N° d'ordre : 4600

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

Par **Cyril Cassagnes**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

Architecture Autonome et Distribuée d'Adressage et de Routage pour la Flexibilité des Communications dans l'Internet

---

Soutenue le : 12 novembre 2012

Après avis des rapporteurs :

Pascal Lorenz	Professeur
Philippe Roose	Maître de conférences - HDR

Devant la commission d'examen composée de :

Serge Chaumette	Professeur	Président du jury
Emmanuelle Anceaume	CR CNRS	Examinatrice
Philippe Owezarski	CR CNRS - HDR	Examinateur
Damien Magoni	Professeur	Directeur de thèse

– 2012 –

Remerciements

C'est avec une immense joie que j'écris ces lignes. Ahhh ! Les remerciements.

Tout d'abord, merci à Philippe Roose et à Pascal Lorenz d'avoir accepté la lourde tâche d'être mes rapporteurs. Merci aussi à Philippe Owezarski et à Emmanuelle Anceaume d'avoir accepté de faire partie de mon jury. Et merci à Serge Chaumette d'avoir accepté de le présider. Encore merci à Serge qui me supporte depuis le Master de m'ouvrir de nouvelles portes en cette fin de thèse.

Ensuite, mes remerciements vont à mon directeur de thèse Damien Magoni sans qui je n'en serai pas là. Cette thèse est l'aboutissement de nos nombreux face-à-face au Haut-Carré où, dans le calme comme dans la tempête, nous décidions du cap. Il est clairement lié à cette réussite. Je le remercie pour ces trois années ainsi que pour son choix il y a trois ans.

Durant ces trois années j'ai travaillé dans un environnement unique au monde. Je ne peux pas continuer ces remerciements sans dire quelques mots sur mon environnement de travail et mes collègues de CVT. Les qualificatifs me manquent... Que dire de ces membres, Telesphore *subscriber* présent quelques mois par an puis Mikey, Bibi, Dub, Joe, Jérémie, Hugox et Kaze qui au quotidien travaillent dans cette enclave du LaBRI. Ce lieu qui est parfois un havre, parfois un enfer catalyse malgré tout la bonne humeur.

Je remercie aussi toutes les personnes que j'ai pu croiser ici, et là, dans le cadre de mes déplacements. Ces échanges m'ont permis d'évoluer et de progresser durant ces trois années.

J'ai aussi à cœur de remercier les personnes qui m'ont guidé directement ou indirectement jusqu'à la thèse car les paroles s'envolent mais les écrits restent. Donc merci Arnaud de m'avoir mis le pied à l'étrier (ou sur le cale-pied) en me permettant de rejoindre une équipe de l'institut universitaire de technologie d'Anglet. Bien sûr je remercie Philippe et Marc pour m'avoir fait confiance. Mais honnêtement, c'est à l'ensemble du personnel de l'IUT d'Anglet que doivent aller mes remerciements pour leur gentillesse et leur sympathie. De Yon à Pierre en passant par Jean-Michel, merci ! Nombreuses sont les personnes qui, de près ou de loin, ont influencé mon parcours mais pour la thèse mon passage à Anglet y est pour beaucoup.

Enfin, je ne peux clore ces remerciements sans remercier ma famille et mes amis qui n'ouvriront certainement jamais la page de couverture de cette thèse. Merci et bonne lecture !

Table des matières

Table des figures

Introduction

Depuis les années 90, le réseau des réseaux, communément appelé Internet (de l'anglais *Inter-networking*), s'étend sur toute la planète. L'accession à Internet est devenue ordinaire et les débits réseaux toujours plus élevés permettent à tous de devenir des acteurs du réseau des réseaux. En effet, chacun peut décider d'apporter sa contribution aux contenus et services déjà présents dans l'Internet. Nous touchons là, peut être la première raison de son succès : la liberté. Par liberté, nous entendons liberté de créer et d'accéder. Toutefois, rappelons que cette liberté est née grâce aux scientifiques qui ont fourni les outils donnant à tout un chacun la possibilité d'interagir et de créer (e.g. l'*hypertext*, les navigateurs, le courrier électronique, etc.). Ainsi n'importe quel utilisateur peut, s'il le souhaite, créer un site ou une application Web [1] accessible au plus grand nombre. Pour reprendre les mots de Vinton Cerf [2] : l'Internet a créé une plate-forme pour l'innovation.

De cette innovation est né un nouveau type d'application où l'utilisateur prend part à un réseau virtuel dit pair-à-pair ou poste-à-poste (de l'anglais *peer-to-peer* (P2P)). Le modèle pair-à-pair est à la base des applications de diffusion / partage de contenus. C'est un paradigme puissant pour mettre à disposition des ressources dans un réseau. Puissant car les services implantés dans ce modèle sont destinés à être décentralisés entre les participants. Ainsi, chaque participant contribue et profite des contenus ou des services disponibles. Les systèmes ou applications P2P permettent un partage de l'information allant directement du producteur au consommateur, c'est-à-dire sans serveur central intermédiaire. Indéniablement, l'absence d'entité centrale rend le réseau robuste aux pannes ou aux attaques (légitime ou non) de tout niveau (physique et logiciel). Cependant les applications P2P ne sont pas nativement robuste aux attaques logicielles mettant en jeu des relations de confiance entre les participants. Malgré tout, cela fait de ce modèle un candidat idéal pour fournir des services ou des contenus pérennes, par exemple :

- Téléphonie (Skype, etc.)
- Flux vidéo (Zattoo, etc.)
- Partage de fichiers (Napster, GNUnet, etc.)
- Calcul distribué (Boinc, etc.)

L'Internet actuel se compose de l'interconnexion de dizaines de milliers de réseaux dits autonomes. L'architecture logicielle qui implante les différents protocoles de communication est basée sur un modèle en couches où seules les couches attenantes échangent des informations. Dans cette architecture c'est la couche dite de transport qui offre le

1. Le Web est un abus de langage provenant du terme anglais *World Wide Web* en abrégé WWW faisant référence à l'organisation des liens *hypertext* telle une toile d'araignée.
2. http ://googleblog.blogspot.fr/2005/11/vint-cerf-speaks-out-on-net-neutrality.html

modèle de bout en bout (de l'anglais *end-to-end model*) si cher aux utilisateurs de la première heure. Ce modèle est destiné à rendre une machine accessible et visible de tous au travers des autres machines du réseau. C'est un des ingrédients qui a favorisé le succès des applications P2P. Néanmoins, l'engouement déclenché par ces applications a engendré de grandes quantités de trafic difficile à évaluer précisément du fait de la variabilité de certains paramètres (e.g. numéro de port). Ce trafic semble s'être stabilisé depuis moins d'une dizaine d'années [KBB+04] et plus récemment dans [Cis12], CISCO donne une estimation du trafic généré pour 2011 et des prévisions jusqu'à l'horizon 2016. Donc sans nul doute depuis Napster les applications P2P se sont bien ancrées dans l'Internet. Toutefois l'énorme quantité d'informations qui transitent par ses liens n'est pas sans conséquence. Cela introduit des soucis considérables en raison du déséquilibre observé dans les flux de trafic. Cela a pour effet de compliquer la gestion du réseau et de réduire les performances utilisateurs. Pour les plus pessimistes cela réduit aussi la robustesse du réseau dans le sens où des éléments deviennent plus indispensable que d'autres. En effet, les utilisateurs d'applications P2P, et d'autres, font transiter de nombreuses données plus ou moins volumineuses. De plus, ces applications génèrent un trafic supplémentaire interne afin de se coordonner. Sans compter les problèmes techniques qui viennent s'additionner à cela tel que le "dernier kilomètre" ou l'asymétrie du réseau qui rendent complexe la création d'application performante. La qualité de la performance s'évalue sur la quantité de trafic interne nécessaire, la rapidité et la garantie de délivrance des messages. À la source de ce déséquilibre des flux de trafic est le choix de l'acheminement de chaque information peu corréler avec les besoins techniques (i.e. Peering).

Depuis la création d'Internet des modifications de ses protocoles sont proposées et quelques fois déployées pour soutenir le nombre toujours plus important d'applications de transfert de contenu point-à-point, du courrier électronique au P2P en passant par les clients-serveurs. De fait, Internet, né de l'interconnexion de nombreux réseaux est aujourd'hui au cœur de nombreux enjeux (sociétaux, économique, sécuritaire pour les données, énergético-écologique). Or l'architecture actuelle ne permet pas un modèle ou les utilisateurs peuvent être à la fois producteur et consommateur de contenu au travers d'un réseau de distribution (CDN *Content Distribution Network*) ou même d'un CCN (*Content Centric Networking*) [JST+09]. C'est pourquoi, l'architecture d'Internet est actuellement dans une phase importante car le réseau des réseaux doit évoluer afin de pouvoir satisfaire une demande grandissante en ressource. En outre, cette nouvelle architecture ne doit pas être un frein à la créativité et donc à l'innovation. Pour faire face, deux approches s'opposent : faire table rase [TBD+11] ou non [BFCW09]. À la base de cette remise en cause sont les limites de la couche transport actuelle et de son service d'acheminement de l'information. Dans ce mémoire, nous n'avons pas l'ambition de décrire une nouvelle architecture pour Internet mais présentons et justifions notre solution au problème clef de l'acheminement de l'information, commun à l'ensemble des réseaux de communication. En effet, l'évolution suivi par les périphériques utilisateurs en bordure n'arrange rien puisque l'accès au réseau ne se cantonne plus seulement aux ordinateurs personnels. Les appareils communicants sont désormais bien implantés dans notre environnement quotidien. Ces objets sont omniprésents, pour ne pas dire pervasif, et offrent la possibilité d'interagir en continu dans un monde virtuel. De plus, les capacités matérielles de ces objets progressent sans cesse. À ce rythme nous pouvons prédire des embouteillages sur l'autoroute de l'informa-

tion qui devra probablement faire face à des yotta octets d'informations car nous sommes tous devenus de potentiels producteurs de contenu, plutôt que de simples consommateurs, même si *free-riders* et *seeders* continueront probablement de catégoriser les utilisateurs. À cela s'ajoute le perfectionnement des technologies, les objets aussi deviennent des utilisateurs lorsqu'ils mettent en œuvre des communications machine-à-machine (M2M) afin d'interagir. Les technologies qui donnent à ces choses la capacité de "penser" (*Things that think* en anglais) sont à la base d'un concept lié à l'informatique ubiquitaire [Wei99] : l'Internet des Objets (IoT). L'entrée massive de ces objets dans l'Internet crée de nouvelles possibilités car Internet permet à ces appareils d'être constamment interconnectés (techniquement et socialement). Toutefois, indépendamment des applications (e.g. P2P) ou du type de réseau de communication sous-jacent (e.g. UMTS, 802.15.4, 802.11, Ethernet, etc.) la fonctionnalité de base requise est l'acheminement de l'information avec un algorithme pouvant gérer des millions d'objets. Or le passage à l'échelle du routage n'est actuellement pas assuré. Cette fonctionnalité, de base, est d'autant plus complexe à mettre en œuvre dans un environnement où la mobilité s'accroît. En effet, la volatilité des participants au réseau physique (*ad hoc*) et / ou logiciel (P2P) est un phénomène récent qui prend de l'ampleur et complique le problème du routage puisque cela rend le réseau instable et donc génère plus de trafic interne (trafic de contrôle). Pourtant, des solutions sont proposées pour inciter les participants à rester dans le réseau de manière prévisible notamment avec la théorie des jeux (*tit for tat* multi-joueurs ou heuristique de prédiction).

Toutes ces raisons nous poussent à aborder le problème de l'acheminement de l'information. Cela fait une dizaine d'années que la communauté des chercheurs en théorie des réseaux accorde un regain d'intérêt à ce problème de longue date qu'est le routage. Car, avec l'avènement d'Internet au rang de réseau de communication mondial, le nombre d'utilisateurs accédant ou contribuant à des services via ce réseau n'a cessé de croître. Cela est d'abord dû à la démocratisation des ordinateurs personnels puis à celle des appareils communicants qui font toujours plus partie de notre environnement quotidien et viennent s'agréger à Internet.

Dans le but de tester et de déployer notre solution, nous étudions les réseaux virtuels. Un réseau virtuel est un graphe abstrait construit sur un graphe sous-jacent. Le degré peut être borné et la topologie peut être dynamique du fait de la volatilité. En général, le degré dépend de la configuration et des capacités physiques des terminaux utilisateurs. De plus, le modèle d'interconnexion idéal pour ces réseaux virtuels est le P2P. L'analyse de tels systèmes est complexe, c'est pourquoi nous nous y intéressons. Afin de tester notre solution nous implantons notre système P2P dans un simulateur offrant les modèles statistiques et probabilistes habituels tel que la loi exponentielle. Le réseau virtuel est destiné à prendre place au-dessus de la couche des protocoles de transport (TCP ou UDP) du modèle OSI (de l'anglais *Open Systems Interconnection*). Ainsi notre système tient compte de l'architecture de l'Internet actuelle. Nous proposons un algorithme distribué capable d'étiqueter n'importe quel type d'entité et de router entre ces entités à l'aide de ces étiquettes. Nous évaluons cet algorithme pour un réseau statique et un réseau dynamique. Le routage s'effectue grâce à une table des voisins locale à chaque pair et dont la taille est variable. Les problèmes sont :

- L'étirement des itinéraires de routage vis-à-vis du routage utilisant l'algorithme de Dijkstra ;

- L'utilisation d'un routage indéfectible qui passe à l'échelle et où le message est dirigé vers le voisin qui rapproche le plus de la destination ;
- L'influence de l'interconnexion des pairs afin de créer une structure particulière (e.g. petit monde).

Néanmoins, ces systèmes sont plus complexes à concevoir que les systèmes respectant le modèle client-serveur. La construction d'un système P2P nécessite de :

- Définir un ou plusieurs protocoles de nommage pour l'ajout et le retrait des pairs dans le système.
- Choisir si la topologie du réseau est structurée ou non et comment les données sont acheminées.
- Définir un mode de gestion des données (indexation, accession) telle qu'une table de hachage distribuée.
- Définir un protocole de transfert de fichiers efficace sur la couche de transport (TCP ou UDP) d'Internet.

Ces étapes dans la construction du système P2P ne sont pas indépendantes car certains choix restreignent le nombre de solutions mais l'absence de serveur permet de répartir la charge entre les participants en termes de : trafic de contrôle, sollicitation à gérer (i.e. nombre de requêtes), stockage des données.

Dans ce mémoire, nous étudions au travers d'un réseau virtuel P2P, le potentiel de notre service d'adressage et de routage pour les membres d'un tel réseau dans un environnement dynamique. La finalité de ce service est de permettre la mise en œuvre d'une table de hachage distribuée destinée à gérer des sessions utilisateur et du contenu. En effet, notre architecture finale vise à supporter la gestion de sessions virtuelles afin de favoriser la flexibilité des communications dans l'Internet. L'application P2P sera capable de lier de manière souple, les utilisateurs, les applications et les périphériques, lors d'une communication grâce à la gestion / restauration des sessions.

Structure du document

Le document est structuré de la manière suivante. Nous exprimons les problèmes avec le niveau d'abstraction le plus élevé possible afin de comprendre au mieux leur portée, donc leur importance. Ensuite, nous raffinons nos explications au fil du document, pour se concentrer uniquement sur les problèmes liés à notre domaine. Nous rappellons, si besoin, les notions nécessaires à chaque début de chapitre.

Chapitre 1 : État de l'art

Dans le chapitre 1 nous introduisons les fondements mathématiques des outils que nous utilisons ensuite pour plonger un graphe dans une surface. Ce plongement est réalisé grâce à deux formules dont nous expliquons l'origine. Puis, nous faisons une étude de la précision de numérisation de la surface choisie et expliquons ces implications pour nos besoins. À partir de la section 1.2, nous présentons les recherches effectuées dans le cadre du routage incluant le nommage et l'adressage. Nous abordons le routage glouton qui est au cœur des chapitres 1 et 2. Naturellement, cela nous mène aux problèmes d'estimation

des distances dans un réseau comme l'Internet. Puis, comme alternative à l'estimation des distances par exploration nous traitons des techniques d'extraction de graphes recouvrants qui permettent d'extraire des graphes où les arêtes ne s'intersectent pas. Ces constructions se révèlent essentielles dans les algorithmes de routage glouton. Les graphes recouvrants sont à la base de nombreux algorithmes de routage ou de navigation dans les surfaces planes car ils permettent, après un plongement, d'utiliser la métrique qui caractérise la surface (i.e une fonction distance).

Chapitre 2 : Adressage et Routage dans le disque de Poincaré

Dans le chapitre 2 nous abordons la mise en œuvre du routage glouton dans la surface hyperbolique au travers du modèle de Poincaré. Nous présentons et évaluons notre solution le plus précisément possible. Sans perte de généralité, nous montrons et expliquons que notre solution est transposable à tout type de réseaux de communications hétérogène statique (Internet) ou dynamique (MANet) où la nécessité d'un routage point-à-point se fait sentir.

