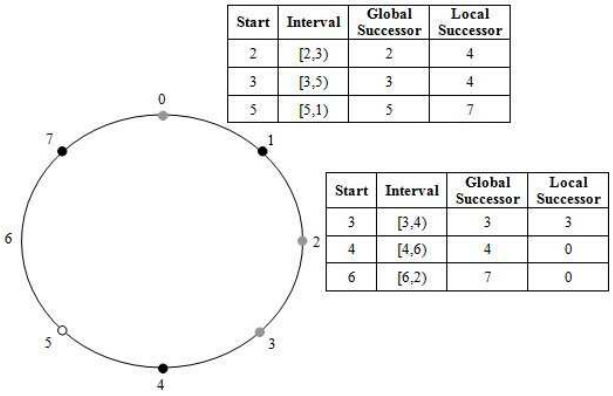


***Figure 6.2 :*** *Représentation logique du système étudié*

En considérant que chaque élément (pair ou objet) garde le même identifiant à tous les niveaux d'*overlay*s, les DHTs logiques du système sont

alors comme données par le tableau 6.1 où *Ni* représente le nœud logique d'identifiant *i*.

***Tableau 6.1 :*** *Vue globale des DHTs du système (avec une fonction idempotente)*

*N0*

| Start | Interval | Global Successor | Local Successor |
|---|---|---|---|
| 1 | [1,2) | 1 | 2 |
| 2 | [2,4) | 2 | 2 |
| 4 | [4,0) | 4 | 0 |

*N1*

| Start | Interval | Global Successor | Local Successor |
|---|---|---|---|
| 2 | [2,3) | 2 | 4 |
| 3 | [3,5) | 3 | 4 |
| 5 | [5,1) | 5 | 7 |

*N2*

| Start | Interval | Global Successor | Local Successor |
|---|---|---|---|
| 3 | [3,4) | 3 | 3 |
| 4 | [4,6) | 4 | 0 |
| 6 | [6,2) | 7 | 0 |

*N3*

| Start | Interval | Global Successor | Local Successor |
|---|---|---|---|
| 4 | [4,5) | 4 | 0 |
| 5 | [5,7) | 5 | 0 |
| 7 | [7,3) | 7 | 0 |

*N4*

| Start | Interval | Global Successor | Local Successor |
|---|---|---|---|
| 5 | [5,6) | 5 | 7 |
| 6 | [6,0) | 7 | 7 |
| 0 | [0,4) | 0 | 1 |

*N5*

| Start | Interval | Global Successor | Local Successor |
|---|---|---|---|
| 6 | [6,7) | 7 | |
| 7 | [7,1) | 7 | |
| 1 | [1,5) | 1 | |

*N6*

| Start | Interval | Global Successor | Local Successor |
|---|---|---|---|
| 7 | [7,0) | | |
| 0 | [0,2) | | |
| 2 | [2,6) | | |

*N7*

| Start | Interval | Global Successor | Local Successor |
|---|---|---|---|
| 0 | [0,1) | 0 | 1 |
| 1 | [1,3) | 1 | 4 |
| 3 | [3,7) | 3 | 4 |

Visualisons à présent chacun des trois *overlay*s du système pris individuellement. Commençons par le plan global pris individuellement. Sa représentation logique est alors donnée par la figure 6.3.

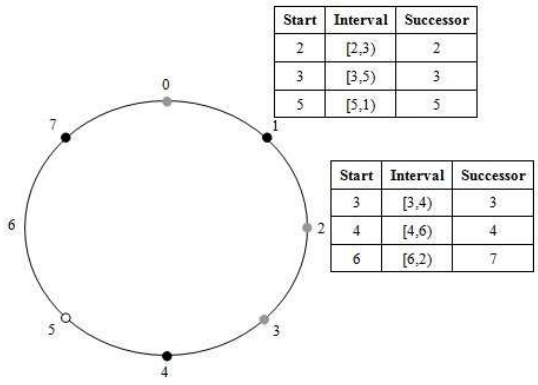


***Figure 6.3 :*** *Représentation logique du plan global pris individuellement*

Le tableau 6.2 donne les DHTs présentes au plan global.

***Tableau 6.2 :*** *Vue des DHTs du plan global*

*N0*

| Start | Interval | Successor | Local Successor |
|---|---|---|---|
| 1 | [1,2) | 1 | 2 |
| 2 | [2,4) | 2 | 2 |
| 4 | [4,0) | 4 | 0 |

*N1*

| Start | Interval | Successor | Local Successor |
|---|---|---|---|
| 2 | [2,3) | 2 | 4 |
| 3 | [3,5) | 3 | 4 |
| 5 | [5,1) | 5 | 7 |

*N2*

| Start | Interval | Successor | Local Successor |
|---|---|---|---|
| 3 | [3,4) | 3 | 3 |
| 4 | [4,6) | 4 | 0 |
| 6 | [6,2) | 7 | 0 |

*N3*

| Start | Interval | Successor | Local Successor |
|---|---|---|---|
| 4 | [4,5) | 4 | 0 |
| 5 | [5,7) | 5 | 0 |
| 7 | [7,3) | 7 | 0 |

*N4*

| Start | Interval | Successor | Local Successor |
|---|---|---|---|
| 5 | [5,6) | 5 | 7 |
| 6 | [6,0) | 7 | 7 |
| 0 | [0,4) | 0 | 1 |

*N5*

| Start | Interval | Successor | Local Successor |
|---|---|---|---|
| 6 | [6,7) | 7 | |
| 7 | [7,1) | 7 | |
| 1 | [1,5) | 1 | |

*N6*

| Start | Interval | Global Successor | Local Successor |
|---|---|---|---|
| 7 | [7,0) | | |
| 0 | [0,2) | | |
| 2 | [2,6) | | |

*N7*

| Start | Interval | Successor | Local Successor |
|---|---|---|---|
| 0 | [0,1) | 0 | 1 |
| 1 | [1,3) | 1 | 4 |
| 3 | [3,7) | 3 | 4 |

Regardons maintenant l'*overlay* relatif à P1, pris individuellement. Il est représenté par la figure 6.4

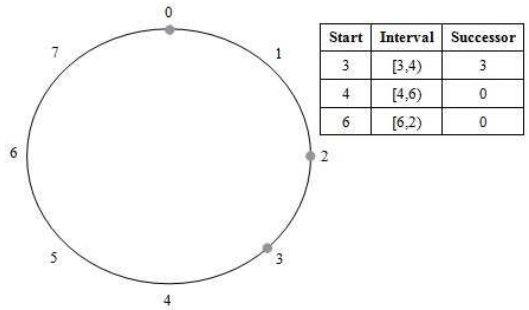


***Figure 6.4 :*** *Représentation logique de l'overlay local identifié par P1*

Les DHTs relatives à l'*overlay* correspondant à P1 sont représentées par le tableau 6.3

***Tableau 6.3 :*** *Vue des DHTs relatives à l'overlay local identifié par P1*

*N0*

| Start | Interval | Global Successor | Successor |
|---|---|---|---|
| 1 | [1,2) | 1 | 2 |
| 2 | [2,4) | 2 | 2 |
| 4 | [4,0) | 4 | 0 |

*N1*

| Start | Interval | Global Successor | Local Successor |
|---|---|---|---|
| 2 | [2,3) | 2 | 4 |
| 3 | [3,5) | 3 | 4 |
| 5 | [5,1) | 5 | 7 |

*N2*

| Start | Interval | Global Successor | Successor |
|---|---|---|---|
| 3 | [3,4) | 3 | 3 |
| 4 | [4,6) | 4 | 0 |
| 6 | [6,2) | 7 | 0 |

*N3*

| Start | Interval | Global Successor | Successor |
|---|---|---|---|
| 4 | [4,5) | 4 | 0 |
| 5 | [5,7) | 5 | 0 |
| 7 | [7,3) | 7 | 0 |

*N4*

| Start | Interval | Global Successor | Local Successor |
|---|---|---|---|
| 5 | [5,6) | 5 | 7 |
| 6 | [6,0) | 7 | 7 |
| 0 | [0,4) | 0 | 1 |

*N5*

| Start | Interval | Global Successor | Local Successor |
|---|---|---|---|
| 6 | [6,7) | 7 | |
| 7 | [7,1) | 7 | |
| 1 | [1,5) | 1 | |

*N6*

| Start | Interval | Global Successor | Local Successor |
|---|---|---|---|
| 7 | [7,0) | | |
| 0 | [0,2) | | |
| 2 | [2,6) | | |

*N7*

| Start | Interval | Global Successor | Local Successor |
|---|---|---|---|
| 0 | [0,1) | 0 | 1 |
| 1 | [1,3) | 1 | 4 |
| 3 | [3,7) | 3 | 4 |

L'*overlay* relatif à P2, pris individuellement, est quant à lui représenté par la figure 6.5

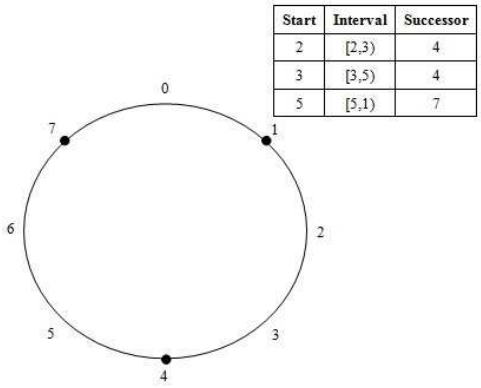


***Figure 6.5 :*** *Représentation logique de l'overlay local identifié par P2*

Le tableau 6.4 représente les DHTs relatives à l'*overlay* local identifié par P2.

***Tableau 6.4 :*** *Vue des DHTs relatives à l'overlay local identifié par P2*

*N0*

| Start | Interval | Global Successor | Local Successor |
|---|---|---|---|
| 1 | [1,2) | 1 | 2 |
| 2 | [2,4) | 2 | 3 |
| 4 | [4,0) | 4 | 0 |

*N1*

| Start | Interval | Global Successor | Successor |
|---|---|---|---|
| 2 | [2,3) | 2 | 4 |
| 3 | [3,5) | 3 | 4 |
| 5 | [5,1) | 5 | 7 |

*N2*

| Start | Interval | Global Successor | Local Successor |
|---|---|---|---|
| 3 | [3,4) | 3 | 3 |
| 4 | [4,6) | 4 | 0 |
| 6 | [6,2) | 7 | 0 |

*N3*

| Start | Interval | Global Successor | Local Successor |
|---|---|---|---|
| 4 | [4,5) | 4 | 0 |
| 5 | [5,7) | 5 | 0 |
| 7 | [7,3) | 7 | 0 |

*N4*

| Start | Interval | Global Successor | Successor |
|---|---|---|---|
| 5 | [5,6) | 5 | 7 |
| 6 | [6,0) | 7 | 7 |
| 0 | [0,4) | 0 | 1 |

*N5*

| Start | Interval | Global Successor | Local Successor |
|---|---|---|---|
| 6 | [6,7) | 7 | |
| 7 | [7,1) | 7 | |
| 1 | [1,5) | 1 | |

*N6*

| Start | Interval | Global Successor | Local Successor |
|---|---|---|---|
| 7 | [7,0) | | |
| 0 | [0,2) | | |
| 2 | [2,6) | | |

*N7*

| Start | Interval | Global Successor | Successor |
|---|---|---|---|
| 0 | [0,1) | 0 | 1 |
| 1 | [1,3) | 1 | 4 |
| 3 | [3,7) | 3 | 4 |

Signalons que nous supposons le système suffisamment actif pour que tous les objets soient disponibles dans chacun des trois *overlay*s considérés (P, P1 et P2). La disponibilité de l'objet au niveau local signifie que l'objet est mis en partage par l'un des nœuds représentés localement, éventuellement après une récupération du niveau global. C'est à partir de cet instant-là que nous nous positionnons pour les exemples qui suivent.

Nous avons donc que dans le cas où les pairs et objets gardent le même identifiant à tous les niveaux, alors si par exemple le nœud 2 veut récupérer l'objet 5, cela donnera d'après les tables ci-dessus :

- sans application de NETPOPPS, une recherche au niveau global uniquement et donc, un premier saut vers 4 et de là un deuxième saut vers 5 (cf. tableau 6.2 page 126) ;
- cependant avec NETPOPPS, la recherche commence au niveau local où 5 se trouve stocké sur 0, et où donc un seul saut est nécessaire pour aller de 2 à 0 (cf. tableau 6.3 page 127).

Considérons maintenant le cas où chaque pair est identifié par autant de *nodeId*s que d'*overlay*s où il est présent (c.-à-d. que la fonction injective utilisée pour la dérivation du *nodeId* primaire n'est pas idempotente). Dans

ce cas, les DHTs du système sont modifiées (les tableaux précédents ne sont plus valables) et l'on doit raisonner du point de vue des nœuds physiques.

Pour retrouver alors les DHTs du système, revenons à la répartition des pairs, comme considérée plus haut (cf. page 124), où on a supposé qu'à l'observation du système, on voit que :

- dans le plan global {0,1,2,3,4,5,7}, il n'y a pas de nœud d'identifiant 6 ;
- le nœud d'identifiant 5 est présent uniquement dans P ;
- l'ensemble des nœuds {0,2,3} du plan global se retrouve au niveau local dans P1 ;
- et l'ensemble des nœuds {1,4,7} du plan global se retrouve au niveau local dans P 2.

De cette répartition des nœuds logiques du système, et en supposant que les *nodeId*s de l'ensemble {0,1,2,3,4,5,7} sont des représentations logiques au plan global des nœuds physiques définis successivement par les éléments de l'ensemble {A,B,C,D,E,F,H}, on a que :

- l'ensemble des nœuds {A,C,D} se retrouve au plan P1 ;
- et l'ensemble des nœuds {B,E,H} se retrouvent au plan P2.

Supposons à présent que la fonction utilisée pour la dérivation des identifiants primaire est la fonction injective de permutation de 3 des éléments de l'ensemble : $h(x) : x \rightarrow (x+3)$ *mod* 8.