Chapitre 3 : Réseaux virtuels et leurs Applications

Dans le chapitre 3 nous commençons par un bref rappel des différents types d'architectures. Puis, nous étudions les réseaux pair-à-pair destinés à diffuser des flux vidéos P2PTV. Nous traitons du passage à l'échelle dans le cadre de la redistribution vidéo dans les réseaux P2PTV. Ainsi, nous proposons et évaluons des mécanismes basés sur l'algorithme de sélection des pairs pour améliorer la croissance du réseaux P2PTV. Les résultats de simulations sont obtenus en simulant l'architecture d'un des plus grands fournisseurs de P2PTV en Europe *Zattoo*.

Chapitre 1

État de l'art

Sommaire

Dans ce chapitre, nous présentons l'état de l'art dans lequel se place notre contribution au domaine des réseaux de communication. Ce domaine regroupe un ensemble de concepts, de protocoles, d'algorithmes et de technologies dont la compréhension est considérée comme acquise. En effet, la machinerie qui se cache derrière les réseaux de communication existants est révélée entre autre dans deux ouvrages de référence du domaine : *Les Réseaux* [Tan04] et *Computer Networking* [KR09]. Toutefois, nous reviendrons sur quelques concepts fondamentaux qui, depuis une décennie, sont de nouveau sur le devant de la scène scientifique. La contribution de notre travail est de répondre au besoin croissant d'un algorithme qui avec une quantité d'information réduite, soit capable d'acheminer l'information de manière décentralisée dans les réseaux filaires ou non

filaires, d'Internet aux réseaux *ad hoc*. En conséquence, nous présentons dans ce chapitre des travaux abordant les questions liées à ce besoin.

Dans la section 1.1, nous introduisons les concepts et les outils nécessaires à la compréhension de nos travaux. En effet, nous présentons les modèles du plan hyperbolique ainsi que les propriétés géométriques de cette surface. Puis, nous présentons, avec un algorithme, une utilisation concrète du modèle hyperbolique choisi. Dans la deuxième section 1.2, nous rappelons les concepts du nommage, de l'adressage et de la résolution de nom. Puis, dans la troisième section 1.3, nous précisons les problèmes auxquels nous devons faire face. Enfin, dans la dernière section 1.4, nous présentons l'état de l'art spécifique à notre solution.

1.1 Vers un modèle du plan hyperbolique

Dans cette section, nous introduisons les notions mathématiques qui permettent à notre algorithme de fonctionner. Nous travaillons avec une surface précise, le plan hyperbolique. Plusieurs propriétés géométriques ou analytiques définissent l'espace hyperbolique. Afin d'étudier ces propriétés les mathématiciens ont élaboré des représentations du plan hyperbolique dans le plan euclidien. Ces représentations sont le résultat d'un plongement permettant de visualiser et de travailler dans le plan euclidien, tout en conservant des propriétés du plan hyperbolique (e.g. la notion d'angle ou de droite parallèle).

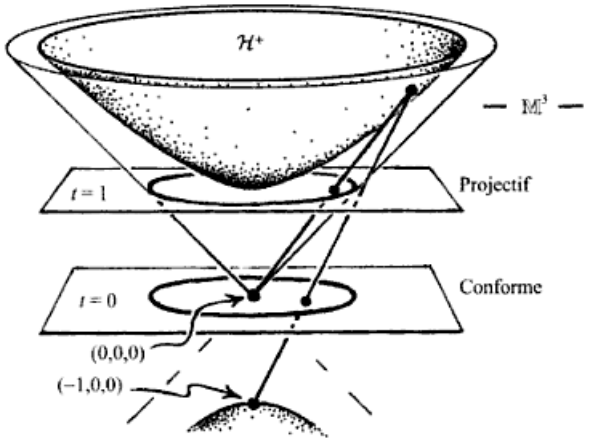

FIGURE 1.1 – Hyperboloïde à deux nappes - extrait du livre "À la découverte des lois de l'univers" par R. Penrose

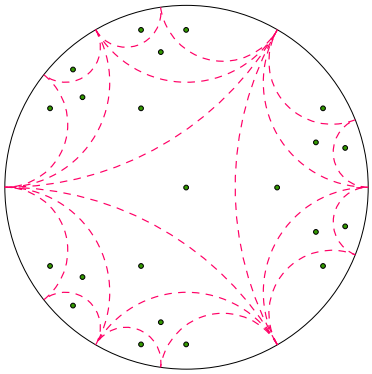

FIGURE 1.2 – Modèle conforme du plan hyperbolique

Par la suite, nous utiliserons le plan hyperbolique pour naviguer dans un graphe d'où son importance. L'hyperboloïde à deux nappes dont la partie supérieure est notée $\mathcal{H}^+$ sur la figure 1.1, est un modèle qui permet de travailler avec la géométrie hyperbolique. De plus, l'hyperboloïde est aussi une représentation de l'espace de Minkowski $\mathbb{M}^3$. La géométrie hyperbolique à deux dimensions et l'espace de Minkowski à trois dimensions $\mathbb{M}^3$ sont directement reliés à la représentation conforme de Beltrami, comme le montre la figure 1.1. Nous verrons l'intérêt du lien entre ces modèles en fin de section. La représentation

conforme de Beltrami (dit de Poincaré) s'obtient en projetant l'espace $\mathcal{H}^+$ depuis le pôle sud (-1,0,0) de l'hyperboloïde sur l'intérieur du cercle unité dans le plan équatorial $t = 0$. Autrement dit, le modèle conforme de Beltrami (dit de Poincaré) est construit par projection stéréographique [RP07]. Pour cette raison, la représentation du plan hyperbolique de la figure 1.2 est parfois nommée "modèle conforme" du plan hyperbolique souvent appelé "disque de Poincaré" (de Beltrami). La transformation permettant de passer de l'espace de Minkowski au disque de Poincaré légitime les équations que nous présentons ensuite pour le pavage du plan hyperbolique. Nous allons maintenant étudier plus en détail les transformations qui opèrent à l'intérieur du disque de Poincaré.

1.1.1 Transformations de base

Tout d'abord quelques précisions. Les points à l'infini du disque de Poincaré sont les points inatteignables du cercle euclidien. Le cercle, noté $\mathcal{U}$, est composé des nombres complexes dont le module vaut une unité, ces points forment la limite ou fermeture du disque. En termes mathématiques, les éléments du cercle $\mathcal{U}$ sont $\{z \in \mathbb{C} \mid |z| = 1\}$. Si le plan euclidien est muni d'un repère orthogonal normé dont un axe représente les imaginaires alors cela permet d'identifier les coordonnées en nombres complexes. L'ensemble des points intérieurs à ce cercle représente alors l'ensemble des points du plan hyperbolique, c'est le disque unité ouvert $\mathbb{D}\{z \in \mathbb{C} \mid |z| < 1\}$. Ouvert signifie que les points du disque limite sont exclus de l'ensemble $\mathbb{D}$. Plusieurs modèles existent pour représenter le plan hyperbolique dans le plan euclidien chacun possédant des propriétés plus ou moins semblable. Dans la suite du mémoire, nous ne traiterons du plan hyperbolique qu'au travers du disque de Poincaré.

Espace métrique

Nous introduisons ici les notions et définitions relatives aux distances et aux espaces métriques. Un espace métrique est un espace topologique où la distance entre chaque point peut être définie. La notion de métrique permet de définir une distance entre deux objets. Une distance d sur un espace X est une application $d : X \mapsto X \to [0; +\infty[$ telle que :

- $d(x, y) = 0$ Si et seulement si $x = y$ (similarité) ;
- $d(x, y) >= 0$ (non-négativité)
- $d(x, y) = d(y, x)$ (symétrie)
- $d(x, z) =< d(x, y) + d(y, z)$. L'inégalité triangulaire implique qu'un chemin allant directement d'un point x à z est plus court qu'un chemin allant de x à z en passant par y.

Donc un espace métrique est un couple (X, d) où d est une fonction distance sur X. Un outil primordial pour travailler dans cette surface est l'équation 1.1 qui permet de calculer la distance minimale entre deux points quelconques du disque représenté figure 1.2. Pour calculer la distance entre deux points, par exemple $z, w \in \mathbb{C}$, nous utilisons les coordonnées sous leur forme cartésienne. L'espace $\mathbb{D}$ et la distance $d_{\mathbb{D}}$ forment l'espace métrique. Nous utiliserons cette métrique pour acheminer l'information entre les pairs du

réseau.

$$d_{\mathbb{D}}(z,w) = acosh(1 + 2\frac{|z-w|^2}{(1-|z|^2)(1-|w|^2)}) \tag{1.1}$$

Nombres complexes

Avant de présenter les transformations complexes de base, faisons un bref rappel des différentes manières de noter ces nombres. Un nombre complexe peut être écrit :

- soit sous sa forme cartésienne $z = a + ib$ avec $|z| = \sqrt{a^2 + b^2}$,
- soit sous sa forme polaire $z = |z| * e^{iarg(z)}$,
- soit avec la formule de Euler $e^{i\theta} = cos(\theta) + isin(\theta)$ où $cos(\theta) = \Re(z)/|z|$ et $sin(\theta) = \Im(z)/|z|$ d'où $z = |z| * (cos(\theta) + isin(\theta))$,

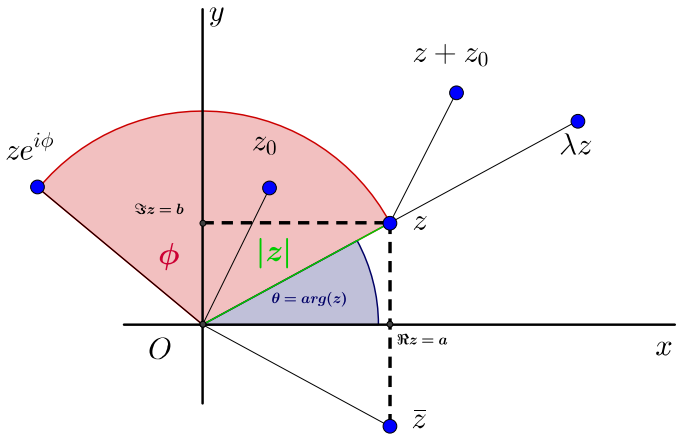

FIGURE 1.3 – Transformations complexes élémentaires

Les transformations complexes les plus élémentaires sont résumées sur la figure 1.3. Des informations détaillées se trouvent dans le chapitre « Möbius transformations » de [Che08]. Pour résumer, la figure 1.3 montre les déplacements engendrés par les opérations d'addition et de multiplication. Par exemple, la multiplication par une exponentielle imaginaire $z \mapsto e^{i\phi}z$ effectue une rotation de centre O et d'angle ϕ. Cela revient à additionner les angles en notation polaire. Enfin, la composition de ces transformations, comme $z \mapsto az + b$, s'appelle une transformation linéaire. Dans notre cas, les transformations qui nous intéressent plus particulièrement sont les transformations bilinéaires ou transformations de Möbius qui sont des fonctions rationnelles $T : \mathbb{C} \cup \{\infty\} \mapsto \mathbb{C} \cup \{\infty\}$ de la forme $T(z) = \frac{az+b}{cz+d}$ où $a, b, c, d \in \mathbb{C}$ sont constants et $ad - bc \neq 0$ [Sti92]. Aussi, la transformation clef qui assure la conservation du disque de Poincaré lors des transformations est l'inversion $z \mapsto 1/z$. Car $T(z)$ est le produit des inversions puisque l'équation peut s'exprimer comme produit des transformations de la forme : $z \mapsto 1/z$, $z \mapsto xz$, $z \mapsto y + z$ où $x, y \in \mathbb{C}$ chacune desquelles peut être vue comme un produit des inversions [Sti92].

Nous venons de voir brièvement l'ensemble des transformations élémentaires du plan complexe figure 1.3. Nous allons dans la suite présenter et utiliser ces transformations, notamment celles de Möbius, au sein du disque unité ouvert.

1.1.2 Transformations conformes

Le concept mis en avant est celui de la "géométrie conforme". Comme expliqué par Roger Penrose, la géométrie conforme s'intéresse à la forme (à l'échelle infinitésimale) mais pas aux grandeurs (à la taille). En général, la transformation d'une région (dite ouverte) du plan en une autre est conforme, si elle préserve les formes infinitésimales, déforme celles qui sont de taille finies [RP07], comme par exemple l'inversion. Afin de nous faciliter la compréhension de ce concept R. Penrose considère de petits cercles (infinitésimaux) tracés sur le plan. *Après une transformation conforme, de petits cercles peuvent être distendus ou contractés, mais ils ne se déformeront pas en ellipse.* De plus, une transformation conforme préserve les angles entre les courbes. Cette dernière propriété est primordiale pour l'usage que nous faisons du plan hyperbolique car cela nous permet de conserver les outils euclidien pour les angles. En effet, dans le plan hyperbolique *la notion d'angle entre deux courbes, à leur point d'intersection, est exactement la même que la mesure euclidienne de l'angle entre deux courbes à leur point d'intersection* [RP07].

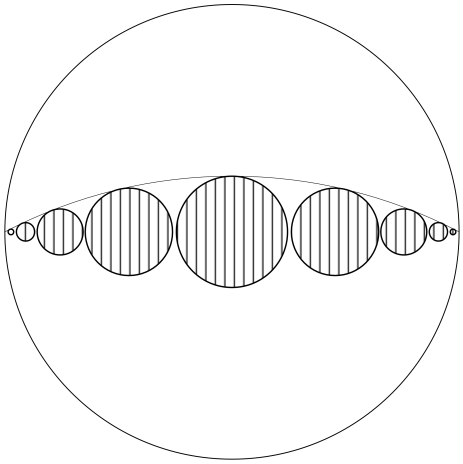

FIGURE 1.4 – Transformation conforme ne préservant pas la taille des éléments

La figure 1.4 est un exemple qui montre le résultat d'une transformation conforme dans le plan hyperbolique. Une telle transformation ne préserve pas la taille des éléments. En effet le dessin de la région centrale de la figure 1.4 est transformé en de tous petits dessins correspondants aux nouvelles positions du cercle dans l'espace. Les plus petits cercles sont situés juste avant le cercle limite. C'est le même principe que R. Penrose explique avec les petits cercles infinitésimaux. Concrètement, les transformations que nous allons détailler permettent de déplacer par translation ou rotation des points au sein du disque de Poincaré tout en conservant les propriétés infinitésimales comme expliqué avec les cercles. Cependant, lors d'une transformation la distance entre deux points ne doit pas être modifiée. C'est une condition nécessaire au fonctionnement de notre algorithme de routage. C'est pourquoi, nous nous intéressons au isométrie. En géométrie, une isométrie est une transformation qui conserve les longueurs. Les mathématiciens parlent aussi des automorphismes du disque de Poincaré, noté $\mathcal{A}ut(\mathbb{D})$, qui sont les transformations complexes préservant le disque [Bia08, Emm08]. Le groupe des automorphismes conformes

du disque unité est l'ensemble des isométries holomorphes de la métrique hyperbolique mais aussi de la distance hyperbolique. Car, un chemin γ joignant z à w est appelé arc géodésique hyperbolique, si $d_{\mathbb{D}}(z,w) = l_{\mathbb{D}}(\gamma)$. Où $l_{\mathbb{D}}(\gamma)$ est la longueur du chemin entre z et w. Cela signifie donc que la longueur est minimale. Or les seuls chemins minimaux existants dans les surfaces courbes tel que le plan hyperbolique sont les géodésiques.