Aux éléments de l'ensemble {0,1,2,3,4,5,6,7}, $h(x)$ fait donc correspondre les éléments de l'ensemble {3,4,5,6,7,0,1,2} pris dans le même ordre. On a donc :

- F est représenté uniquement au plan global ; il l'est par le *nodeId* 5 ;
- {A,C,D} seront représentés dans le plan global par {0,2,3}, pris dans l'ordre, et dans P1 par {3,5,6}, pris dans le même ordre ;
- et {B,E,H} seront représentés dans le plan global par {1,4,7}, pris dans l'ordre, et dans P2 par {4,7,2}, pris dans le même ordre.

De tout cela découle la vue globale des DHTs au niveau du système, comme représentée par le tableau 6.5.

***Tableau 6.5 :*** *Vue globale des DHTs du système (avec une fonction injective non idempotente)*

*N0*

| Start | Interval | Global Successor | Local Successor |
|---|---|---|---|
| 1 | [1,2) | 1 | |
| 2 | [2,4) | 2 | |
| 4 | [4,0) | 4 | |

*N1*

| Start | Interval | Global Successor | Local Successor |
|---|---|---|---|
| 2 | [2,3) | 2 | |
| 3 | [3,5) | 3 | |
| 5 | [5,1) | 5 | |

*N2*

| Start | Interval | Global Successor | P2 Successor |
|---|---|---|---|
| 3 | [3,4) | 3 | 4 |
| 4 | [4,6) | 4 | 4 |
| 6 | [6,2) | 7 | 7 |

*N3*

| Start | Interval | Global Successor | P1 Successor |
|---|---|---|---|
| 4 | [4,5) | 4 | 5 |
| 5 | [5,7) | 5 | 5 |
| 7 | [7,3) | 7 | 3 |

*N4*

| Start | Interval | Global Successor | P2 Successor |
|---|---|---|---|
| 5 | [5,6) | 5 | 7 |
| 6 | [6,0) | 7 | 7 |
| 0 | [0,4) | 0 | 2 |

*N5*

| Start | Interval | Global Successor | P1 Successor |
|---|---|---|---|
| 6 | [6,7) | 7 | 6 |
| 7 | [7,1) | 7 | 3 |
| 1 | [1,5) | 1 | 3 |

*N6*

| Start | Interval | Global Successor | P1 Successor |
|---|---|---|---|
| 7 | [7,0) | | 3 |
| 0 | [0,2) | | 3 |
| 2 | [2,6) | | 3 |

*N7*

| Start | Interval | Global Successor | P2 Successor |
|---|---|---|---|
| 0 | [0,1) | 0 | 2 |
| 1 | [1,3) | 1 | 2 |
| 3 | [3,7) | 3 | 4 |

Pour mettre en relief les incidences positives de NETPOPPS sur l'efficacité du routage, selon la répartition des pairs (et le confinement des ressources) conformément à un modèle économique donné, nous distinguons, pour chacun des exemples qui vont suivre, trois cas différents :

- premièrement, le chemin de la requête dans le cas basique où le système fonctionne avec uniquement le réseau global (et donc avant la mise en œuvre de NETPOPPS) ;
- deuxièmement, le cas où seulement les objets gardent un même identifiant à chacun des niveaux où ils sont représentés (c'est le cas où une certaine fonction injective idempotente est appliquée pour la génération des *objectId*s uniquement) ;
- enfin, troisièmement, le cas où même les objets changent d'identifiant entre un niveau et l'autre.

Avec la fonction injective de permutation de 3, définie plus-haut, et grâce aux DHTs représentées par le tableau 6.5 (cf. page précédente), on a donc les exemples non exhaustifs suivants :

- si le pair physique C (donc 2, dans P) veut récupérer l'objet 5, on a :

- au niveau global, un premier saut a lieu vers 4 (donc pair physique D qui est dans l'entité de C suivant le modèle économique) et de là un deuxième saut vers 5 (donc pair physique E qui suivant le modèle économique n'est pas dans l'entité de D, ni de C) ;
- cependant au niveau local, C correspond à 5, et donc s'il veut récupérer l'objet 5, l'objet 5 est bien chez lui (=> plus aucun saut : on a gagné suivant le modèle économique et on a aussi gagné en nombre de sauts logiques) ;
- mais si l'objet change d'*objectId*, alors localement C (c.-à-d. 5) va chercher 0 ; alors il y aura un saut vers 3 (donc un seul saut, et en plus vers le pair physique A qui est dans l'entité de C suivant le modèle économique).

– si le pair physique C (donc 2, dans P) veut récupérer l'objet 6 :
- au niveau global, un saut a lieu vers 7 (donc pair physique H qui n'est pas dans l'entité de C suivant le modèle économique) ;
- cependant au niveau local, C correspond à 5, et donc s'il veut récupérer l'objet 6, il y aura un saut vers 6 (donc aussi un seul saut, mais vers le pair physique D qui est dans l'entité de C suivant le modèle économique) ;
- mais si l'objet change d'*objectId*, alors localement C (c.-à-d. 5) va chercher 1 ; alors il y aura un saut vers 3 (donc un seul saut, et en plus vers le pair physique A qui est dans l'entité de C suivant le modèle économique).

– si le pair physique C (donc 2, dans P) veut récupérer l'objet 0 :
- au niveau global, un saut a lieu vers 7 (donc pair physique H qui n'est pas dans l'entité de C suivant le modèle économique) et de là un deuxième saut vers 0 (donc pair physique A qui suivant le modèle économique n'est pas dans l'entité de H mais de C : donc un aller-retour vers double coût, pour revenir chez soi !) ;
- cependant au niveau local, C correspond à 5, et donc s'il veut récupérer l'objet 0, il y aura un saut vers 3 (donc un seul saut, et en

plus vers le pair physique A qui est dans l'entité de C suivant le modèle économique) ;

- mais si l'objet change d'*objectId*, alors localement C (c.-à-d. 5) va chercher 3 ; alors il y aura un saut vers 3 (donc un seul saut, et en plus vers le pair physique A qui est dans l'entité de C suivant le modèle économique).

Les exemples non exhaustifs présentés ci-haut montrent bien une amélioration du diamètre (nombre de sauts pour une requête) conforme au modèle 'économique' considéré. Ils nous montrent aussi que l'utilisation d'une fonction idempotente pour la dérivation de l'*objectId* primaire peut introduire encore des gains sur le chemin d'une requête (en tout cas pas de perte). Ces exemples nous montrent aussi que notre contribution respecte la localisation du chemin des requêtes conformément au confinement des ressources selon le modèle économique considéré.

## CONCLUSION DU CHAPITRE –

*Ce chapitre a établi une analyse comparative et applicative de nos deux contributions. Il a en plus expliqué un exemple illustré de leur incidence positive sur l'efficacité du routage P2P.*

# CONCLUSION

*« Foi et science*
*appellent l'une et l'autre*
*à la modestie et à la patience. »*

PHILIPPE DETERRE

## Récapitulation du fil conducteur du mémoire

Avec les discussions récentes et actuelles autour du piratage et des droits d'auteur, le P2P n'a pas toujours bonne presse. Une confusion commune en émerge, mêlant les applications et services fournis avec les différents mécanismes et infrastructures mis en jeu (ou pouvant l'être) dans ce mode de communication. Ainsi, et afin d'éviter toute confusion, nous avons rappelé la signification du terme P2P et ses différentes approches. Dans la suite, nous nous sommes positionnés au niveau infrastructure et mécanismes de routage.