Sur la figure 1.5, les arcs de l'arbre semble se rétracter lorsqu'ils approchent du cercle limite ou sur la figure 1.4 le diamètre des cercles semble diminuer. Or ce n'est qu'une impression de notre point de vue d'observateur. En effet, la longueur des arcs verts de la figure 1.5 reste strictement la même au sein du plan hyperbolique, de même pour le diamètre des cercles 1.4. Le groupe des isométries du disque pour la métrique de Poincaré s'identifie au groupe des bijections holomorphes du disque [Pan05]. Toutes les bijections holomorphes de $\mathbb{D} \mapsto \mathbb{D}$ sont de la forme : $\frac{az+b}{\bar{b}z+\bar{a}}$. Cette forme représente un ensemble des transformations de Möbius qui préserve l'orientation (e.g. un vecteur). C'est un cas particulier de la transformation $T(z) = \frac{az+b}{cz+d}$. Pour le disque unité ouvert dans le plan complexe, si $\alpha \in \mathbb{D}$ et p est un élément de $\mathbb{D}$ alors la transformation paramétrée définie par $f_{\alpha,p} = e^{i\alpha}.\frac{z-a}{1-\bar{a}z}$ préserve le disque unité [Ghy06]. Cela nous mène à la première équation indispensable à notre algorithme :

$$z \mapsto \frac{z_p + p}{1 + \bar{p}z_p} = \frac{e^{i\theta}w + p}{1 + \bar{p}e^{i\theta}w} \tag{1.2}$$

Étant donnés deux points quelconques, z_p et $p \in \mathbb{D}$, l'un des éléments du groupe de transformation envoie le premier point p sur le second z_p où z_p est paramétré par une longueur d'arc et un angle [BM06]. Les bijections holomorphes sont les restrictions au disque des transformations bilinéaires de la forme de l'équation 1.2. Pour en arriver à l'équation 1.2, il faut considérer z_p au travers de ses coordonnées polaires. Les coordonnées polaires hyperboliques centrées se définissent en fixant un point p dans $\mathbb{D}$ appelé centre des coordonnées polaires basées en p. Pour $\theta \in \mathbb{R}$, $\rho_\theta(p)$ désigne l'unique rayon géodésique hyperbolique partant de p et tangent au vecteur unité euclidien $e^{i\theta}$ en p. Par exemple dans le cas $\theta = 0$, le vecteur unité tangent euclidien est confondu avec le rayon (0,1) et $\rho_0(p)$ est appelé la géodésique hyperbolique horizontale partant de p parce que le vecteur unité tangent en p est horizontal. Soit $z_p(s,\theta) = w$ et $s = d_{\mathbb{D}}(p,w) \mapsto z_p(s,\theta)$ où s est la distance hyperbolique de p à w et θ est l'angle entre le rayon géodésique horizontal $\rho_0(p)$ et le rayon $\rho_\theta(p)$ qui contient w. En fait, le centre des coordonnées peut être vu comme un cercle de rayon la longueur d'arc s [MM06, BM06]. Alors z_p est la paramétrisation de la longueur d'arc hyperbolique de $\rho_\theta(p)$. Deux rayons géodésiques avec des vecteurs unités tangents distincts en p sont disjoints sauf en p d'où ils émanent. Ou, autrement dit, pour chaque point w dans $\mathbb{D} - \{p\}$, il y a un unique rayon géodésique $\rho_\theta(p)$ qui contient w les coordonnées hyperboliques polaires du point w par rapport au centre p qui sont la paire $(s,\theta) = w$.

L'équation 1.2 fait son apparition en informatique au travers de travaux de visualisations où le modèle de Poincaré est utilisé pour naviguer dans de larges hiérarchies de données. Car l'équation 1.2, présentée dans l'article de [LRP95] ou plus tard de [Miq00], permet de déplacer le contenu du disque. Donc ces travaux nous montrent que le modèle de Poincaré associé à une équation de transformation particulière fonctionne correctement une fois numérisé à la précision de calcul près. En effet, avec cette équation nous pouvons

calculer de manière distribuée les coordonnées des nœuds qui sont les sommets des ∞-gones décris dans la sous-section suivante. Néanmoins, la fonction distance (cf. équation 1.1) joue aussi un rôle important. C'est d'ailleurs le point faible du travail de [Miq00] qui nous a tout de même permis de prendre en main le travail de Lamping. Cependant l'équation 1.2 n'est pas suffisante pour satisfaire nos besoins. Lorsque nous observons le plan hyperbolique d'un œil extérieur, nous voyons des chemins non rectilignes ou en arc de cercle. Cela est dû à la courbure de l'espace et implique lors du déplacement une variation d'un vecteur direction. C'est une autre équation qui nous permet de corriger l'écart angulaire. Chaque nouvelle position devient le nouveau centre des coordonnées polaires pour le déplacement suivant. Or, du fait de la courbure de l'espace hyperbolique, cela entraîne une variation du vecteur direction qui définit la trajectoire du déplacement. Donc, si nous n'effectuons pas la correction angulaire du vecteur direction les calculs suivants seront erronés. En effet, l'équation 1.2 agit dans le disque. Or dans le disque un déplacement rectiligne suit une géodésique qui d'un point de vue euclidien est une courbe.

Pour mieux comprendre, il est nécessaire de passer des mathématiques à la physique avec la cinématique du point mobile car nous devons connaître l'interprétation des déplacements au sens physique. En effet, la cinématique du point nous permet de décrire, à l'aide des outils mathématiques, la trajectoire suivie par un point que nous déplaçons avec l'équation 1.2. De plus, cette interprétation nécessite aussi un système de coordonnées adapté au problème. Pour ce faire, nous considérons dans le plan euclidien un repère orthonormé Oxy dans lequel nous notons $x(t)$ et $y(t)$ deux fonctions dépendants du temps. Notre point mobile n'est soumis qu'à la seule force d'inertie induite par la courbure négative de l'espace. C'est une force identique qui agit sur le déplacement d'un avion sur la sphère terrestre, à la différence que dans une sphère le signe de la courbure est positif. Le déplacement de notre point peut être caractérisé par un vecteur unitaire tangent à la trajectoire dans le sens du mouvement et par un vecteur unitaire normal perpendiculaire au vecteur tangent et dirigé vers le centre de courbure. Le sens du mouvement démarre du centre des coordonnées polaires vers le point paramétré (e.g. $z_p(s, \theta)$). Maintenant, si nous prenons le point α de coordonnée $x(t)$ et $y(t)$ alors α décrit une courbe paramétrée en fonction du temps et $x(t)$ et $y(t)$ sont appelés les équations paramétriques de la courbe. Le paramètre étant la variable temps. Cette courbe est aussi appelée la trajectoire de α. Comme notre courbe, notée $\alpha(t)$, est régulière et paramétrée en fonction du temps, les lois de la physique nous permettent d'exprimer le vecteur unitaire de la façon suivante : $Tg(t) = \frac{\alpha^{'}(t)}{|\alpha^{'}(t)|}$.

Le pont entre mathématique et physique s'obtient grâce à l'espace de Minkowski qui nous donne la possibilité d'interpréter l'équation 1.2 à notre convenance. En effet, l'espace des vitesses en théorie de la relativité est aussi l'espace hyperbolique (unitaire) $\mathcal{H}^+$ dans lequel la vélocité ρ mesure la distance hyperbolique le long de $\mathcal{H}^+$ avec $\rho = atanh(v)$. Notons que les frontières qui décrivent l'infini de la géométrie hyperbolique avec $\rho = \infty$ représentent l'inaccessible vitesse limite dont la valeur est 1. Cela est équivalent à dire que le module d'un nombre complexe interne au disque de Poincaré ne peut jamais atteindre la valeur 1. Nous écrivons même que $|z| = vitesse = tanh(\rho)$. Dans nos déplacements la vélocité est constante et a pour valeur la longueur d'arc qui caractérise le point z_p en coordonnée polaire. Nous allons voir dans la suite comment calculer la valeur de ρ.

Ce qui est un autre point capital pour la justesse de notre algorithme. Enfin, ce pont entre les disciplines nous offre la possibilité de reformuler l'équation 1.2 en remplaçant le terme z par la fonction paramétrique $\alpha(t) = tanh(\rho)$. Ainsi nous pouvons déterminer la trajectoire suivit par le vecteur unitaire tangent et calculer sa nouvelle position égale à la dérivée de l'équation 1.2 divisée par sa dérivée normalisé. Ce qui nous permet d'obtenir la deuxième équation indispensable à notre algorithme. Bien que l'équation 1.3 soit donnée dans [Miq00] aucune justification n'est fournie.

$$z \mapsto \frac{e^{i\theta} + w.p}{1 + \bar{p}e^{i\theta}.w} = e^{i\theta}\frac{1 + pwe^{-i\theta}}{1 + \bar{p}we^{i\theta}} = e^{i\theta}\frac{1 + \bar{z}p}{1 + z\bar{p}}, z = e^{i\theta}w \tag{1.3}$$

Les deux équations que nous venons de présenter permettent respectivement de calculer une nouvelle position et une nouvelle direction pour un point dans le plan hyperbolique. Le calcul d'une nouvelle position NP s'apparente à un déplacement le long d'une géodésique émanant de la position initiale p centre des coordonnées polaires. NP est défini par sa longueur d'arc ou vélocité et un angle par rapport à $\rho_0(p)$. Ces deux équations associées à la fonction distance sont la clef de voûte de notre algorithme de pavage du disque de Poincaré.

1.1.3 Algorithme distribué de pavage

Nous allons détailler ici comment les coordonnées de l'espace hyperbolique sont calculées. La partie suivante a pour objectif d'expliquer concrètement comment le pavage est réalisé de manière algorithmique. Nous utilisons les transformations précédentes, définies avec les équations 1.2 et 1.3. Nous utilisons une propriété de pavage du plan hyperbolique laquelle permet de réaliser un pavage avec des polygones quelconques, nommés les q-gones. Sur la figure 1.5 les q-gones sont des triangles (i.e. $q = 3$) représentés en pointillés.

Le plan peut être vu comme une carte combinatoire à deux dimensions dans laquelle chaque q-gone (ici triangle) est une région distincte délimitée par les côtés du q-gone. Les q-gones sont idéaux, c'est à dire que les sommets sont à l'infini (i.e. sur le cercle unité). Cette propriété nous assure l'unicité des coordonnées hyperboliques. Chaque pavage est représenté par une notation de la forme $\{p, q\}$. Cette notation est appelée un *symbole de schläfli*. Il existe un pavage hyperbolique $\{p, q\}$ pour tout couple (p, q) respectant $(p - 2) * (q - 2) > 4$ où chaque polygone à p côtés et q d'entre eux à chaque sommet. Ainsi, la valeur de q conditionne le degré de l'arbre figure 1.5. Cet arbre représente notre espace d'adressage virtuel embarqué dans le système P2P.

Un exemple d'arbre hyperbolique avec $q = 3$ est montré sur la figure 1.5. Sur la figure, nous pavons le plan avec des triangles idéaux comme sur la figure 1.2. Autrement dit les côtés du triangle sont tangents entre eux à l'infini, chaque sommet du triangle est à l'infini. Dans un pavage, p est le nombre des côtés des polygones du *primal* (les arêtes vertes figure 1.5) et q est le nombre des côtés des polygones du *dual* (les triangles en pointillés figure 1.5). En posant p égal à l'infini, le *primal* se transforme en un arbre régulier théoriquement infini de degré q.

Comme expliqué dans la thèse de [Emm08], dans $\mathbb{D}$, la somme des angles d'un triangle est strictement inférieur à π et pour tous r, p, q entiers tels que $1/r + 1/p + 1/q < 1$, il existe un triangle T ayant pour angles aux sommets $\pi/r + \pi/p + \pi/q$. Le groupe des

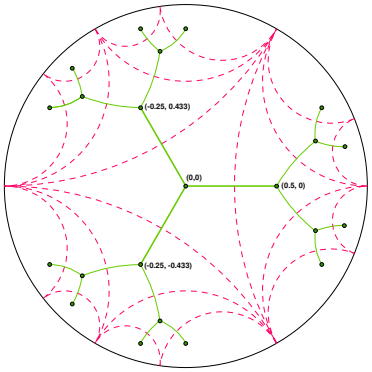

FIGURE 1.5 – Pavage de triangle

isométries engendré par les réflexions par rapport aux côtés de T est noté $\rho_0(r,p,q)$, et l'ensemble de ses éléments préservant l'orientation est appelé $\rho(r,p,q)$. C'est le groupe de triangle associé à T. Soit le groupe de triangle $\rho(r,p,q)$ qui crée un pavage $\mathcal{P}$ du plan hyperbolique $\mathbb{D}$. Si nous posons $r=2$, le pavage P est alors constitué de p-gones de côtés $2c$ avec $coshc = \frac{cos(\pi/p)}{sin(\pi/q)}$. Comme illustré sur la figure 1.6 les angles aux sommets sont tous égaux à $2\pi/q$.

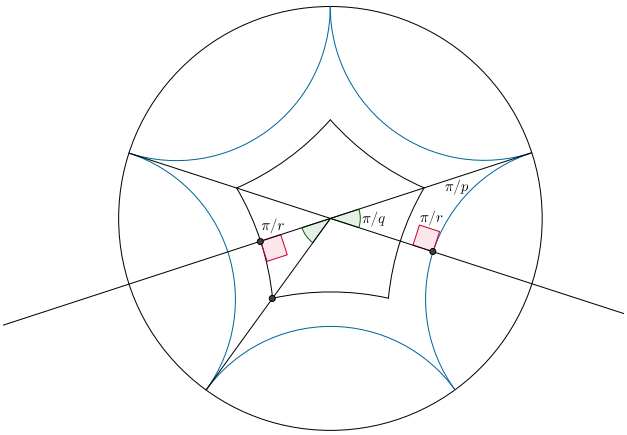

FIGURE 1.6 – Longueur du côté du polygone

La figure 1.5 montre un exemple de pavage avec des q-gones et les p-gones correspondants où p est égal à l'infini. En effet, la structure arborescente est liée à la valeur de p qui est l'infini. La figure 1.6 montre que la distance c, qui représente le côté d'un p-gone varie en fonction du nombre de côtés du p-gone. En effet, lorsque p est à l'infini, cela implique des q-gones idéaux et la distance c est à son maximum. Cette distance c nous permet de calculer notre première coordonnée et représente la vélocité de notre point. Donc $c = acosh(\frac{cos(\pi/p)}{sin(\pi/q}))$.

Ces propriétés, nous permettent de proposer un algorithme distribué 1 qui garantit un plongement glouton et ne requiert aucun traitement sur le graphe de départ. C'est la

structure arborescente crée par les ∞-gones que nous utiliserons pour affecter les coordonnées virtuelles aux nœuds. Dans l'algorithme, le *pas* (l.3) représente la longueur $2 * c$. De plus, le $pas/2$ indique la distance optimale c pour créer une nouvelle région et ainsi assurer l'unicité des adresses. Chaque nœud peut donner q adresses. La valeur de q est fixée par le premier nœud plongé. Une évaluation de l'impact du choix du degré de l'arbre q-régulier est faite dans [CTBM11].

Algorithme 1 : Calculer les coordonnées des descendants d'un nœud

```
1  CalcFilsCoords(nœud, q) ;
2  begin
3      pas ← acosh(1/sin(π/q));
4      angle ← 2π/q;
5      for i ← 1, q − 1 do
6          noeud.Coords.rotationGauche(angle);
7          Fils.Coords ← noeud.Coords;
8          Fils.Coords.translation(pas);
9          Fils.Coords.rotationDroite(π);
10         StockFuturFilsCoords(Fils.Coords);
11     endfor
12 end
```

L'angle (l.4) détermine le nombre de régions qui sont créées. Chaque nœud calcule les coordonnées de ses futurs fils afin de les assigner le moment venu. La ligne 9 permet juste de faciliter le calcul de l'angle pour la rotation, ainsi nous tournons dans le même sens $q-1$ fois et avec le même angle. Notre but est de calculer de manière locale la position des sommets voisins. Nous venons de présenter un algorithme distribué permettant de paver le plan hyperbolique avec des ∞-gones. Cet algorithme nous permet de calculer les positions des sommets des poly-gones de proche en proche. En effet, chaque sommet calcule les positions pour les n sommets suivants. Ces positions sont des coordonnées hyperboliques qui sont structurées en un arbre théoriquement infini. L'algorithme de pavage du plan hyperbolique que nous venons de présenter peut-être entièrement distribué. En effet, chaque sommet calcule les coordonnées des nœuds suivants auxquels il est relié en fonction de ses propres coordonnées. Nous pouvons donc utiliser cet algorithme pour calculer des coordonnées dans le plan hyperbolique et créer un espace d'adressage. Nous sommes à présent capables de calculer des coordonnées hyperboliques. Toutefois, l'utilisation du modèle de Poincaré, comme tout modèle embarquant l'infini est limité en informatique par la précision de calcul.

1.2 Du nommage à l'adressage

À partir de cette section, le réseau d'interconnexion, Internet, nous servira de cas d'école afin de fournir des exemples concrets aux problèmes qui se posent. Toutefois, nous abordons le problème du routage pour l'ensemble des réseaux soumis à la volatilité de ses participants (i.e. MANet, P2P). Car notre solution est naturellement transposable et aisément combinable avec des contraintes supplémentaires (e.g. délai, bande passante).

1.2.1 La théorie des graphes

La théorie des graphes est l'étude des graphiques, des structures mathématiques utilisées pour modéliser les relations entre les paires d'objets d'un ensemble donné. Pour un graphe la topologie représente la structure de celui-ci, autrement dit cela caractérise l'agencement des relations entre les objets. Dans ce contexte, un graphe se réfère à un ensemble de points et à une collection de lien qui relient les paires de points. Formellement, un graphe (fini) $G = (V, E)$ est la donnée de deux ensembles finis où V est l'ensemble des sommets (ou nœuds) et E est l'ensemble des arêtes (ou liens). Si u est un sommet, $deg(u)$ s'appelle la valence ou le degré de u (i.e. le nombre d'arêtes issues et incidentes à u). Deux types de relations existent entre les pairs de sommets :

- Soit une relation symétrique dans le cas d'un graphe dit non orienté.
- Soit une relation asymétrique dans le cas d'un graphe orienté.

Un graphe tracé dans un espace à deux dimensions dont les arêtes ne s'intersectent pas est dit planaire, la réalisation d'un tel dessin est appelée un plongement du graphe dans le plan. Le concept de plongement peut-être exprimé ainsi :

Définition 1.2.1 ***Plongement de graphes*** *: Le plongement d'un graphe dans un plan est le résultat du tracé des arêtes du graphe dans ce plan à la condition que ses arêtes soient des courbes continues entre deux points qui ne s'intersectent pas avec une des autres arêtes.*

Du concept de graphe à celui de réseaux, il n'y a qu'un pas : un réseau est un graphe dans lequel il y a des échanges d'information entre les objets physiques ou virtuels. Comme nous l'avons vu, la topologie représente la structure d'un graphe. Une topologie de réseau est un modèle d'agencement des interconnexions des divers éléments. Plus précisément, toute topologie de réseau particulière est déterminée par la représentation graphique de la configuration des connexions entre les objets.

L'étude topologique en théorie des réseaux a conduit, dans la dernière décennie à trois grandes catégories de réseaux, caractérisés par la distribution du degré de leurs nœuds. La distribution du degré des nœuds peut être aléatoire, en loi de puissance ou petit monde (implique un niveau de clusterisation/clique). Les réseaux dits petit monde ne disposent pas d'une définition claire. Dans ses travaux, Schnettler revient sur 50 ans de recherche sur le concept maintenant reconnu d'effet petit monde [Sch09].

Le point clef qui attira toute l'attention sur ces réseaux est l'acheminement de l'information. Cela commença en 1966 avec Stanley Milgram qui élabora une expérience pour évaluer l'effet petit monde. Son expérience consiste au transfert d'un paquet de proche en proche au travers des liens d'accointances de chacun des intermédiaires. L'émergence naturelle des trois types de réseaux est étudiée par Barabási dans [BB03, Bar09] qui présente entre autre l'évolution d'un graphe dans lequel chaque nouvel arrivant se connecte aux sommets existants avec une probabilité proportionnelle à leur degré. La distribution du degré des nœuds du graphe construit est alors en loi de puissance. Il qualifie le modèle d'attachement préférentiel du fait de l'analogie aux réseaux sociaux où un degré d'accointance élevé indique une popularité élevée.

L'acheminement de l'information au travers des nœuds d'un réseau nécessite d'identifier les entités (i.e. les nœuds ou les liens) à traverser afin de calculer, de manière centralisée ou décentralisée, un itinéraire jusqu'à une destination. Dans le domaine des réseaux,

le nom (ou étiquette ou label) permet d'identifier chacune des entités qui composent le graphe de connexion physique ou virtuel. Lors d'une communication, un message emprunte un itinéraire composé des noms des entités intermédiaires (i.e. nœud ou lien). En effet, les interactions entre les nœuds ne sont possibles que si les entités sont discernables, autrement dit, nommées. Seul un acheminement par inondation pourrait faire fi des noms. Nous parlons dans cette section du problème du nommage [Fra94].

De manière plus formelle, un nom est une chaîne de caractère unique S, générée avec un alphabet A, qui désigne un objet. Un espace de nommage NS est l'ensemble des noms S parmi lesquels tous les noms sont pris pour un ensemble d'objet donné. Un itinéraire représente un trajet entre un objet localisé par un nom source et un objet localisé par un nom destination. En 1978, John Shoch dans [Sho78] note trois concepts distincts pour les réseaux de communications :

- Les noms indépendants du lieu (de l'anglais *Location-independent*)
- Les noms dépendants du lieu (de l'anglais *Location-dependent*)
- Les itinéraires (de l'anglais *routes*)

Par rapport à l'architecture Internet actuelle un nom indépendant du lieu signifie que le nom reste associé avec un nœud quels que soient les changements du point d'attachement (par exemple, l'adresse MAC d'une machine allant d'un immeuble à un autre). Alors qu'un nom dépendant du lieu signifie que le nom est dépendant du point d'attachement (par exemple, l'adresse IP d'une machine allant d'un immeuble à un autre). Lorsque un nom permet aussi de localiser l'entité dans le réseau alors nous parlons d'une adresse (i.e. nom dépendant du lieu). Depuis, ces concepts ont été plusieurs fois repris dans la littérature [TZ01, AGM+04, SGF+10] notamment avec le regain d'intérêt pour l'acheminement de l'information dans les réseaux de grande taille. Cela nous mène à deux définitions que nous avons résumées ainsi :

Définition 1.2.2 ***Name-dependent model*** *: Les algorithmes d'acheminement requièrent un format d'adresse spécifique (i.e. un label de l'anglais labeled routing) dans le but de contenir des informations topologiques dans les adresses (topology-aware ou Name-dependent model). L'adresse informe sur la position du nœud dans le réseau (Location-dependent).*

La topologie permet l'étude structurelle d'un réseau. Ici, le choix de l'espace de nommage doit parfaitement épouser la topologie du réseau ou d'un graphe couvrant du réseau. Une limitation connue des schémas de routage dépendant du nom est la robustesse au changement topologique. Pourtant, nous verrons au chapitre 2 qu'avec un algorithme d'adressage adapté ce modèle présente de bon résultats. En outre, le modèle P2P est parfaitement conservé puisque aucun nœud particulier n'est nécessaire.

Définition 1.2.3 ***Name-independent model*** *: L'adresse des nœuds est choisie sans lien avec la position dans le réseau (en anglais Name-independent model). La topologie ne suit pas l'adressage et inversement (en anglais topology-unaware ou Location-independent). Les algorithmes d'acheminement ne tiennent pas compte du format des adresses.*

Dans ce deuxième cas les nœuds doivent s'appuyer sur un nœud central ou sur plusieurs nœuds repères (*landmark*) qui maintiennent des itinéraires à jour. En général ce modèle est

utilisé avec du routage explicite ou par la source où tout ou partie de l'itinéraire est décidé avant l'envoi (e.g. avec un oracle). Finalement, la seule méthode qui peut s'affranchir de nœuds spéciaux ou d'adresses est l'inondation puisque aucune connaissance de la topologie ou des adresses n'est utilisée pour diffuser un message. Or nous l'avons dit, cela ne permet pas de gérer un nombre croissant de nœuds.

1.2.2 Espace d'adressage

La taxonomie de Shoch est reprise en septembre 1981 dans la description des fonctionnalités du protocole Internet de la RFC791 qui spécifie que :

- Un nom indique : Que cherchons-nous ? (de l'anglais *What we seek ?*)
- Une adresse indique : Où est-ce ? (de l'anglais *Where it is ?*)
- Un itinéraire indique : Comment s'y rendre ? (de l'anglais *how to get there ?*)

Cependant, la définition d'une adresse comporte quelques ambiguïtés car localiser un objet sans l'identifier n'est pas possible alors que l'inverse est possible. Cela implique qu'une attention particulière doit être portée sur le rôle de l'espace d'adressage car une adresse peut jouer le rôle d'un nom. D'après l'analyse de [Fra94], Shoch essaye d'introduire une définition unificatrice de ce qu'est une adresse contrairement à Saltzer qui en 1982 revisite les concepts du nommage et de l'adressage pour les réseaux d'ordinateurs [Sal82]. Saltzer décrit quatre niveaux plutôt que trois : les services et les utilisateurs, les nœuds, les points d'attachements et les itinéraires. Le concept d'adresse au sens de Shoch est supprimé de la taxonomie de Saltzer où l'adresse d'un objet et le nom de l'objet auquel il est rattaché. Ainsi une adresse pourrait être le nom d'un nœud (comme une adresse MAC), ou le nom d'un point d'attachement (comme une adresse IP) ou même le nom d'un itinéraire (trajet calculé par la source).

Notre but n'est pas de donner une nouvelle taxonomie mais de bien comprendre les rôles attribués aux espaces de nommage et aux espaces d'adressage. Francis dans [Fra94] fait une étude beaucoup plus approfondie des quelques points essentiels que nous venons de résumer. Ces travaux s'accordent sur l'importance de disposer de plusieurs espaces de nommage / adressage et sur le besoin de clairement définir le rôle de chacun d'eux.

Les espaces d'adressage peuvent être élaborés de deux façons. Un espace d'adressage peut être hiérarchique (e.g. numéro de téléphone, IP) ou plat (e.g. le numéro de sécurité sociale, MAC). Dans la conception initiale d'Internet une adresse permettait de communiquer directement avec l'équipement associé, cette adresse était dite publique. Les équipements pouvaient tous se joindre grâce à un espace d'adressage global. En effet, la première propriété commune aux deux types d'adressage est l'adéquation entre le nombre d'équipements et le nombre d'adresses. Or la première faiblesse du protocole Internet est la capacité de son espace d'adressage en version 4. La pénurie d'adresse IPv4 a entraîné la mise en place d'un mécanisme de translation d'adresse dynamique (NAT dynamique) (de l'anglais *Dynamic Network Address Translation*) permettant d'associer une adresse publique à un sous-réseau privé [Tan04, KR09]. De ce fait, une adresse publique peut maintenant faire référence à un ensemble de machines masqué derrière un équipement implantant du NAT. Toutefois, le NAT a seulement ralenti la pénurie des adresses de la version 4 du protocole Internet alors que le déploiement des adresses de la version 6 (IPv6) offre un espace considérable de 2^{128} adresses. Cependant, la quantité des adresses

ne résout pas les problèmes plus profond que nous allons aborder.

Dans le cas d'un adressage plat, nous considérons que l'adresse ne localise pas les nœuds dans le réseau. Par exemple, les adresses MAC sont prévues pour identifier de manière unique les équipements. Néanmoins une adresse MAC ne permet pas de localiser un équipement parmi d'autres. C'est pourquoi, dans le cas d'un adressage à plat la localisation n'a plus vraiment de sens. Les adresses MAC sont associées aux nœuds indépendamment des points d'attachements alors que dans IP les adresses ne dépendent que du point d'attachement car une structure ou hiérarchie est respectée. En effet, des machines appartenant à un même sous-réseau ont des préfixes d'adresses égaux. De ce fait, les équipements sont localisés car l'adresse indique leur position dans la hiérarchie. C'est en 1977 que Kleinrock et Kamoun ont publié un article fondateur sur le routage hiérarchique [KK77]. Ils ont montré comment l'agrégation hiérarchique peut être utilisée avec des adresses de nœuds appropriées (comme IP) pour produire des tables de routage passant à l'échelle d'où l'intérêt pour un réseau d'interconnexion de grande taille. Le point clef est de grouper (ou agréger) les nœuds proches dans des clusters, puis les clusters dans des clusters de plus haut niveau, et ainsi de suite, dans une approche de bas vers le haut (bottom-up) [KK77]. Déterminer les clusters peut être fait avec un algorithme de groupement hiérarchique ou, comme c'est le cas dans l'Internet, avec un organisme comme l'ICANN (*Internet Corporation for Assigned Names and Numbers*). Sinon, faire émerger un cluster en environnement distribué n'est pas simple (cf. *bin covering*).

L'ICANN gère la distribution des blocs d'adresses IP donc le respect de la structure d'adressage. Dans un adressage hiérarchique si la structure d'adressage du réseau est respectée les adresses peuvent s'agréger afin de réduire le nombre d'entrées dans les tables de routage. Or le problème du routage dans l'Internet vient de deux choses. D'une part la taille des tables de routage dans les routeurs devient trop importante malgré les techniques mises en place pour modifier la spécification originelle (e.g. CIDR (Classless Inter Domain Routing)). Cela a pour effet de saturer les ressources des routeurs. Une des principales causes de ce phénomène est la désaggrégation des adresses du protocole Internet [BFCW09]. En effet, dans IP les adresses similaires impliquent la proximité au sein du réseau car les adresses réseaux respectent une hiérarchie. Les adresses structurées permettent une entrée unique (grâce à un préfixe) dans la table de routage pour représenter l'itinéraire vers un groupe de périphériques. En conséquence, cela diminue la taille des tables de routage dans les routeurs. Or la structure hiérarchique de l'adressage Internet n'a pas été rigoureusement respectée. Les routeurs doivent donc stocker des préfixes plus longs. Les opérateurs, eux-mêmes victimes de leurs propres choix économiques, sont à l'origine du phénomène avec les partenariats (*Peering*) et la distribution d'adresse multiple (*multi-homing*). D'autre part, les mises à jour des tables de routage sont de plus en plus lourdes et fréquentes. Actuellement, la solution consiste à augmenter les ressources (mémoire et puissance de calcul) des routeurs. De plus, la dernière version de IP ne permettra pas d'enrayer le phénomène puisque elle utilise des adresses plus longues. Donc, en l'état actuel, un retour au modèle de bout-en-bout semble plus que jamais compromis.

Pour résoudre ces problèmes, certains travaux proposent des mesures drastiques. En effet, John Day et al. [DMM08] ou Joe Touch et al. [TBD+11] proposent de faire table rase de l'existant. Ces deux solutions, dites *clean-state*, se basent sur une architecture réseau récursive plutôt que sur un ensemble de couches qui réalise chacune des fonctions

différentes. La nouvelle architecture réseau, présentée dans [DMM08], est basée sur le principe fondamental que le réseau est formé d'une couche dans laquelle les éléments communiquent selon les méthodes de communication inter processus distribuées (IPC). Alors que l'architecture de [TBD+11] reste basée sur un modèle en couche qu'ils disent plus flexible pour supporter la composition dynamique de service et l'adaptabilité. Leur idée est de créer un protocole de base réutilisable comme un canevas. Sinon dans un style beaucoup moins radical le travail des auteurs de [BFCW09] s'insère dans l'existant sans modification. Leur solution utilise des routeurs virtuels de sorte à agréger les adresses communes et ainsi diminuer la charge des routeurs. Dans ce mémoire, nous ne prétendons pas concevoir une nouvelle architecture pour Internet. Nous avons abordé le sujet pour montrer qu'une importante évolution est à venir. Si certaines solutions sont aussi radicales, c'est parce que le protocole Internet a encore d'autres faiblesses mais la plus critique selon nous est celle qui touche l'acheminement de l'information.

1.2.3 Résolution de nom

Avant de continuer sur le sujet de l'acheminement de l'information, introduisons le dernier concept permettant de lier les noms et les adresses ainsi qu'une raison supplémentaire pour la remise en cause des adresses du protocole IP, toutes versions confondues. Lorsque nous analysons le rôle des adresses du protocole Internet, nous remarquons que l'adresse est, par définition, utilisée pour identifier un objet de manière unique et pour indiquer où l'objet se trouve dans le réseau. En effet, avec l'utilisation qui est faite des adresses IP la sémantique d'une adresse signifie à la fois la localisation d'un objet et son identité. Autrement dit, l'adresse est utilisée comme un nom et une adresse. De ce constat est né un nouveau concept proposé par CISCO : le protocole de séparation LISP (de l'anglais *Locator/Identifier Separation Protocol* ou Loc/ID split). Le protocole de séparation de la localisation et de l'identification pour le nommage des périphériques dans l'Internet est étudié, entre autre, dans [Bon09, JCAC+10] et [Mey08]. LISP est une architecture dans laquelle un nom identifie un objet et un nom différent (l'adresse) indique sa localisation. Les changements topologiques engendrent seulement la modification de l'adresse. Aussi, l'adresse est censée suivre la topologie et permettre son agrégation. LISP doit permettre une séparation de l'espace de nommage et d'adressage, ainsi les nœuds communiquent au travers de leur nom et changent d'adresse de manière transparente si nécessaire. Le but étant de rendre le routage dans l'Internet plus flexible. C'est une autre alternative aux travaux de [DMM08] et [TBD+11]. En revanche, c'est un manque pour le travail des auteurs de [BFCW09] qui ne corrigent pas le problème de la sémantique des adresses IP.

S'ajoute à ce problème sémantique le *multi-homing* qui consomme des adresses supplémentaires et donc de l'espace dans les tables de routage. Car un équipement possède une adresse MAC et une adresse IP par carte réseau. De plus, si nous reprenons la taxonomie de Shoch, quel que soit le type d'adressage (i.e. plat ou hiérarchique), alors un objet est localisé à plusieurs endroits puisque qu'il a plusieurs adresses. Cette taxonomie ne pose pas de problème lorsque la relation adresse - nom est bijective et que l'espace d'adressage est hiérarchique. Or le *multi-homing* induit plutôt une relation surjective, plusieurs adresses font références à un nom.