Plusieurs modèles d'architecture P2P peuvent exister. Mais typiquement c'est le degré de décentralisation de l'index des ressources qui est retenu pour leur classification, puisque c'est là l'idée de base du P2P.

La première génération des applications P2P était à architecture centralisée. Une deuxième génération à architecture décentralisée non structurée a vite vu le jour, suivie par une troisième génération d'applications à architecture hybride. Parallèlement, à partir de la deuxième génération, des protocoles et des architectures décentralisés structurés ont été développés. Toutes ces architectures et leurs principaux protocoles de découverte de ressources dynamiques ont été présentés et analysés (au ***premier*** chapitre). La majorité d'entre eux ainsi que les applications P2P les plus populaires utilisent des tables de hachage distribuées (DHT). C'est pour cela que nous avons basé nos travaux de recherche sur les DHTs.

Les DHTs ont aussi été analysées dans ce mémoire. Ce sont des algorithmes qui fournissent des mécanismes efficaces pour la découverte des ressources. Ils

garantissent de trouver la ressource recherchée en un nombre minimal de sauts, si celle-ci existe dans le réseau. Cependant, les DHTs supposent que tous les pairs qui y sont référencés sont dans le même domaine de transport, et que les ressources disponibles sont uniformément réparties dans le réseau. Or, ces deux suppositions ne se vérifient pas dans la pratique.

Concernant la disponibilité des ressources, le P2P est communément assimilé à un échange de fichiers alors qu'il permet toute sorte d'applications et services distribués. Cependant, au-delà du partage, nous sommes confrontés à une puissance disponible (en termes de CPU, bande passante, stockage, densité, etc.) qui varie considérablement d'un pair à l'autre.

Quant à ce qui est de la localité des pairs et des chemins, il ne faut pas oublier que le P2P tire effectivement ses racines des infrastructures IP. Il se déploie donc bien au-dessus de celles-ci. Toutefois pour un même système, les topologies P2P et IP sont souvent bien différentes puisque le réseau *overlay* ne prend pas naturellement en compte les caractéristiques du réseau *underlay*. Le réseau *overlay* n'est donc pas sensible au réseau *underlay*. De plus, tous les pairs sont des sources potentielles de données, et les communications se développent de manière autonome et spontanée entre eux. Du coup, deux pairs voisins au niveau P2P ne le sont pas nécessairement au niveau IP. Et un saut logique (au sens P2P) engendre bien souvent plus d'un saut IP, en passant aussi par différents domaines de transport.

Par conséquent, la rapidité dans les recherches ou les transferts de contenu, même dans les réseaux P2P, va nécessairement dépendre des ressources qui y sont disponibles, ainsi que de la topologie du réseau IP sous-jacent. Nous avons pour cela établi les exigences pour un routage P2P efficace (au ***deuxième*** chapitre).

Du fait que les réseaux P2P recouvrent le réseau Internet, qui est totalement distribué et régi par des ISPs, il découle que les informations relatives à la topologie sous-jacente d'une DHT sont distribuées à l'échelle d'un ISP, d'un opérateur de réseaux, d'un AS ou d'un certain domaine administratif. Par conséquent, un système P2P sensible à la topologie sous-jacente doit, en sus du passage à l'échelle et la robustesse, satisfaire à la fois les usagers du P2P et les intérêts des ISPs et autres opérateurs de réseaux ou services.

Les usagers du P2P sont intéressés par des services de bonnes performances et qualités perçues, et par des temps de réponses et de récupération des ressources constamment améliorés. Quant aux ISPs et autres opérateurs, ils sont intéressés par l'évitement de congestion des liens critiques, et aspirent à l'optimisation de l'usage des ressources réseau (notamment la bande-passante) et à la réduction des différents coûts d'opération et de maintenance.

La sensibilité du réseau P2P au réseau sous-jacent nécessite l'usage effectif des informations de topologie et de proximité du réseau *underlay*. Ceci implique :

- premièrement, des techniques de découverte et génération de ces informations ;
- deuxièmement, des techniques d'exploitation de l'information fournie. L'exploitation doit être possible au niveau des requêtes et du processus de récupération des données, mais aussi au niveau de l'insertion de nouveaux pairs dans le réseau.

Les techniques existantes de génération de l'information de proximité ont été présentées au ***deuxième*** chapitre. Celles permettant l'exploitation d'une telle information ont été présentées au ***troisième*** chapitre. Ce chapitre s'est aussi poursuivi par une présentation et analyse des principales solutions de prise en compte de la topologie sous-jacente, ainsi que des solutions émergeantes permettant une coopération entre le flux P2P et un opérateur de réseaux.

Nos travaux de recherche sur la sensibilité à la structure du réseau sous-jacent d'un réseau P2P utilisant une DHT ont abouti à la définition, la conception et la spécification de deux systèmes différents :

- CAP (*Context-Aware P2P system*), un système P2P sensible au contexte, qui introduit une sémantique dans les identifiants des pairs et des objets. Il a été présenté et analysé au ***quatrième*** chapitre ;
- et NETPOPPS (*NETwork Provider Oriented P2P System*), un système P2P orienté opérateur de réseau, qui adapte les flux P2P à la topologie sous-jacente et aux politiques de l'opérateur, tout en simplifiant la gestion des identifiants des différents pairs et objets. Ce système a été présenté et analysé au ***cinquième*** chapitre.

Enfin, nos deux contributions ont été comparées et discutées analytiquement, au ***sixième*** chapitre, aussi bien au niveau structurel que du point de vue applicatif.

**Perspectives de recherche**

Les réseaux P2P à base de DHT ont été validés par diverses applications à grandes échelles. Ce type de routage applicatif se développera dans les années à venir. Il sera utilisé par des applications diversifiées. Des exigences nouvelles seront demandées à ces architectures comme : la sécurité, le cloisonnement, la qualité de service. Des constructions de DHTs à base de simples fonctions de hachages ne suffisent pas pour atteindre ces exigences. De nouveaux modèles de routage (peut être autres que les DHTs) seront nécessaires. En tout cas il sera nécessaire de prospecter dans différentes voies (peut-être une combinaison hybride avec les *skip graph*s [AS03, HJS+03]).

Les approches proposées interviennent dans chaque requête (c'est le cas du service intermédiaire appelé *oracle* [AFS07]), ou nécessitent des infrastructures dédiées (c'est le cas de P4P [XKS+07]) ne pouvant pas être déployées facilement puisqu'elles nécessitent des équipements propres.

Nous avons choisi de travailler sur les DHTs et de les augmenter par petit pas afin d'atteindre la recherche des ressources dans un domaine spécifique. Cette approche est dans la logique des DHTs et elle offre une sémantique à la recherche. Il manque à cette approche une validation à échelle suffisante. Cette validation est en cours sur la plateforme OverSim [Ove, BHK07].

Une voie qui nous paraît intéressante à prospecter est celle qui consiste à distinguer les identifiants des pairs et ceux des objets. C'est-à-dire un réseau *overlay* unique de l'ensemble des pairs, et plusieurs réseaux *overlay*s d'objets, chaque *overlay* possédant SA propre sémantique.

Une piste pourrait être la combinaison des deux techniques de base de nos contributions : appliquer la dérivation de clé à une fonction de hachage HMAC afin de gagner de la richesse de CAP et de la simplicité de gestion des identifiants de NETPOPPS.

La flexibilité de CAP et son caractère orienté usager nous amènent à réfléchir à de nouveaux services à offrir dans les réseaux P2P, mais aussi, à introduire les réseaux P2P dans de nouveaux domaines, tel l'identification par radio fréquence (ou *RFID* pour *Radio Frequency IDentification*).

# LISTE DES ABRÉVIATIONS

| | |
|---|---|
| ADSL | Asymmetrical Digital Subscriber Line |
| ARD | Autonomous Routing Domain |
| AS | Autonomous System |
| ASN | AS Number |
| BE | Basic Entity |
| BGP | Border Gateway Protocol |
| CAN | Content-Addressable Network |
| CAP | Context-Aware P2P system |
| CDN | Content Delivery Network |
| CFS | Collaborative File System |
| CIDR | Classless Inter-Domain Routing |
| CN | Control Node |
| CPU | Central Processing Unit |
| DHT | Distributed Hash Tables |
| DKS | Distributed K-ary Search |
| DNS | Domain Name Service |
| DoS | Denial of Service |
| FAI | Fournisseur d'Accès à Internet |
| GPS | Global Positioning System |
| GUID | Global Unique IDentifier |
| HMAC | Hashed Message Authentication Code |

| | |
|---|---|
| IE | Intermediate Entity |
| IETF | Internet Engineering Task Force |
| IP | Internet Protocol |
| ISP | Internet Service Provider |
| IT | Information Technology |
| LRU | Least Recently Used |
| MAC | Message Authentication Code |
| MD5 | Message Digest 5 |
| NETPOPPS | NETwork-Provider Oriented P2P System |
| NP | Network Provider |
| ODRI | Optimal Diameter Routing Infrastructure |
| P2P | Pair-à-Pair / Peer-to-Peer |
| P4P | Provider Portal for P2P |
| PNS | Proximity Neighbor Selection |
| RFC | Request For Comments |
| RFID | Radio Frequency IDentification |
| RON | Resilient Overlay Network |
| RP | Rendezvous Point |
| RT | Reference Table |
| RTT | Round Trip Time |
| SHA1 | Secure Hashing Algorithm 1 |
| SDSI | Simple Distributed Security Infrastructure |
| SIMD | Single Instruction Multiple Data |

| | |
|---|---|
| TTL | Time To Live |
| UDP | User Datagram Protocol |
| ZT | Zone Table |

# LISTE DES FIGURES

# LISTE DES TABLEAUX

# BIBLIOGRAPHIE


[ABK+01] Andersen D. G., Balakrishnan H., Kaashoek F., Morris R. (2001) *Resilient Overlay Networks*. In: 18th Symposium on Operating Systems Principles (SOSP), Oct 21-24, Banff, Canada. ACM Press, pp.131–145

[ACM+02] Ajmani S., Clarke D. E., Moh C., Richman S. (2002) *ConChord : Cooperative SDSI certificate storage and name resolution*. In: 1st International Workshop on Peer-to-Peer Systems (IPTPS), Mar 07-08, Cambridge, Massachusetts, États-Unis. Springer Verlag, number 2429 in LNCS, pp. 141–154

[AFS07] Aggrawal V., Feldmann A., Scheideler C. (2007) *Can ISPs and P2P Users Cooperate for Improved Performance*. In: ACM SIGCOMM Computer Communications Review 37(3):31–40

[Aka] Akamai Technologies : <http://www.akamai.com>

[And96] Anderson R. J. (1996) *The Eternity Service*. In: 1st International Conference on the Theory and Applications of Cryptology (Pragocrypt), Sep 30 – Oct 03, Prague, République Tchèque. Czech Technical University (CTU) Publishing House, pp. 242–252

[AS03] Aspnes J., Shah G. (2003) *Skip graphs*. In: 14th Symposium on Discrete Algorithms (SODA), Jan 12-14, Baltimore, Maryland, États-Unis. Society for Industrial and Applied Mathematics (SIAM), pp. 384–393

[AS04] Antonopoulos N., Salter J. (2004) *Improving Query Routing Efficiency in Peer-to-Peer Networks*. CS-04-01. University of Surrey, 10 p.

[BHK07] Baumgart I., Heep B., Krause S. (2007) *OverSim: A Flexible Overlay Network Simulation Framework,* In: 10th IEEE Global Internet Symposium (GI) in conjunction with IEEE INFOCOM, May 11, Anchorage, AK, USA.

[Bit] BitTorrent, Inc. : <http://www.bittorrent.com>

[BS06a] Baset S. and Schulzrinne H. (2006) *An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol*. In: 25th IEEE International Conference on Computer Communications (INFOCOM), Apr 23-29, Barcelone, Espagne. IEEE Communication Society, 11 p.

[BS06b] Balke W.-T., Siberski W. (2006) *Structured Peer-to-Peer Networks* [en ligne]. L3S Research Center, University of Hannover. Disponible sur : <http://www.l3s.de/~balke/lecture-p2p/Vorlesung_3.pdf> (consulté le 20.01.2009)

[CC03] Cavaillé G., Chalouhi O. (2003) *BitTorrent : le guide complet !* [en ligne]. Disponible sur : <http://ratiatum.com/dossier1356_BitTorrent_Le_guide_complet.html> (consulté le 20.01.2009)

[CCL+04] Castro R., Coates M.J., Liang G., Nowak R., Yu B. (2004) *Network Tomography: Recent Developments*. Statistical Science 19(3):499–517

[CDH+02] Castro M., Druschel P., Hu Y. C., Rowstron A. (2002) *Topology-Aware Routing in Structured Peer-to-Peer Overlay Networks*. MSR-TR-2002-82. Microsoft Research, Redmond, 19 p.

[CDK+02] Castro M., Druschel P., Kermarrec A.-M., Rowstron A. (2002) *SCRIBE: A large-scale and decentralized application-level multicast infrastructure*. In: IEEE Journal on Selected Areas in Communications (JSAC) (Special issue on Network Support for Multicast Communications), 20(8):1489–1499

[CDK+03] Cox R., Dabek F., Kaashoek F., Li J., Morris R. (2003) *Practical, distributed network coordinates*. In: Proceedings of HotNets-II. ACM Press, New York

[Cha05] Champeau G. (2005) *BitTorrent officiellement décentralisé* [en ligne]. Disponible sur : <http://www.ratiatum.com/journal.php?mode=pdf&id=2200> (consulté le 20.01.2009)

[Cis] Cisco Systems, Inc. : <http://www.cisco.com>

[Cis08] Cisco Systems, Inc. (Jun 2008) *Approaching the Zettabyte Era*, 23 p.

[Cli01] Clip2 Distributed Search Solutions. (2001) *The Gnutella Protocol Specification v0.4* (Document Revision 1.2), 10 p.