Enfin, nous avons jusqu'à présent éludé ce point primordial que Saltzer souligne dans

son travail. C'est l'importance du lien entre les espaces de nommage. Lorsque plusieurs espaces de nommage sont utilisés, un mécanisme de liaison est nécessaire mais ce type de mécanisme n'est pas nouveau en informatique. Les fonctions de translation d'adresse (ou cartographie) peuvent être, dans le cas d'une architecture de mémoire de l'ordinateur, une unité de gestion mémoire (MMU) entre le processeur et le bus mémoire. Une adresse logique est alors l'adresse à laquelle un élément (cellule de mémoire, élément de stockage) semble résider dans la perspective d'un programme d'exécution. En réseau aussi, une adresse logique peut être différente de l'adresse physique dû à l'exploitation d'un traducteur d'adresses ou d'une fonction de cartographie (*Binding* en anglais). Par exemple, une solution actuellement déployée dans l'Internet utilise des serveurs de nom de domaine implantant le service DNS (*Domain Name Service*) qui permettent par un envoi direct de la requête au serveur d'obtenir l'adresse IP publique d'une machine. Ce service permet de traduire des adresses humainement lisibles avec les adresses de la couche transport. Donc pour IP la séparation des rôles de l'adresse implique un espace de nommage supplémentaire et un mécanisme de résolution de nom. Un autre mécanisme très étudié et essayé par les développeurs de LISP est celui des tables de hachage distribuées, mais nous en parlerons dans un autre chapitre.

Dans ce mémoire, l'objectif est de construire un système P2P basé sur un algorithme de routage passant à l'échelle et permettant plus de flexibilité dans les communications à l'aide d'une structure de données. Cela est possible en utilisant au moins deux espaces de nommage et une couche de traduction d'adresses performante (e.g. DHT). C'est pour cette raison que nous avons réintroduit quelques concepts clés. Nous pouvons maintenant expliciter les problèmes auxquels nous devons faire face.

1.3 Problème

Le système de routage de l'Internet atteint actuellement un point critique en raison de l'incapacité des équipements à assumer la charge des protocoles qui deviennent trop gourmands en ressource. Des solutions capables de calculer le plus court chemin en termes de sauts entre une source et une destination dans un graphe existent depuis longtemps. Clairement, à une époque, la théorie des graphes, au travers de Dijkstra, Bellman, Ford et Fulkerson, a apporté une forte contribution pour résoudre la question du routage. Gregory G. Finn [Fin87] :

> Routing has been approached as a graph theoretic question, and work in this area began at least as early as the middle 1950's. Key works are Dijkstra's discussion of the shortest path between two network nodes, calculated from the network distance matrix, and the distributed routing algorithm of Ford and Fulkerson.

L'algorithme de Dijkstra est toujours utilisé pour choisir le plus court chemin en premier dans OSPF (de l'anglais *Open Shorthest Path First*). OSPF est un protocole à "état de lien" qui permet à un nœud d'avoir une vision globale du réseau utilisant OSPF. L'algorithme de Bellman-Ford est utilisé comme protocole de routage de l'information (de l'anglais *Routing Information Protocol* RIP). RIP est un protocole à "vecteur de distance" qui permet à un nœud d'avoir une vision locale. Les premières versions des protocoles

OSPF et RIP sont respectivement définies dans les RFC1131 et RFC1058. Par la suite, des variantes de ces protocoles ont été créées pour répondre à de nouvelles contraintes. Mais une fois de plus ils atteignent leurs limites. Plus d'informations sur les différents protocoles de routage Intra ou Inter système autonome, sont données dans les ouvrages de référence [Tan04, KR09]. Comme nous l'avons expliqué dans la section précédente la combinaison des problèmes liés à l'espace d'adressage et ceux liés aux algorithmes de routage actuellement déployés surchargent les routeurs de l'Internet.

L'acheminement de l'information avec un algorithme scalable et efficient reste un problème récurrent pour tous les réseaux de communication (UMTS, 802.15.4, 802.11, Ethernet, Fibre optique) atteignant une certaine taille. Du fait de la divergence des besoins, il n'existe actuellement pas de solution universelle pour le routage. Néanmoins, quelles que soient les contraintes, un point de convergence existe pour que les algorithmes de routage soient efficients. Efficient signifie ici que l'algorithme choisit toujours le "meilleur" chemin au vu des critères définissant le choix. Ce point de convergence est la nécessité de disposer d'un espace d'adressage et d'une fonction de routage passant à l'échelle. Une fonction de routage passe à l'échelle si les structures de données qu'elle utilise et la puissance de calcul qu'elle requiert augmentent moins que linéairement avec le nombre de nœuds.

1.3.1 Passer à l'échelle

L'élaboration d'un algorithme de routage dans un environnement distribué est idéale si cet algorithme respecte les éléments clefs suivants :

- Le calcul des itinéraires est distribué et le temps de calcul ne dépend pas de la taille du réseau.
- L'étirement des itinéraires est constant, c'est-à-dire qu'il n'est pas modifié par la taille du réseau.
- La taille des tables de routage croît moins vite que linéairement (polylogarithmique) et n'est pas affectée par la taille du réseau.
- Les structures de données utiles au routage sont stockées dans la mémoire locale du nœud.
- L'adresse des nœuds et la taille des entêtes d'un paquet croissent moins que linéairement (polylogarithmique). La taille des adresses définit aussi la taille des entrées dans la table.

Un nombre abondant d'articles traitent de la conception d'une méthode de routage passant à l'échelle. Plusieurs articles du domaine sont [CCN+06, CCK+06] et [GGK+04]. Pour résoudre le problème de taille des tables de routage dans l'Internet l'intérêt s'est porté sur le routage dit compact (RC) [TZ01, AGM+04]. De nombreux modèles émergent pour résoudre de manière plus efficiente le problème du routage dans les graphes. Le routage compact est un domaine de recherche qui étudie les limites fondamentales du passage à l'échelle du routage et conçoit des algorithmes qui tentent de répondre à ces limites. En particulier, la recherche sur le routage compact montre que le routage par le plus court chemin qui forme le noyau des algorithmes de routage traditionnel, ne peut garantir des tailles de table de routage grossissant moins que linéairement en fonction de la taille du réseau sur toutes les topologies réseaux [KCFB07]. Le RC est une alternative qui montre de bonne qualité pour le passage à l'échelle. Cependant, d'après les

travaux présentés dans [KCFB07] aucun algorithme de routage compact ne peut garantir un coût de convergence passant à l'échelle mieux que linaire en fonction de la taille du réseau. Donc, selon Krioukov et al. le RC ne passe pas à l'échelle. En effet, Krioukov et al. dans [KCFB07] démontrent que compte tenu des résultats récents dans la recherche sur le routage compact, un grossissement logarithmique des tables sur des topologies Internet semblable est fondamentalement impossible en présence de topologie dynamique ou d'un adressage à plat indépendant de la topologie. Ils utilisent des arguments analytiques pour montrer que le nombre des messages de contrôle du routage dûs aux changements topologiques ne peut pas évoluer mieux que linéairement sur des topologies Internet semblables. Ils utilisent également des simulations pour confirmer que la limite de taille des tables grossissant logarithmiquement est brisée par un adressage indépendant de la topologie. Or ce modèle d'adressage est le point clef de la proposition LISP visant à améliorer le passage à l'échelle du routage en présence de topologie réseau dynamique ou de mobilité des hôtes. Les conclusions de Krioukov et al. sont pessimistes et ces derniers souhaitent un réexamen fondamental des modèles d'acheminement de l'information afin de trouver une architecture de routage qui serait en mesure de passer à l'échelle indéfiniment. En particulier, il existe des schémas de routage compact conçus pour les grilles, les arbres et les topologies Internet semblables qui offrent des tailles de tables qui croissent de façon logarithmique avec la taille du réseau.

1.3.2 L'itinéraire optimal

Lorsque nous parlons d'allongement ou d'étirement des itinéraires nous faisons référence à un facteur d'élongation obtenu par le rapport entre un itinéraire quelconque et un itinéraire calculé avec une connaissance globale du réseau. Un RR ne reflète pas nécessairement la topologie du réseau sous-jacent d'où le problème d'étirement des chemins lorsque le routage intervient dans un réseau virtuel. Aussi la longueur d'un chemin est définie par le nombre de sauts.

Cependant, dans un réseau P2P on ne souhaite pas nécessairement minimiser le nombre de sauts. Par exemple, pour un réseau P2PTV le délai va être privilégié. Pour le transfert de fichiers volumineux c'est la bande passante qui est primordiale. Pour cette raison des travaux étudient l'influence d'une sélection aléatoire sur les pairs entrants dans le réseau [VF07]. En effet, tout l'enjeu est de réussir à se connecter aux pairs qui vont nous permettre de bénéficier des meilleurs performances. Toutefois, dans l'étude de systèmes P2P réels, les résultats obtenus ne sont pas toujours à la hauteur des espérances. Selon nous cela est dû aux polices de routage et aux partenariats entre opérateurs qui biaisent le routage. Lorsque un pair sélectionne des voisins, il utilise des critères pour leurs choix. Or ces critères peuvent être influencés entre autre par des politiques de répartition de charge.

En lisant, les travaux de Matray et al. [MHL+11] nous observons que des points de convergence émergent au niveau des routes empruntées dans le réseau. Les routes les plus évidentes apparaissent lors de communications Transpacifiques ou Transatlantiques puisque l'information n'a pas d'autre choix que de transiter par un des quelques câbles sous-marins. Ces observations de convergence vers certaines entrées de liens peuvent avoir plusieurs raisons (accord entre fournisseur influençant les polices de routage ou transit

classique par le cœur du réseau devant être plus performant. Ensuite, l'utilisation commune de tronçons de chemin semble évidente du fait de la structure hiérarchique des éléments de l'Internet et ceci indépendamment de toutes polices de routage ou techniques de répartition de charge ou autre subtilités techniques de niveau trois (de la couche OSI) destinées à rendre plus performant le réseau. C'est pourquoi, d'après les constats précédents, nous trouvons pertinent de proposer une méthode de construction de réseau recouvrant indépendante des protocoles supérieurs à la couche 2 afin d'optimiser les échanges de contenus en tirant parti de certains invariants structurels (physiques). Ainsi le réseau n'est pas dépendant d'un modèle architectural (ancien ou à venir) et ne se préoccupe pas des contraintes liées aux administrateurs des éléments de la couche 3. Nous prétendons que le meilleur modèle pour le partage et la diffusion de contenus est le pair-à-pair. Les réseaux pair-à-pair sont idéals. C'est pourquoi dans cette thèse, nous avons voulu construire un réseau recouvrant dans l'Internet permettant d'optimiser les échanges d'informations entre ces participants. De plus, dans des travaux antérieurs [RMK+09] et [ST04], nous pouvons voir ce même constat sur la convergence des itinéraires empruntés par les paquets.

1.3.3 Localiser une destination mobile

Cependant, l'introduction de la mobilité remet en cause les solutions en place. Déjà en 1987, dans [Fin87], Finn soulevait le problème de la mobilité croissante.

> The rapid expansion of cellular mobile telephones suggests that any pervasive public network must efficiently allow host mobility. Mobility is already important to certain business sectors.

C'est intéressant lorsque cela est mis en parallèle avec l'avancement du déploiement du support de la mobilité dans IP décrit par la RFC6275 ou la RFC5944 selon la version du protocole Internet. Une évolution de l'Internet actuelle s'impose car il faut désormais composer avec cette mobilité. En effet, la périphérie d'Internet est de plus en plus mobile du fait de ces utilisateurs.

De ce fait, continuer de localiser un périphérique avec son identifiant devient complexe et nuit à la flexibilité des communications étant donné que l'on identifie le récepteur en fonction de sa position dans le réseau. Autrement dit, lors de changement de position très dynamique d'un périphérique, il y aura régulièrement modification de son "identité" à moins de diffuser une mise à jour dans les éléments intermédiaires. Car le périphérique en bout de chaîne est identifié via son point d'attachement. Alors qu'un découplage de l'espace de nommage avec l'espace d'adressage permet de masquer et de gérer la mobilité. Par exemple, dans le cas de LISP, le mécanisme de liaison proposé est une table de hachage distribuée. Or de nombreux travaux étudient comment créer une table de hachage distribuée performante (e.g. CHORD, CAN, etc.). C'est-à-dire résistante à la volatilité des utilisateurs et avec un temps de transmissions des requêtes le plus faible possible. En effet, les DHT robustes à la mobilité sont des mécanismes complexes à mettre en œuvre. D'autant que le temps de résolution a une importance ainsi que la fraîcheur et la consistance des données. Enfin un tel mécanisme utilise nécessairement une couche de routage. C'est une raison de plus d'aborder le sujet.

1.3.4 Complexité des adresses

Les problèmes qui viennent d'être décrits sont fortement liés à un paramètre : les adresses. En effet, la clef de voûte d'un algorithme de routage est son espace d'adressage ou plus précisément ses adresses car ce sont elles qui encombrent la table de routage. La taille des adresses, définie par le nombre de bits requis pour leur représentation, est aussi un cas d'étude. C'est pourquoi, la taille des adresses doit aussi être limitée ou croître de façon logarithmique car elles sont stockées dans les tables de routage et dans les entêtes de paquets. Par exemple, IP définit le format et la taille des adresses utilisées pour diriger les données dans les routeurs. Dans IP la taille des adresses est fixe : 4 octets pour la v4 et 16 octets pour la v6. Nous pouvons souligner au passage que l'agrégation ne sera pas améliorée avec les adresses IPv6 car plus longue.

Avec un plan d'adressage hiérarchique standard (i.e. en arbre) la longueur des adresses est variable car dépendante du rayon du graphe. En effet, le premier nœud est par exemple marqué 1, son quatrième descendant sera marqué 1.4, le sixième descendant de son descendant sera marqué 1.4.6 et ainsi de suite. Si nous utilisons 1 octet pour chaque niveau de l'arbre standard, nous pouvons voir que le nombre d'octets dans une adresse dépendra de la distance maximale avec la racine. De ce fait, la longueur des adresses est fortement liée à la profondeur du graphe mais ce n'est pas un problème car la structure hiérarchique permet d'agréger les adresses. Cependant, si des liens additionnels sont créés pour raccourcir certains trajets, alors l'agrégation ne sera pas possible puisque les adresses n'auront pas de préfixe commun. Or si ces liens ne sont pas ajoutés les itinéraires sont plus long. Du coup, les paquets consomment plus de ressources réseau du fait du transit plus long. C'est pourquoi, un compromis doit être trouvé. C'est le phénomène qu'a connu IP, mis à part que le grand nombre d'ajouts de liens a eu raison des protocoles en place. Une alternative peut consister à augmenter le degré des nœuds au maximum : avec 1 octet, 256 voisins sont envisageables. Cela augmente la taille des tables mais le diamètre du graphe est plus petit, donc les itinéraires sont aussi plus petits. De plus, l'agrégation reste possible. Néanmoins le réseau perd beaucoup de sa résilience. Or c'est une force de l'Internet. Donc la taille des adresses est un critère à prendre en compte. Nous citerons quelques travaux lorsque nous parlerons du plongement.

Enfin le mode d'adressage impose certains choix de conception. Par exemple, dans un modèle topologique indépendant du nom, l'algorithme ne tient pas compte du format des adresses. De ce fait, une entité centrale au niveau d'un cluster ou au niveau global est requise pour le calcul des itinéraires. Inversement avec un modèle dépendant de la topologie le format des adresses est primordial puisqu'il induit l'itinéraire emprunté.

1.4 De l'adressage au routage

Le routage est le processus d'acheminement de l'information dans un réseau. Au moins un nœud intermédiaire doit être traversé durant le trajet entre la source et la ou les destinations. Au cœur de ce processus se trouve la fonction de routage. C'est le mécanisme par lequel un ou des itinéraires sont sélectionnés dans un réseau pour acheminer les données d'un expéditeur jusqu'à un ou plusieurs destinataires. La sélection d'un itinéraire s'effec-

tue de manière centralisée (i.e. le nœud ou un oracle [1] a une vision globale du réseau) ou décentralisée (i.e. le nœud a une vision locale du réseau). Lors du routage décentralisé, la fonction de routage d'un nœud intermédiaire choisit, dans sa table de routage, un nœud parmi ses voisins auquel le message est transféré. Autrement dit, l'itinéraire est construit par la concaténation des choix locaux. Ce choix est produit de sorte à rapprocher le message de sa destination finale mais le choix du voisin peut être influencé par un ensemble de contraintes (e.g. délai, congestion, partenariat entre opérateurs, etc.).