[CMM02] Cox R., Muthitacharoen A., Morris R. (2002) *Serving DNS using Chord*. In: 1st International Workshop on Peer-to-Peer Systems (IPTPS'02), Mar 07-08, Cambridge, Massachusetts, États-Unis. Springer Verlag, number 2429 in LNCS, pp. 155–165

[CMN02] Cox L. P., Murray C. D., Noble B. D. (2002) *Pastiche: making backup cheap and easy*. ACM Special Interest Group On Operating Systems (SIGOPS) Oper. Syst. Rev., 36(SI):285–298

[CSW+00] Clark I., Sandberg O., Willey B., Hong T.W. (2000) *Freenet: A distributed anonymous information storage and retrieval system*. In: Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability, July 25-26, Berkeley, California, États-Unis. International Computer Science Institute (ICSI), pp. 311–320

[Dal03] Dallons Q. (2003) *JXTA: Un Framework Peer-to-Peer Open Source*. Namur : Institut d'Informatique FUNDP, 18 p.

[deB46] de Bruijn N. (1979) *A combinatorial problem*. In: Koninklijke Nederlandse Academie van Wetenschappen, vol. 49, pp. 758–764

[DGA04] Datta A., Girdzijauskas S., Aberer K. (2004) *On de Bruijn routing in distributed hash tables: there and back again*. In: 4th International Conference on P2P Computing (P2P), Aug 25-27, Zürich, Suisse. IEEE Computer Society, pp. 159–166

[Dic08] Dickson F. (2008) *P2P Networking: Content's "Bad Boy" Becomes Tomorrow's Distribution Channel*. MultiMedia Intelligence, 43 p.

[DKK+01] Dabek F., Kaashoek M. F., Karger D., Morris R., Stoica I. (2001) *Wide-area cooperative storage with CFS*. In: 18th Symposium on Operating Systems Principles (SOSP), Oct 21-24, Banff, Canada. ACM Press, pp. 202–215

[DR01] Druschel P., Rowstron A. (2001) *PAST: A large-scale, persistent peer-to-peer storage utility*. In: 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII), May 20-23, Elmau, Allemagne. IEEE Computer Society, pp. 65–70

[EAB+02] El-Ansary S., Alima L. O., Brand P., Haridi S. (2002) *A Framework for Peer-To-Peer Lookup Services based on k-ary search*. SICS-T-2002/06-SE. Kista, Suède : Royal Institute of Technology, 13 p.

[eMu] eMule-Project. Site officiel d'eMule : <http://www.emule-project.net/>

[eMu04] eMule-Project.net. (2004) *eMule v.40f is available*. [en ligne]. Disponible sur : <http://www.emule-project.net/home/perl/news.cgi?l=1&cat_id=&b=20> (consulté le 20.01.2009)

[Evi08] Evidenzia. (2008) *P2P Monitoring Systems*. [en ligne]. Disponible sur : <http://www.evidenzia.de/eng_stats_all_releases_line.html> (consulté le 20.01.2009)

[Fer06] Ferguson D. (2006) *Trends and Statistics in Peer-to-Peer*. VP Engineering CacheLogic.

[FG06] Fraigniaud P., Gauron P. (2006) *D2B: a de Bruijn Based Content-Addressable Network*. Theoretical Computer Science 355(1):65–79

[FGJ04] Ferreira R. A., Grama A., Jagannathan S. (2004) *Plethora: An Efficient Wide-Area Storage System*. In: High Performance Computing (HiPC), Springer Verlag, number 3296 in LNCS

[Fre] FreePastry : <http://www.freepastry.org>

[GDJ06] Guha S., Daswani N., Jain R. (2006) *An Experimental Study of the Skype Peer-to-Peer VoIP System*. In: 5th International Workshop on Peer-to-Peer Systems (IPTPS), Feb 27-28, Santa Barabara, Californie, États-Unis. 6 p.

[GER+03] Garcés-Erice L., Ross K. W., Biersack E., Felber P. A., Urvoy-Keller G. (2003) *TOPLUS: Topology Centric Lookup Service*. In: Group Communications and Charges (NGC). Springer Verlag, number 2816 in LNCS

[GGG04] Ganesan P., Gummadi K., Garcia-Molina H. (2004) *Canon in G Major: Designing DHTs with Hierarchical Structure*. In : 24th International Conference on Distributed Computing Systems (ICDCS), Mar 23-26, Tokyo, Japon. IEEE Computer Society, 2004, pp. 263–272

[GGG+03] Gummadi P .K., Gummadi R., Gribble S., Ratnasamy S., Shenker S., Stoica I. (2003) *The Impact of DHT Routing Geometry on Resilience*

*and Proximity*. In: Proceedings of SIGCOMM. ACM Press, New York

[GV04] Gai A.-T., Viennot L. (2004) *Broose: A Practical Distributed Hashtable Based on the De-Brujin Topology*. RR-5238. France : Institut National de Recherche en Informatique et en Automatique (INRIA), 19 p.

[Har04] Hargreaves T. (2004) *The FastTrack Protocol* [en ligne]. Disponible sur : <http://cvs.berlios.de/cgi-bin/viewcvs.cgi/gift-fasttrack/giFT-FastTrack/PROTOCOL?rev=HEAD> (consulté le 20.01.2009)

[HBB+05] Hassan H. R., Bouabdallah A., Bettahar H., Challal Y. (2005) *An Efficient Key Management Algorithm for Hierarchical Group Communication*. In: First International Conference on Security and Privacy for Emerging Areas (SecureComm), Sep 05-09, Athens, Greece. IEEE Computer Society, pp. 270–276

[HJS+03] Harvey N. J. A., Jones M. B., Saroiu S., Theimer M., Wolman A. (2003) *SkipNet: A Scalable Overlay Network with Practical Locality Properties*. In: 4th USENIX Symposium on Internet Technologies and Systems (USITS), Mar 26-28, Seattle, Washington, États-Unis. USENIX Press, pp. 113–126

[JXTA] JXTA™ Community Projects : <https://jxta.dev.java.net/>

[JZD+02] Joseph A. D., Zhao B. Y., Duan Y., Huang L., Kubiatowicz J. D. (2002) *Brocade: Landmark Routing on Overlay Networks*. In: Peer-to-Peer Systems (IPTPS). Springer Verlag, number 2429 in LNCS

[KB05] Kulbak Y., Bickson D. (2005) *The eMule Protocol Specification*. HUJI-CSE-LTR-2005-3. Jérusalem : The Hebrew University of Jerusalem, 68 p.