Le routage est un service implanté sur l'ensemble des nœuds qui forment le réseau. Par exemple, le réseau d'interconnexion Internet, utilise des protocoles et des algorithmes qui sont testés et éprouvés dans le seul but de transmettre de l'information. De ce fait, Internet sert de support aux communications. Dans Internet les nœuds qui implantent le routage et permettent l'interconnexion des réseaux sont les routeurs. Ce sont des équipements utilisant la couche de protocole 3 du modèle OSI dans laquelle est implanté le protocole Internet IP (de l'anglais *Internet Protocol*). Un routeur est un élément intermédiaire dans un réseau de communication assurant le transfert des paquets. Les paquets sont transmis d'un équipement vers un autre, selon un ensemble de règles formant la table de routage. Lorsqu'un nœud transmet un message à tous les nœuds auxquels il est connecté alors ce n'est plus du routage mais de l'inondation. Cette technique est, par exemple, utilisée par la couche de protocole 2 du modèle OSI qui malgré tout identifie de manière unique les équipements implantant cette couche avec des adresses MAC (de l'anglais *Media Access Control*). De plus, l'inondation ne passe pas à l'échelle car l'augmentation du nombre des messages sature les ressources du réseau et ne permet plus une bonne gestion des collisions. Aussi, la sécurité des transmissions peut-être remise en cause puisque toutes les machines connectées reçoivent les messages et décident de leur filtrage. Nous n'irons pas plus loin ou plus haut dans la description des couches qui forment l'architecture de l'Internet actuel [Tan04].

Les algorithmes de routage existants permettent d'échanger des messages selon un des trois paradigmes de communications suivants :

- De l'un vers un autre (point-à-point, unicast, one-to-one).
- De l'un vers plusieurs (point-à-multipoint, multicast, one-to-many) ou tous (broadcast, one-to-all)
- De plusieurs vers plusieurs (multipoint-à-multipoint, multicast, many-to-many)

Dans cette section, nous attaquons le sujet central de ce mémoire, c'est à dire le routage point-à-point dans les réseaux de communications plus ou moins dynamique.

Dans ce mémoire, nous abordons précisément un type de solution le routage basé sur la position des nœuds (de l'anglais *position-based routing*). En effet, la démocratisation des technologies permettant de localiser géographiquement des périphériques a ouvert le champ des possibilités du routage avec une nouvelle forme de routage dite géométrique. L'idée de transmettre des messages en utilisant les positions des nœuds dans l'espace a au moins été proposée depuis les années 1980 avec Hideaki Takagi [TK84] dans les réseaux radios puis avec Gregory Finn [Fin87] pour les réseaux filaires. Pour réaliser un tel routage, la première intuition, provenant des réseaux *ad hoc*, est d'affecter aux nœuds des coordonnées basées sur leur espace physique sous-jacent (e.g. euclidien). L'espace

1. L'oracle calcule l'itinéraire en une opération

des coordonnées est dit réel. Pour ce faire, il est nécessaire de disposer d'équipements ayant accès à un service de positionnement comme le *Global Positionning System* (GPS). L'espace implicite est alors la sphère pour un réseau étendu sur une planète. Pour un réseau à l'échelle de la France le plan euclidien peut suffire.

1.4.1 Des coordonnées comme adresse

Néanmoins le procédé de GPS, n'est pas adapté à tous les réseaux car jusqu'à récemment la plupart des périphériques, nomades ou non, ne disposaient pas de puce GPS. C'est pourquoi une dernière méthode utilisant des coordonnées virtuelles a émergé.

Un autre processus consiste à déterminer/approximer la position réelle des nœuds les un par rapport aux autres à l'aide de mesure et de sonde dans le réseau. Déterminer la position nécessite de calculer une distance, du coup la distance dépend de la métrique considérée souvent en rapport avec des caractéristiques du réseau (délai, bande passante, nombre de sauts, congestion). Les nœuds sont alors placés dans un espace de coordonnées qui permet d'évaluer la distance avec la fonction distance de l'espace considéré. Aucune mesure supplémentaire n'est requise. L'estimation des distances dans l'Internet est étudiée depuis plusieurs années. L'idée est de déterminer si deux nœuds sont proches dans le réseau. Encore faut-il définir la notion de proximité.

De nombreux algorithmes conçu pour l'estimation de distances dans l'Internet prennent l'hypothèse que la proximité est définie par le délai d'aller-retour d'un paquet (RTT). Certes, cette caractéristique est importante puisqu'elle représente le temps de transit d'un paquet entre deux équipements du réseau. Les services d'estimation de délai [DGK10] (IDMaps, GNP) ou Vivaldi et Sequoia [RMK+09], essayent d'estimer de la manière la plus précise possible les délais entre deux nœuds distants en les plongeant dans un espace de coordonnées virtuel. Le but est qu'il y ait une correspondance des distances entre les nœuds dans l'espace et le délai inter-nœuds. Néanmoins, toutes ces solutions ont été mises en défaut par le non respect de l'inégalité triangulaire [LBSB09, ZCZ+10]. Cependant des travaux tel que ceux de [FLV08] modélise l'espace des délais avec une infra métrique. D'autres travaux comme l'article [AK08] essayent de caractériser la dimension de l'espace correspondant aux délais. En outre, d'autres caractéristiques peuvent être utilisées, c'est d'ailleurs pour cette raison que bon nombre d'algorithmes sont élaborés pour la construction de RR ou pour la conception de schéma d'adressage utile au routage et au routage par la clef présent dans les DHT. Enfin, Shavitt et Tankel s'attaquent aussi à la caractérisation de l'espace des délais mais ils le font au travers de l'espace hyperbolique [ST08]. Ils tentent de déterminer une corrélation entre la courbure du plan hyperbolique et la courbure des chemins suivit dans IP. La courbure des chemins IP signifie que les messages traversent les différentes hiérarchies matérielles avant de redescendre par le même type de hiérarchie vers l'utilisateur final.

Malheureusement le ou les critères choisis ne forment pas nécessairement un espace métrique et à notre connaissance aucune des grandeurs spécifiques aux réseaux existants ne forme un espace métrique (e.g. délai, saut, RTT, etc.) mais au mieux l'espace métrique est relâché (i.e. l'inégalité triangulaire n'est pas strictement respectée). C'est pourquoi, nous choisissons l'approche qui consiste à plonger un graphe dans un espace mathématique. En effet, un plongement bien choisi offre la *fonction de calcul de distance* nécessaire

pour pouvoir déterminer la proximité entre deux nœuds quelconques de l'espace. À cette fin, il est primordial de choisir un espace cible et la *fonction distance* adéquate. De ce fait, les espaces mathématiques sont attractifs grâce aux propriétés qu'ils offrent. Cette méthode n'est pas obligatoirement utilisée pour le routage mais elle peut l'être. De plus, elle permet une certaine corrélation entre l'espace d'adressage et la topologie sous-jacente. En fait, cela permet d'extraire une topologie virtuelle ou un sous graphe du réseau.

La méthode que nous présentons maintenant a pour but de d'abord extraire un sous graphe du réseau en ne tenant compte que du graphe de connexion virtuel des nœuds. Nous décrivons comment calculer de manière distribuée dans le réseau un plongement géométrique spécial du graphe. Ce plongement supporte un protocole de routage géométrique appelé "routage glouton" qui est basé sur les coordonnées dites virtuelles des nœuds plongés.

Un algorithme glouton est un algorithme qui, confronté à un choix, choisit ce qui lui semble le meilleur pour avancer. C'est un choix local, et on espère que la succession de choix locaux va amener à une bonne solution. Le **plongement glouton** d'un graphe $G = (V, E)$ non orienté et non pondéré dans un espace métrique (X, d) est une fonction $f : V(G) \to X$ munie de la propriété suivante : pour toutes paires distinctes de sommet $s, t \in V(G)$ il existe un sommet u adjacent à s tel que $d(f(u), f(t)) < d(f(s), f(t))$. Par exemple, Westphal and Pei présentent dans [WP09] un schéma d'adressage permettant un routage scalable. Les adresses sont obtenues par un plongement glouton sur un espace de dimension $O(log(n))$ avec pour chaque nœud des tables de routage de taille polylogarithmique. Avec le même objectif Flury et al. propose dans [FPW09] le premier algorithme en temps polynomial qui plonge les graphes de disque unité dans les espaces de $O(log_2(n))$ dimensions. De plus, leur algorithme de plongement glouton assure un étirement borné, bien qu'il utilise des étiquettes plus longues pour les nœuds.

La plus simple des techniques de routage basée sur des coordonnées géographiques est gloutonne, dans le sens où les nœuds transfèrent toujours le paquet au voisin qui est le plus proche de la destination finale en utilisant, par exemple, la métrique euclidienne [LPMS09, SCIE10].

Un schéma de routage géométrique simple est un routage glouton. Avec le routage glouton, lorsqu'un nœud reçoit un paquet, il transmet le paquet vers le voisin le plus proche dans un certain sens pour le nœud destination. Ce choix peut être dû à la structure de l'adressage ou à un ensemble de règles ou être le résultat de la mesure d'une grandeur signifiant la proximité entre les nœuds ; en d'autres termes, c'est la métrique telle que décrite précédemment. Le principal problème avec le routage glouton, c'est qu'il peut rencontrer des minima locaux, également connus sous le nom de routage autour des vides (*voids*) ou des trous (*holes*), lorsque le nœud courant n'a aucun voisin de plus proche de la destination que lui-même. Quand un tel minimum local est rencontré, le paquet est « coincé », le routage glouton ne peut plus continuer, et l'exécution échoue. Un exemple de routage glouton est le routage glouton euclidien, qui est basée sur la distance euclidienne avec la destination, ou le routage à la boussole (*compass routing*), basé sur une distance angulaire avec la destination [KSU99]. Une question importante est la conception de mesures de proximité (c'est-à-dire la distance) qui garantissent la livraison de tous les paquets, indépendamment du nœud source ou destination. Étant donné que cette mesure est habituellement une distance dans un espace où les nœuds ont été plongés, le problème

de positionnement des nœuds dans un tel espace est attribué au problème du calcul d'un plongement glouton pour un graphe donné.

L'exemple le plus naturel de routage glouton est le routage glouton euclidien où la proximité des nœuds est mesurée par la distance euclidienne. Ce scénario a été étudié en détail par Papadimitriou et Ratajczak [PR05], qui ont conjecturé que tout graphe planaire 3-connecté admet un plongement glouton euclidien, c'est-à-dire, un plongement glouton pour la distance euclidienne. Comme ils ne prouvent pas leur conjecture, Papadimitriou et Ratajczak proposent d'autres schémas de routage glouton, plus particulièrement, le routage 3D polyédrique. Cela consiste à plonger le graphe planaire 3-connecté dans un polyèdre aux arêtes tangentes à la sphère unité dans $\mathbb{R}^3$. Un paquet est acheminé par son transfère au sommet voisin qui maximise le produit scalaire avec le sommet de destination. Un tel plongement existe toujours, et ils prouvent que le schéma de routage fonctionne toujours à condition d'avoir un graphe 3-connectés.

La notion de plongement glouton est justement définie par Papadimitriou et Ratajczak [PR05] mais c'est J. Kleinberg qui a permis de la faire connaître avec *Nature* [Kle00]. En effet, de nombreux travaux traitent des schémas de routage glouton [LW06, NMF$^+$07, TAC08, SCIE10]. Le routage glouton comme algorithme de routage distribué efficient a alors attiré toute la communauté réseau. Le plongement est motivé par les applications qu'il trouve dans le routage de l'information où l'idée est de n'utiliser que les informations locales. Tout plongement f définit un unique chemin dans G entre les paires de sommets distinctes (s, t) lequel est le chemin à emprunter par un message allant de s à t et inversement d'où une symétrie du chemin. Cette définition implique que router de manière gloutonne dans G avec la métrique d réussit toujours. En effet, un algorithme de routage induit par un plongement glouton donné fonctionne comme suit. Pour délivrer un message depuis s à t avec $s, t \in V$, l'algorithme fait suivre récursivement le message au voisin dont la distance de plongement est minimale dans l'espace cible (i.e. $d(f(u), f(t))$). Autrement, le routage stoppe ce qui signifie que la cible est atteinte. Donc, si le plongement est glouton alors le message est garanti d'atteindre sa destination. Les techniques les plus simples de routage géographiques sont gloutonnes, dans le sens où les noeuds transfèrent toujours le message au voisin qui est le plus proche de la destination en utilisant par exemple la métrique euclidienne [XLPH06, LLM07, LPMS09].

Les minima locaux

Cependant, cette efficacité peut être stoppée quand il existe un nœud qui est plus proche de la destination que tous ses voisins : le routage atteint un minimum local et échoue à atteindre la destination. Ce nœud est appelé un minimum local car les paquets qui l'atteignent ne peuvent pas être dirigés vers leur destination.

Comme toujours plusieurs solutions sont décrites pour surmonter le problème des minima locaux dont le premier objectif est de favoriser une reprise rapide du routage alternatif vers le mode glouton. Les premières propositions de routage géographique étaient de simples schémas de transfert glouton qui ne garantissaient pas la livraison des paquets dans un réseau connecté [TK84]. Les paquets sont simplement supprimés lorsque le transfert glouton provoque leur transfert dans un minimum local. Le premier algorithme de routage géographique (ou géométrique) à offrir la livraison garantie est le routage sur les

faces [KSU99] (initialement appelé "compass routing" II). Plusieurs algorithmes pratiques qui sont des variantes du routage sur les faces ont été développés depuis, incluant GFG, GPSR et la famille des algorithmes GOAFR+ [KWZ03, KWZZ03]. Le dernier ajout en date est GPVFR qui améliore l'efficacité du routage en exploitant les informations locales aux faces. GOAFR+ est asymptotiquement optimal et limite la chute des performances dans le pire cas avec une recherche en ellipse croissante, GPVFR atteint le meilleur résultat dans le cas moyen pour l'étirement parmi les algorithmes de routage géographique sur les faces.

Les auteurs de [LW06] proposent une solution basée sur le transfert sur les faces. Ils partitionnent le réseau en *k régions de destinations* et présument que chaque nœud a, a priori, connaissance de sa région. Ils utilisent un mécanisme de prédiction, une sorte de longue de vue, pour anticiper les minima locaux potentiels sur le trajet. En quelque sorte les nœuds peuvent être marqués comme ML lorsque plusieurs nœuds les considèrent comme tel. Malgré le mécanisme de prédiction de Liu and Wu [LW06] les performances restent peu attractives. Effectivement, comme expliqué dans beaucoup d'articles dont [DC09], le maintien d'une structure de graphe planaire est coûteux en ressources du fait du trafic généré pour la coordination des nœuds. Dans [DC09], les auteurs proposent une nouvelle technique de routage géographique sensible au niveau énergétique des nœuds. Ils définissent donc une métrique prenant en compte l'énergie. La proposition combine un routage glouton sensible à l'énergie et une marche aléatoire biaisée pour surmonter les ML. En plus, Ils définissent pour un paquet en mode alternatif un nombre maximal de saut avec un paramètre *Hops To live* équivalent au champ IP bien connu *Time To Live*. Lorsque le nombre de sauts est atteint le paquet est détruit. Malheureusement, le taux de délivrance du schéma de routage n'est pas de 100%

Effectivement, notre méthode se démarque par sa simplicité qui réside dans les équations qui la composent. De ce fait, la difficulté est en amont avec les équations et non pas lors de l'implémentation.

1.4.2 Mieux acheminer l'information

Les algorithmes de planarisation qui étaient initialement disponibles se fondent sur l'hypothèse que le réseau sous-jacent est un graphique de disque unité (UDG) pour la connexité et étaient inutilisables dans des réseaux pratiques. Une percée a été faite par Kim et al. dans le développement du Cross-Link Detection Protocol (CLDP) [jKGkS05a] qui produit un sous-graphe dans lequel les algorithmes basés sur le routage sur les faces sont garantis de fonctionner correctement. Tous proposent des algorithmes de routage géographiques qui sont basés sur le routage avec les faces [KSU99], qui garantissent la livraison de paquets par le routage sur un sous-graphe planaire du réseau. Il s'avère que la planarisation distribuée est difficile pour les réseaux réels sans fil [Kar01] et le problème n'a été résolu seulement récemment par Kim et al. dans [jKGkS05a], avec le protocole de détection de croisement de liens (CLDP). Cependant, CLDP est complexe et coûteux alors que le routage sur les faces nécessite la manipulation de nombreux cas de minima locaux [jKGKS05b]. Le coût de maintenance et la complexité élevée associés avec le déploiement d'un algorithme de routage par les faces tel que CLDP incite à explorer des solutions alternatives.

Les auteurs de [RRP+03] proposent un algorithme de routage qui attribue des coordonnées virtuelles dans le plan euclidien et transfère des messages en utilisant le routage glouton. Aussi, dans [RRP+03] les coordonnées virtuelles sont assignées à l'aide d'une version distribuée de l'algorithme *rubber band* de Tutte afin que le graphe plongé soit planaire. W.T. Tutte a prouvé la propriété non triviale suivante :

> nail the nodes of any face of a 3-connected planar graph to a convex polygon, and replace the edges by rubber bands. The equilibrium provides an embedding in the plane [Tut63].