[KBC+00] Kubiatowicz J., Bindel D., Chen Y., Czerwinski S., Eaton P., Geels D., Gummadi R., Rhea S., Weatherspoon H., Weimer W., Wells C., Zhao B. (2000) *Oceanstore: An architecture for global-scale persistent storage*. In: 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Nov 12-15, Cambridge, Massachusetts, États-Unis. ACM Press

[KK03] Kaashoek M. F., Karger D. R. (2003) *Koorde: A simple degree-optimal distributed hash table*. In: 2nd International Workshop on Peer-to-Peer Systems (IPTPS), Feb 20-21, Berkeley, California, États-Unis. Springer Verlag, number 2735 in LNCS, 2003, pp. 98–107

[KLL+97] Karger D., Lehman E., Leighton F., Levine M., Lewin D., Panigrahy R. (1997) *Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web*. In: Proceedings of STOC. ACM Press

[KMX+03] Kumar A., Merugu S., Xu J., Yu X. (2003) *Ulysses: A Robust, Low-Diameter Low-Latency Peer-to-Peer Network*. In: 11th International Conference on Network Protocols (ICNP), Nov 04-07, Atlanta, Georgia, États-Unis. IEEE Communications Society, pp. 258–267

[KRP05] Karagiannis T., Rodriguez P., Papagiannaki K. (2005) *Should ISPs fear Peer-Assisted Content Distribution?* In: ACM SIGCOMM Proceedings of IMC. USENIX Association

[KTC+04] Keralapura R., Taft N., Chuah C.N., Iannaccone G. (2004) *Can ISPs Take the Heat from Overlay Networks?* In: Proceedings of HotNets-III. ACM Press, New York

[Loe08] Loewenstern A. (2008) *DHT Protocol.* BitTorrent.org. [en ligne]. Disponible sur : <http://www.bittorrent.org/beps/bep_0005.html> (consulté le 20.01.2009)

[LKR+03] Loguinov D., Kumar A., Rai V., Ganesh S. (2003) *Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience*. In: Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM), Aug 25-29, Karlsruhe, Allemagne. ACM Press, pp. 395–406

[LLX+04] Liu Y., Liu X., Xiao L., Ni L. M., Zhang X. (2004) *Location-Aware Topology Matching in P2P Systems*. In: 23th Conference of the IEEE Communications Society (INFOCOM), Mar 07-11, Hong-Kong. IEEE Computer Society.

[LZG+06] Liu Y., Zhang H., Gong W., Towsley D. (2005) *On the interaction between overlay routing and traffic engineering*. In: Proceedings of INFOCOM. IEEE Communications Society

[Meu07] Meulle M. (2007) *Interference of AS business relationships and routing policies in the Internet*. PhD Thesis. Université Blaise Pascal Clermont-Ferrand, France

[MG08] Marocco E., Gurbani V. (2008) *Application-Layer Traffic Optimization (ALTO) Problem Statement*. draft-marocco-alto-problem-statement-01 (work in progress)

[MIP+06] Madhyastha H.V., Isdal T., Piatek M., Dixon C., Anderson T., Krishnamurthy A., Venkataramani A. (2006) *iPlane: an information plane for distributed services*. In: Proceedings of OSDI. USENIX Association

[MJB05] Montresor A., Jelasity M., Babaoglu O. (2005) *Chord on Demand*. In: 5th International Conference on P2P Computing (P2P), Aug 31 – Sep 02, Konstanz, Allemagne. IEEE Computer Society, pp. 87–94

[MKL+02] Milojičić D. S., Kalogeraki V., Lukose R., Nagaraja K., Pruyne J., Richard B., Rollins S., Xu Z. (2002) *Peer-to-Peer Computing*. HPL-2002-57. Hewlett-Packard Laboratories, 52 p.

[MM02] Maymounkov P., Mazières D. (2002) *Kademlia: A peer-to-peer information system based on the XOR metric*. In: 1st International Workshop on Peer-to-Peer Systems (IPTPS), Mar 07-08, Cambridge, Massachusetts, États-Unis. Springer Verlag, number 2429 in LNCS, 2002, pp.53–65

[MNR02] Malkhi D., Naor M., Ratajczak D. (2002) *Viceroy: A Scalable and Dynamic Emulation of the Butterfly*. In: Proceedings of PODC. ACM Press, New York

[Mor03] Morlon J. (2003) *CDN pour Content Delivery Network* [en ligne]. Le Journal du Net, 2003. Disponible sur : <http://www.journaldunet.com/solutions/0302/030211_faq_cdn.shtml> (consulté le 20.01.2009)

[Nap99] NAPSTER : <http://www.napster.com>

[Oce] OceanStore : <http://www.oceanstore.org/>

[Ove] The OverSim P2P Simulator : <http://www.oversim.org/>

[PAC+02] Peterson L., Anderson T., Culler R., Roscoe T. (2002) *A Blueprint for Introducing Disruptive Technology into the Internet*. In: 1st Workshop on Hot Topics in Networks (HotNets-I), Oct 28-29, Princeton, États-Unis. ACM Press.

[Par04] Parker A. (2004) *The True Picture of Peer-to-Peer Filesharing*. CacheLogic Ltd.

[Pla] PlanetLab : <http://www.planet-lab.org>

[PRR97] Plaxton C., Rajaram R., Richa A. (1997) *Accessing Nearby Copies of Replicated Objects in a Distributed Environment*. In: 9th ACM Symposium on Parallel Algorithms and Architectures (SPAA), Jun 23-25, Newport, Rhode Island, États-Unis. ACM Press, pp. 311–320

[Pug90] Pugh W. (1990) *Skip Lists: A Probabilistic Alternative to Balanced Trees*. Communications of the ACM, 33(6):668–676

[Rat02] Ratnasamy S. (2002) *A Scalable Content-Addressable Network*. Ph.D. Thesis, University of California at Berkeley

[RD01] Rowstron A., Druschel P. (2001) *Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems*. In: IFIP/ACM Middleware. Springer Verlag, number 2218 in LNCS

[RFC4632] Fuller V. and Li T. (2006) *Classless Inter-Domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan*. IETF. RFC 4632, 27 p.

[RFC1771] Rekhter Y. and Li T. (1995) *A Border Gateway Protocol 4 (BGP-4).* IETF. RFC 1771, 57 p.

[RFC2104] Krawczyk H., Bellare M., and Canetti R. (1997) *HMAC: keyed-hashing for message authentication*. IETF. RFC 2104, 11 p.

[RFH+01] Ratnasamy S. P., Francis P., Handley M., Karp R., Shenker S. (2001) *A scalable content-addressable network.* In: Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM), Aug, San Diego, CA, US, pp. 161–172

[RH07] Rostami H., Habibi J. (2007) *Topology awareness of overlay P2P networks*. Concurrency and Computation: Practice & Experience 19(7):999–1021

[Rhe05] Rhea S. (2005) *OpenDHT: A Public DHT Service*. Ph.D. Thesis, University of California, Berkeley

[RHK+02] Ratnasamy S., Handley M., Karp R, Shenker S. (2002) *Topologically aware overlay construction and server selection*. In: Proceedings of INFOCOM. IEEE Communications Society

[Rip01] Ripeanu M. (2001) *Peer-to-Peer Architecture Case Study: Gnutella Network*. In: 1st International Conference on Peer-to-Peer Computing (P2P), Aug 27-29, Linköpings, Suède. IEEE Computer Society, pp. 99–100

[RKC+01] Rowstron A., Kermarrec A.-M., Castro M., Druschel P. (2001) *SCRIBE: The design of a large-scale event notification infrastructure*. In: 3rd International Workshop on Networked Group Communications (NGC), Nov 07-09, Londres, Royaume-Uni. UCL, pp. 30–43

[RM06] Risson J., Moors, T. (2006) *Survey of research towards robust peer-to-peer networks: Search methods*. Computer Networks, 50(17):3485–3521

[RSS02] Ratnasamy S., Shenker S., Stoica I. (2002) *Routing Algorithms for DHTs: Some Open Questions*. In: Peer-to-Peer Systems (IPTPS). Springer Verlag, number 2429 in LNCS

[SA06] Seetharaman S., Ammar M. (2006) *On the interaction between dynamic routing in the overlay and native layers*. In: Proceedings of INFOCOM. IEEE Communications Society

[San08a] Sandvine Incorporated ULC. (Jun 2008) *2008 Analysis of Traffic Demographics in North-American Broadband Networks*, 7p.