Donc plonger un graphe dans un espace de coordonnées virtuel requiert de fixer un espace et un sous graphe. Plus récemment, Leighton et Moitra [AT08] prouvent que tous les graphes planaires trois-connectés admettent un plongement glouton dans le plan euclidien. Cette preuve répond à la conjecture de [PR05]. Dhandapani a aussi participé à la résolution de la conjecture de Papadimitriou et Ratajczak pour le cas spécial d'un graphe de triangle. À l'aide du plongement de Schnyder de triangulation de la sphère, Dhandapani a montré l'existence d'un plongement glouton euclidien pour toutes ces triangulations. Malheureusement, la preuve n'est apparemment pas constructible.

Ensuite, le problème de certaines méthodes comme l'algorithme de [WP09], appelé GLoVE, crée des minima locaux lors du plongement des nœuds dans le plan euclidien. Durant la construction du plongement, ils garantissent que le plongement est glouton entre chaque nœud et la racine de l'arbre. Mais le plongement réalisé par GLoVE ne permet pas de trouver tous les chemins sources destinations. Donc, si un message est dans un minimum local, il n'est pas sur le chemin entre la racine et la destination où le routage glouton est assuré par leur construction. Pour surmonter les minima locaux, ils proposent une variante de GLoVE, notée GLoVE-U. Le nœud transfère simplement le message à ses parents dans l'arbre. Ils nomment cette méthode *up-tracking*. Le message remonte dans l'arbre. Nous n'avons pas exploré plus leur méthode car elle crée des minima locaux.

Dans [May06], Maymounkov fournit une méthode de plongement glouton pour l'espace hyperbolique à 3 dimensions. Ce plongement permet de représenter les sommets avec $O(log^2(n))$ bits. Son objectif est d'apporter une solution au problème de la complexité des adresses. C'est-à-dire le nombre de bits nécessaire pour identifier les sommets. Dans le même contexte, les auteurs de [Goo08] décrivent une méthode pour plonger n'importe quel graphe simple à n sommets dans le plan hyperbolique. Ils montrent qu'avec leur méthode tout graphe peut être dessiné dans le plan hyperbolique avec des coordonnées utilisant une représentation en $O(log(n))$ bits. Leur technique est basée sur la combinaison de plusieurs méthodes de dessin (*graph drawing*) et de structures de données pour créer : *weight-balanced binary trees, heavy path decompositions et dyadic tree*. En effet, nous trouvons des similitudes entre plonger et dessiner un graphe dans l'espace. Cependant, ces deux propositions sont complexes à mettre en œuvre dans un contexte distribué, cet aspect n'est ni traité ni évalué dans leur article. Le nombre de bits requis pour la représentation des coordonnées des sommets du graphe est leur force mais leur méthode est difficilement implémentable. De plus, nous utilisons des adresses de longueur constante. Donc, pour plonger des nœuds dans un espace de coordonnées virtuels, la première chose est de définir un espace et un sous graphe.

D'autres espaces ont été considérés comme espace de plongement pour le problème du routage glouton. R. Kleinberg a étudié la question du plongement d'un réseau dans

le plan hyperbolique et a été capable de construire un plongement glouton dans le plan hyperbolique pour tout graphe fini connecté. Notez que le graphe n'est pas supposé être planaire, ni avoir une propriété particulière de connectivité, au-delà de la connexité.

1.4.3 Routage géométrique hyperbolique

À notre connaissance, la première apparition du plan hyperbolique pour aborder des problématiques liées à l'Internet est dans le travail de Shavitt et Tankel [ST04, ST08]. Dans ce travail, les auteurs considèrent un heuristique de plongement pour le plan hyperbolique. Ils évaluent différentes courbures de l'espace hyperbolique afin de trouver celui qui décrit mieux le comportement des distances Internet. Ils montrent que la courbure négative de l'espace cible améliore la précision et l'efficacité du réseau de recouvrement (RR) construit ainsi que l'estimation des distances en termes de délai.

Dans notre approche nous sélectionnons le plan hyperbolique et plongeons un graphe sans traitement préliminaire sur le graphe de départ. En effet, notre algorithme nous permet de définir des coordonnées précises calculables localement par chaque nouveau nœud et préservant la propriété du plongement glouton.

Nous introduisons les travaux directement liés à notre solution, c'est-à-dire traitant du plongement des graphes dans le plan hyperbolique [Kle07, KPBV09, CC09]. L'article de R. Kleinberg [Kle07] sur l'utilisation du plan hyperbolique est incontournable. Il contribue avec le théorème suivant : Tout graphe finit connecté a un plongement glouton dans le plan hyperbolique. En effet, il propose une technique pour plonger de manière gloutonne n'importe quel graphe dans le plan hyperbolique. Néanmoins, son algorithme nécessite d'une part d'extraire un arbre du graphe et d'autre part de déterminer le degré maximal de l'arbre avant d'effectuer le plongement. Ainsi, pour connaître le degré maximal du graphe, au moins un parcours global est obligatoire. Dans cette partie, nous détaillons l'approche de Kleinberg et proposons un algorithme distribué ne nécessitant aucune opération supplémentaire sur le graphe sous-jacent. Le point commun des travaux de [Kle07] et [KCFB07] (routage compact) est que les deux requièrent une connaissance globale du graphe.

Ensuite, les auteurs de [CC09] proposent une méthode heuristique, appelée *Gravity-Pressure*, permettant de surmonter le problème du minimum local. Lorsque le message est dans un minimum local, le nœud utilise l'heuristique pour router le message. Le message stocke alors une liste des nœuds traversés pour s'éloigner du minimum local afin de reprendre un routage glouton mais cela est coûteux pour la taille du message. Leur algorithme n'est évalué qu'avec 50 nœuds car le maintien de la liste des nœuds traversés pour s'éloigner du minimum local, avant de reprendre un routage glouton, est coûteux pour la taille du message car la liste est contenue dans le message. Puis, Les auteurs proposent un algorithme distribué de plongement glouton. Néanmoins, leur solution nécessite un algorithme distribué de construction d'arbre couvrant et un algorithme d'élection pour la racine. Ils affirment qu'avec leur algorithme dans le cas d'un graphe dynamique, le degré d'un nœud n'est pas limité donc que le nombre de régions combinatoires libres n'est pas limité. Or cela n'est pas vrai en pratique, c'est pourquoi, nous fixons le nombre d'adresses à q_0. En effet, les sous régions créées avec leur méthode doivent rester tangentes, c'est essentiel afin de conserver un algorithme de plongement glouton. Contrairement à un dé-

coupage homogène du plan tel que Kleinberg ou nous le faisons, ils utilisent un placement dichotomique, i.e. à chaque fois ils découpent la moitié de la moitié restante du plan pour placer les nœuds. Lorsque les premiers nœuds plongés ont un degré élevé d, l'espace de départ est sectionné à moitié, puis l'espace restant encore à moitié et ainsi de suite pour positionner les d nœuds correspondant. Effectivement, cela permet de conserver quelques régions libres et les sous espaces tangents deux à deux mais du fait de la dichotomie, le $d$ème nœuds possèdent un sous espace proche du cercle limite. Dés le début, cela a pour effet de positionner les nœuds de l'arbre proches du cercle limite. Donc si le graphe est profond les nœuds ne pourront pas obtenir d'adresse à moins de disposer d'une précision de calcul infiniment grande. Donc la conséquence immédiate est une perte de la capacité d'adressage potentiellement importante mais cette capacité n'est pas évaluée dans l'article. De notre côté, nous évaluons cette capacité et montrons cette limitation dans [CTBM11].

Krioukov and al. [KPBV09] présentent une autre approche utilisant l'espace hyperbolique. Ils génèrent un graphe dans lequel les nœuds sont uniformément distribués dans le disque de Poincaré et connectés en fonction d'une probabilité dépendant de la distance entre les nœuds. Puis, ils montrent qu'un graphe avec une distribution du degré en loi de puissance émerge naturellement. Ensuite, ils affirment que le graphe qu'ils génèrent est congruent à l'espace métrique que forme le disque de Poincaré. Pour le prouver ils utilisent une stratégie de transfère glouton qui peut être efficace seulement s'il n'y a pas de minima local. Or dans leurs travaux, le taux de délivrance du routage glouton est élevé mais pas de 100% ; preuve que la distribution du degré des nœuds donne une topologie de graphe moins congruente que celle construite avec la une position des nœuds optimale dans l'espace [Kle07]. L'heuristique *Gravity-Pressure* est aussi utilisée dans [PKBV10]. Dans [Kle07], on part d'une topologie de graphe entièrement connu et on la plonge pour bénéficier du routage glouton. À l'inverse dans [KPBV09], le problème est en fixant l'espace hyperbolique, vérifier si il existe une procédure de construction de graphe simple menant à une topologie scale-free. Est ce que le routage glouton est efficient sur ce type de topologie.

Toujours dans [CTBM11] nous évaluons l'étirement mais sur un sous graphe construit avec un parcours en largeur à degré fixe q sur des cartes d'Internet (e.g. BGP, Ipv4, Ipv6). Nous trouvons un étirement proche de 2.

1.5 Synthèse

> De l'Internet aux réseaux *ad hoc*, en passant par les réseaux de pair à pair, la façon dont les messages sont acheminés constitue le problème central des réseaux [Laurent Viennot].

Le routage est un problème central dans la conception de n'importe quel réseau d'interconnexion de grande taille [Fin87]. En effet, les réseaux de type *ad hoc* ou mobile *ad hoc*, qu'ils s'agrègent ou non à l'Internet avec le "tout IP", connaissent de part leur croissance les mêmes problèmes mais avec des contraintes telles que la mobilité ou la consommation énergétique encore plus forte. Dans un réseau *ad hoc*, les nœuds mobiles communiquent entre eux directement, plutôt que par le biais d'infrastructures fixes tels que les antennes

de téléphonie mobile ou les stations de base. En ne s'appuyant sur aucune infrastructure fixe, les réseaux *ad hoc* peuvent être déployés rapidement et spontanément. De ce fait, Ils sont plus flexibles que les réseaux filaires ou sans fil typiques et sont plus aptes à répondre aux besoins imprévus (catastrophe). Cependant, du fait des contraintes, le potentiel de ces réseaux a du mal à être pleinement exploité [MCK11]. De plus, avec la démocratisation des appareils nomades, les applications P2P risquent à l'avenir de se déporter sur les réseaux MANet. Le routage dans les réseaux sans fil est une tâche ardue. En effet, la propriété fondamentale des réseaux *ad hoc* est l'imprévisibilité des changements structurels qui sont beaucoup moins présent dans les réseaux filaires. Toutefois, nous retrouvons ces mêmes changement au sein des réseaux P2P. Cette imprévisibilité rend alors difficile l'élaboration de méthode de communication pour le transport de données d'un émetteur à un récepteur. De telles méthodes peuvent être qualifiées de stratégies de communication. De nombreuses approches ont été proposées dans la littérature. Nous pouvons identifier trois grandes catégories de protocoles de routage : (i) proactif comme OLSR , (ii) réactif comme AODV et (iii) géométrique ou georouting. Cette dernière approche reçoit de plus en plus d'attention car elle utilise peu de mémoire et passe à l'échelle, contrairement aux deux autres. En outre, cela convient mieux aux contraintes des réseaux *ad hoc* où la mémoire et la capacités de traitement des périphériques sont très faibles et dont le nombre de nœuds est potentiellement très élevé [MRSR10]. Cependant réaliser un routage géométrique nécessite un espace d'adressage ou espace de coordonnées adapté. De plus, la section sur le nommage nous a permis de rappeler que l'espace d'adressage ne peut pas se suffire à lui-même pour assurer une bonne flexibilité des communications. Tous ces ingrédients permettent de répondre aux problèmes soulevés dans la section 1.3.

Pour finir, l'étude récente et intéressante de [MCK11] tente de définir un cadre pour choisir, en fonction de la dynamique de la topologie, la stratégie de communication adaptée. Une variété de stratégies de communication existe. Malheureusement, il n'y a aucune recommandation précise pour choisir parmi les différentes stratégies qui englobent une large catégorie de réseaux *ad hoc* : acheminement point à point, transmission épidémique / inondation, transmission pour réseau tolérant (DTN). Évidemment, la stratégie va dépendre de l'utilisation finale du réseau. Ce travail étudie au travers de traces réseaux un cadre permettant de gérer la stratégie de communication appropriée en fonction de l'évolution du réseau.

Chapitre 2

Adressage et Routage dans le disque de Poincaré

Sommaire

Dans ce chapitre, nous abordons la problématique du routage point-à-point issu d'un plongement glouton dans un contexte distribué, dynamique et hétérogène. Ces trois mots caractérisent l'environnement dans lequel nous travaillons. En effet, ces contraintes sont inhérentes aux appareils nomades. Or la périphérie d'Internet en est de plus en plus composés. En outre, nous retrouvons ces mêmes contraintes mais à un niveau plus élevé dans la pile des protocoles de communication pour le déploiement d'application P2P. De ce fait, nous parlerons dans ce chapitre, sans perte de généralité, de pair plutôt que de nœud. Le défi consiste à élaborer, pour une topologie quelconque dynamique, un algorithme de routage glouton distribué capable d'acheminer les messages au travers d'un réseau d'équipement aux ressources hétérogènes en utilisant uniquement les informations locales aux pairs traversés. Les informations locales à un pair englobent celles de ses voisins directs et celle du message en transit. L'hétérogénéité des équipements est représentée

par la distribution du degré des pairs qui ne suit pas de loi de distribution particulière, nous précisons ce point dans la section 2.1 ainsi que les différents protocoles internes à notre système. Car notre schéma de routage est évalué au travers d'un RR P2P avec lequel nous proposons des mécanismes pour faire face aux défaillances des pairs engendrant les défaillances du routage. Le routage glouton utilise des coordonnées virtuelles prises dans le plan hyperbolique.

Au chapitre 1, nous avons expliqué comment nous plongeons un graphe dans un espace métrique. L'espace considéré pour le plongement est le disque de Poincaré, présenté dans le même chapitre. Dans l'état de l'art nous avons vu l'intérêt d'une telle solution. En effet, nous avons introduit les travaux de R. Kleinberg qui, pour une topologie statique, propose un algorithme de routage glouton point-à-point destiné aux réseaux de capteurs ou plus largement aux réseaux sans-fil *ad hoc* mais de petite taille (i.e. cinquantaine de nœuds). Dans ce chapitre, nous présentons les modifications et améliorations que nous apportons à la solution de Kleinberg pour fournir un algorithme de routage point-à-point résilient aux environnements dynamiques et passant à l'échelle. Notre algorithme de plongement glouton est entièrement basé sur l'algorithme de pavage de la section 1.1 du chapitre 1. Nous utilisons l'algorithme pour étiqueter les nœuds d'un réseau virtuel afin de router des messages à l'aide des étiquettes. C'est un algorithme qui s'exécute de manière distribuée et asynchrone avec des informations locales.

Dans un premier temps, nous expliquons la mécanique interne de notre système P2P. Puis, dans un second temps nous évaluons notre schéma de routage sur des topologies statiques section 2.2. Ensuite, sur des topologies dynamiques section 2.3. Puis, dans le cas d'un réseau dynamique, nous proposons des solutions pour surmonter les minima locaux, comme expliqué au chapitre 1, afin de maintenir le routage glouton. Enfin, nous discutons aussi des méthodes capables de restaurer un arbre d'adressage brisé. Enfin, dans la section 2.4, nous discutons des limites pratiques auxquelles nous sommes confrontés pour numériser le modèle mathématique.

2.1 Comportement des pairs

Dans cette section nous détaillons les protocoles qui permettent de déployer notre réseau P2P non structuré. Tout d'abord, précisons un point clef, le problème bien connu du démarrage est rappelé au chapitre 3. Cet état que nous qualifions de démarrage permet aux pairs d'obtenir des adresses de contact. Ainsi un pair entrant peut émettre des requêtes à destination de ces adresses qui représentent de potentiels points d'attachements au réseau P2P. Lors du processus de démarrage le pair sollicite un sous-ensemble de pairs afin d'obtenir une coordonnée hyperbolique. Dans cette section, les adresses font références aux adresses de la couche transport (i.e. adresse IP) alors que les coordonnées (hyperboliques) font références aux adresses internes au réseau P2P.