[San08b] Sandvine Incorporated ULC. (Oct 2008) *2008 Global Broadband Phenomena*, 8p.

[SF03] Schoder D., Fischbach K. (2003) *Peer-to-Peer-Netzwerke für das Ressourcenmanagement*. In: Wirtschaftsinformatik, 45(3):313–323

[Sie79] Siegel H. J. (1979) *Interconnection Networks for SIMD Machines.* In: IEEE Computer, 12(6):57–65

[Sky] Skype Limited : <http://www.skype.com>

[SM02] Sit E., Morris R.T. (2002) *Security Considerations for Peer-to-Peer Distributed Hash Tables*. In: Peer-to-Peer Systems (IPTPS). Springer Verlag, number 2429 in LNCS

[SM07] Schulze H., Mochalski K. (2007) *Internet Study 2007*. Allemagne : ipoque GmbH, 25 p.

[SML+01] Stoica I., Morris R., Liben-Nowell D., Karger D. R., Kaashoek M. F., Dabek F., Balakrishnan H. (2001) *Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications*. In: Proceedings of SIGCOMM. ACM Press, pp. 149–160

[SR04] Stutzbach D., Rejaie R. (2004) *Characterizing Today's Gnutella Topology.* CIS-TR-04-02. Oregon : Department of Computer Science, University of Oregon, 12 p.

[SR06] Stutzbach D., Rejaie R. (2006) *Improving Lookup Performance over a Widely-Deployed DHT*. In: 25th IEEE International Conference on Computer Communications (INFOCOM), Apr 23-29, Barcelone, Espagne. IEEE Communication Society, 12 p.

[Sto04] Stoica I. (2004) *Internet Indirection Infrastructure*. In: 1st Dagstuhl Seminar, Mar 07-10, Dagstuhl, Allemagne

[Vie05] Viennot L. (2005) *Pair-A-Pair : Routage dans les réseaux de pair à pair*. In: Panorama des Recherches Incitatives en STIC (PaRISTIC), Nov 21-23, Bordeaux, France

[WGR05] Wehrle K., Götz S., Rieche S. (2005) *Distributed Hash Tables*. In: Steinmetz R. and Wehrle K. (Eds.). P2P Systems and Applications. Springer Verlag, number 3485 in LNCS, p. 84

[XKS+07] Xie H., Krishnamurthy A., Silberschatz A., Yang Y. R. (2007) *P4P: Explicit Communications for Cooperative Control Between P2P and Network Providers*. P4PWG Whitepaper

[XMH03] Xu Z., Min R., Hu Y. (2003) *Hieras: A DHT Based Hierarchical P2P Routing Algorithm*. In: Proceedings of ICPP. IEEE Computer Society

[XMK03] Xu Z., Mahalingam M., Karlsson M. (2003) *Turning Heterogeneity into an Advantage in Overlay Routing*. In: Proceedings of INFOCOM. IEEE Communications Society

[XTZ02] Xu Z., Tang C., Zhang Z. (2002) *Building Topology-Aware Overlays using Global Soft-State.* HPL-2002-281. Hewlett-Packard Laboratories, 11 p.

[YGM03] Yang B., Garcia-Molina H. (2003) *Designing a Super-Peer Network*. In: 21st International Council for Open and Distance Education (ICDE), Mar 05-08, Bangalore, Inde. IEEE Computer Society, pp. 49–60

[ZHS+04] Zhao B. Y., Huang L., Stribling J., Rhea S. C., Joseph A. D., Kubiatowicz J. D. (2004) *Tapestry: A Resilient Global-Scale Overlay for Service Deployment*. IEEE J-SAC 22(1):41–53

[ZKJ01] Zhao B. Y., Kubiatowicz J. D., Joseph A. D. (2001) *Tapestry: An infrastructure for fault-tolerant wide-area location and routing*. In Tech. Report No. UCB/CSD-01-1141, Computer Science Division (EECS), University of California at Berkeley, California 94720, USA.

[ZZZ+03] Zhou F., Zhuang L., Zhao B. Y., Huang L., Joseph A. D., Kubiatowicz J. (2003) *Approximate Object Location and Spam Filtering on Peer-to-peer Systems*. In: 4th International Middleware Conference (Middleware), Jun 16-20, Rio de Janeiro, Brésil. ACM Press, pp. 01–20

# PUBLICATIONS

**Publications avec comité de lecture :**

Fayçal M., Mathieu B. (2007) *Topology-aware Peer-to-Peer Routing – Analysis and Perspectives*. In: First IEEE International Global Information Infrastructure Symposium (GIIS), Jul 02-06, Marrakech, Maroc

Fayçal M., Serhrouchni A. (2007) *CAP: A Context Aware Peer-to-Peer System*. In: On the Move to Meaningful Internet Systems (OTM) Workshop on Peer to Peer Networks (PPN), Nov 25-30, Vilamoura, Portugal

Fayçal M., Serhrouchni A. (2008) *An Efficient Management Technique for Peer-to-Peer Networks*. In: 16th IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Sep 25-27, Split – Dubrovnik, Croatie

Fayçal M., Serhrouchni A. (2008) *NETPOPPS: A Network Provider Oriented Peer-to-Peer System*. In: 2nd IFIP International Conference on New Technologies, Mobility and Security (NTMS), Nov 05-07, Tanger, Maroc

Fayçal M., Serhrouchni A. (2009) *Network-Aware DHT-based P2P Systems*. In: Handbook of Peer-to-Peer Networking. X. Shen, H. Yu, J. Buford, M. Akon (Eds.), Springer Verlag

**Rapport technique :**

Fayçal M. (2006) "*État de l'art des réseaux Pair-à-Pair*", France Télécom R&D, août, 45 p.

**Présentations invitées :**

Fayçal M., Mathieu B., Serhrouchni A. (2005) *Peer-to-Peer : découverte de ressources, évolution des algorithmes et limitations*. In: De nouvelles Architectures pour les Communications (DNAC), Apr 25-27, Beyrouth, Liban

Fayçal M., Mathieu B. (2006) *Vers un protocole P2P adapté réseau*, article de vulgarisation technique, France Télécom, revue R&D n°24, traduit pour la version anglaise de la même revue : *Towards a P2P protocol that works with the network*

Fayçal M. (2006) *Les réseaux P2P – état de l'art, analyse et perspectives*. Tutoriel in Colloque sur le P2P, May 16, ITIN, Cergy, France

Fayçal M. (2006) *Routage P2P adapté au réseau IP*. In: séminaire sur le P2P, Nov 21, France Télécom R&D, Cesson-Sévigné, France

Fayçal M., Serhrouchni A. (2008) *Vers de nouvelles architectures de type P4P*. In: Forum ATENA (à la convergence des NTIC et de l'enseignement supérieur) sur le Peer to Peer, Oct 30, Telecom ParisTech, Paris, France