2.1.1 Protocole de démarrage

Notre protocole se déroule de la manière suivante, un pair inactif tente de joindre le réseau. Pour cela il doit obtenir au moins une coordonnée via un de ses futurs voisins au

réseau P2P. Le pair entrant passera à l'état actif simplement si une des coordonnées reçues par les pairs actifs est valide. En attendant, l'état intermédiaire est celui du démarrage. Le pair à l'origine de la coordonnée valide devient alors le point d'attachement et les autres pairs ayant répondus deviennent de simples voisins. Si les pairs actifs ont déjà donné toutes leurs adresses, le pair entrant doit trouver d'autres adresses de contacts via le service adéquat (e.g. DNS). En effet, comme la connaissance globale du RR n'est pas nécessaire, un mécanisme donnant accès aux adresses de contact est requis. Pour nos simulations, nous utilisons un mécanisme interne qui centralise les requêtes tel un DNS. Une autre méthode possible est de diriger chaque nouveau pair entrant vers la racine qui si elle n'a pas d'adresse diffusera la requête à ses descendants. De même si les descendants n'ont pas d'adresse, ils transmettront à leurs descendants. Cela assure de parcourir l'ensemble du réseau, néanmoins le nombre de requêtes peut augmenter de façon exponentielle. En effet, les feuilles de l'arbre hyperbolique augmentent de manière exponentielle donc le pair entrant pourrait se retrouver en déni de service. De plus, il conviendrait de s'assurer par un processus de certification que les adresses proviennent bien du réseau que le pair souhaite intégrer mais le problème reste vrai avec un DNS. Concernant les échanges de messages notre système est asynchrone, pour cette raison un pair n'attend pas de recevoir toutes les réponses (adresses disponibles) aux requêtes émises, quelques une doivent suffire. C'est pourquoi, nous préférons utiliser un service de type DNS qui gère la structure de données permettant aux pairs entrant d'obtenir une liste valide de pair actif.

2.1.2 Protocole de message

Nous décrivons sur la figure 2.1 le processus qui déclenche le protocole d'échange de messages permettant d'intégrer un nouveau pair au réseau P2P. Dés l'instant qu'un pair entrant lance la procédure pour intégrer le réseau, plusieurs événements peuvent se produire et interférer avec le bon déroulement des opérations. Nous avons identifié deux problèmes majeurs.

La première zone critique que nous avons identifiée commence à partir du moment où une requête de type réponse, c'est-à-dire contenant la coordonnée hyperbolique, est émise par un pair actif. Autrement dit, un pair actif reçoit une requête JOIN, état d'entrée jaune sur la figure 2.1, à laquelle il répond. Le début de la zone critique est identifié par un marqueur rouge sur la figure. La zone est identifiée entre les marqueurs rouges $M1$ et $M2$. Cette zone est critique car le pair actif bloque une adresse jusqu'à réception d'une requête de confirmation émise par le pair entrant, état d'entrée jaune sur la figure 2.1, donc si le pair entrant disparaît ou sort du réseau, pour une raison quelconque, durant le processus d'intégration alors l'adresse est perdue durant un certain temps. Cette zone se termine lorsque le pair entrant émet la requête de confirmation pour l'ajout dans son voisinage, que ce soit comme point d'attachement ou comme voisin, et que le pair actif la reçoit, zone aussi identifiée par un marqueur rouge. Bien entendu plusieurs pairs actifs sont concernés en même temps. Une méthode pour remédier au problème est de mettre un minuteur au niveau du pair actif. Car le seul problème des pairs actifs est de ne pas rester avec des adresses bloquées inutilement. D'ailleurs lorsque des pairs actifs constatent la déconnexion d'un de leurs descendants (aussi pairs actifs) alors ils libèrent l'adresse correspondante.

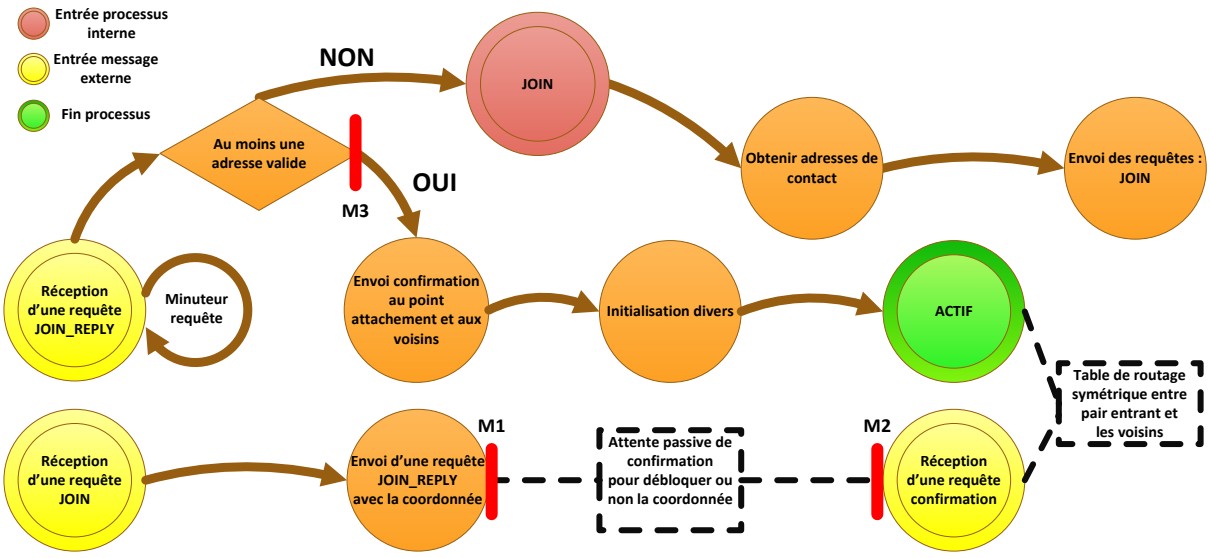

FIGURE 2.1 – Protocole d'intégration au réseau P2P

Ensuite un second problème se pose si un pair actif lance une procédure de maintenance dans laquelle il diffuse un message à son voisinage afin de se déconnecter. En effet, l'adresse hyperbolique est bloquée mais le pair entrant n'a pas encore été ajouté à la table de routage du pair actif. Donc le pair entrant n'est pas informé des intentions du pair actif. Or si le pair entrant a sélectionné ce pair actif comme point d'attachement cela pose un problème car le pair entrant devenu actif, du fait de son point d'attachement, découvrira le problème après 2 fois l'intervalle de temps spécifié pour la mise à jour des tables de routage. La raison est que tout pair entrant dans le réseau contient tous ses voisins durant le premier intervalle de temps des mises à jour. Ce problème n'apparaît que si le pair actif disparaît, suite par exemple au déclenchement de la maintenance. La zone critique débute avec le marqueur rouge $M3$ de la figure 2.1 et se termine après 2 fois l'intervalle de temps spécifié pour les mises à jour. De même, si un pair disparaît juste après l'émission de son message ALIVE, dont le fonctionnement est présenté à la figure 2.2, les voisins découvriront son absence avec un temps borné entre 1 fois l'intervalle de temps des mises à jour et 2 fois l'intervalle de temps des mises à jour. Pour évaluer l'impact de cet intervalle de temps et choisir le plus adapté à nos besoins, nous le faisons varier lors des simulations à la section 2.3.

Enfin, si des problèmes interviennent hors des zones critiques, la gestion se fait via la procédure normale de vérification du voisinage qui a lieu à intervalle de temps régulier. Maintenant d'un point de vue technique la fermeture du canal de communication suffit à alerter le pair concerné d'une anomalie avec son voisin. Aussi, les tables de routage du système P2P sont symétriques mais cela n'est pas une nécessité. C'est d'abord le pair entrant qui enregistre ses voisins puis les informe avec une requête de confirmation. Quel que soit le délai d'arriver de la requête, il n'y a pas de risque que les voisins qui sont des pairs actifs suppriment le pair entrant à moins que le délai de la requête n'atteigne

2 fois le temps spécifié pour les mises à jour. Autrement dit, c'est comme si le pair avait disparu. La procédure de vérification du voisinage consiste à évaluer à intervalle de temps régulier les modifications locales qui sont apparues. Pour ce faire, un pair actif regarde si ses descendants et son ascendant sont bien présents dans sa table de routage.

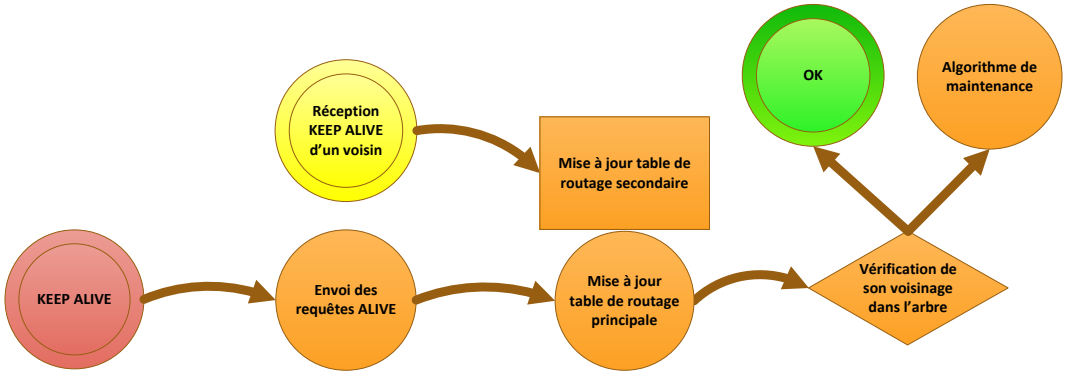

FIGURE 2.2 – Protocole d'échange des messages ALIVE

La figure 2.2 présente le protocole de message pour le maintien à jour des tables de routage. Chaque pair actif reçoit à intervalle régulier, au délai près, des messages ALIVE de ses voisins, état d'entrée jaune sur la figure 2.2. Ces messages mettent à jour une table de routage secondaire. Lorsque un pair actif commence son processus KEEP_ALIVE, état d'entrée rouge sur la figure 2.2, il émet en utilisant sa table principale l'ensemble des messages ALIVE à son voisinage. Puis il met à jour sa table de routage principale en fonction de la table secondaire. C'est aussi à ce moment la qu'un pair vérifie son ascendant et ses descendants dans l'arbre pour lancer, si nécessaire, un algorithme de maintenance. Nous venons de présenter les deux principaux processus qui régissent notre système P2P au niveau du réseau recouvrant.

Politique de gestion des adresses

Lorsque un pair entrant a obtenu quelques coordonnées (ou adresses hyperboliques), il est en choisir une. Ce choix interne peut être aléatoire ou utiliser un algorithme. Dans notre cas nous sélectionnons la coordonnée la plus proche de la racine. Nous essayons ainsi de tendre vers un arbre de diamètre minimal. De plus, le degré des pairs est dépendant de l'algorithme de sélection puisque c'est lui qui conditionne le choix du point d'attachement. Cependant ici le seul critère qui nous importe est la proximité d'une adresse avec la racine. C'est pourquoi afin de favoriser la bonne répartition des pairs dans l'arbre d'adressage notre sélection est basée sur cette proximité avec la racine mais ne dépend pas de caractéristiques réseaux liées à la topologie sous-jacente.

Un pair peut donner un nombre limité d'adresses (i.e. coordonnées hyperboliques) fixés par la racine mais le degré du pair au sein du réseau virtuel n'a aucune limite. Potentiellement un pair peut se connecter à n'importe quels autres pairs à chaque instant. Lorsque un pair entrant demande une coordonnée deux cas de figure se présentent alors.

Soit tous les pairs sollicités répondent à la requête du pair entrant qu'ils disposent ou non d'une coordonnée libre. À partir du moment où une adresse valide est extraite des requêtes le pair la choisit. Puis, si d'autres adresses sont extraites il peut effectuer une sélection à l'aide d'un algorithme. Soit, dans l'autre cas de figure, aucune des coordonnées reçues n'est valide alors le pair relance le processus de démarrage. Le nombre de pairs à contacter lors du démarrage est configuré par l'utilisateur final en fonction des capacités de son équipement. Dans nos simulations dynamiques nous fixons le nombre de requêtes à 3. Autrement dit, lors de son entrée dans le réseau, le pair communique avec trois autres pairs. Nous prenons un degré de départ bas pour ne pas créer un graphe complet mais au contraire favoriser un réseau assez large et profond afin d'évaluer des itinéraires de routage intéressants.

Suite à l'obtention d'une adresse, un pair calcule les coordonnées hyperboliques de ses futurs descendants à l'aide de l'algorithme présenté au chapitre 1. La structure arborescente est donc construite en même temps que le RR. Nous rappelons que les coordonnées hyperboliques (i.e. nombres complexes) des pairs sont utilisées comme les adresses de correspondance dans le RR. Les coordonnées des pairs correspondent aux sommets des ∞-gones hyperboliques. L'arbre hyperbolique résultant des ∞-gones étant q-régulier, le degré est, de fait, fixé à q par la racine. Le degré q de l'arbre est fixé par le premier pair au commencement et pour toute la durée de vie de l'arbre d'adressage. Néanmoins cela n'empêche pas le réseau de s'étendre car le degré du RR n'est lui pas limité. Une étude de l'influence du degré sur la capacité d'adressage est faite dans ce chapitre. Comme nous l'avons observé un degré élevé ne favorise pas la capacité d'adressage (le nombre des coordonnées). Nous utilisons l'arbre q-régulier théoriquement infini pour assigner les coordonnées aux pairs, ainsi un pair de l'arbre d'adressage peut donner $q-1$ adresses qui correspondent à ses descendants dans l'arbre. Le premier pair actif prend l'adresse hyperbolique $(0,0)$ et devient la racine de l'arbre qui est placée au centre du disque euclidien. Chaque pair actif exécute le calcul pour ses descendants, l'algorithme 1 montre comment les adresses sont calculées pour chaque pair. Toutes les étapes de l'algorithme présenté sont adaptées pour un calcul distribué et asynchrone. Pour les mêmes raisons que précédemment, nous fixons le nombre des coordonnées calculables à 5 qui est une petite valeur. La racine peut assigner q adresses alors que les autres peuvent assigner $q-1$ adresses. Tout pair, autre que la racine, peut donc donner 4 coordonnées. Nous créons des pentagones idéaux qui délimitent à chaque étape 4 nouvelles régions (donc coordonnées) dans le plan hyperbolique.

Router dans le réseau

Un pair entrant peut commencer à émettre des paquets dans le réseau recouvrant après avoir obtenu une coordonnée d'un pair actif. Le processus de routage est effectué dans chaque pair traversé depuis l'émetteur jusqu'à la destination en utilisant un algorithme glouton basé sur la distance hyperbolique définie par l'équation 1.1.

À la réception d'un paquet, un pair calcule la distance entre la destination et chacun de ses voisins puis il vérifie que la distance calculée est inférieure à la distance entre lui et la destination. Si la condition est respectée, le pair transfère le paquet au voisin dont la distance est minimale, voir l'algorithme 2. Si aucun voisin n'est plus proche que le pair

Algorithme 2 : Router un paquet dans le réseau recouvrant

```
1  ProchainSaut(pair, paquet) return Pair ;
2  begin
3      w = paquet.destinationPairCoords;
4      m = pair.Coords;
5      d_min = argcosh(1 + 2 |m−w|² / ((1−|m|²)(1−|w|²)));
6      p_min = pair;
7      forall the voisin ∈ pair.voisin do
8          n = voisin.Coords;
9          d = argcosh(1 + 2 |n−w|² / ((1−|n|²)(1−|w|²)));
10         if d < d_min then
11             d_min = d;
12             p_min = voisin;
13         endif
14     endfall
15     return p_min
16 end
```

lui-même alors le paquet a atteint un minimum local. Dans ce cas, d'autres méthodes expliquées dans la gestion des graphes dynamiques doivent être utilisées pour réussir à router le paquet vers la destination. Si aucune des autres méthodes ne réussit alors le paquet est abandonné.

Nous l'avons expliqué dans l'état de l'art, plusieurs espaces de nommage doivent être liés par un mécanisme de résolution de nom-adresse. Pour notre système P2P nous pouvons utiliser une table de hachage répartie sur les pairs. En effet, l'idée est qu'un pair ne communique dans le réseau qu'avec des identités de type nom indépendant de la position dans le réseau. Nous présentons les prémisses de cette solution au chapitre 3.

2.1.3 Gestion des graphes dynamiques

Notons que chaque pair du réseau recouvrant calcule ses coordonnées dans le plan hyperbolique en ayant pour seule connaissance un point d'attachement dans l'arbre d'adressage. De plus, Le réseau recouvrant peut s'étendre et se rétrécir au cours du temps. Le graphe que nous plongeons peut être issu d'un réseau *ad hoc* ou d'un réseau logique construit sur un réseau filaire. Ce type de réseau est construit au fil de l'eau et forme notre graphe de départ. Le routage glouton dans le plan hyperbolique est robuste tant que la cohérence de la structure d'adressage est préservée. Cette cohérence est fortement liée à la résilience des pairs qui forment le graphe et l'arbre d'adressage correspondant. Les pannes que nous traitons sont des pannes qui entraînent la disparition des pairs. Si un pair de l'arbre d'adressage autre qu'une feuille disparaît, alors l'arbre est brisé en une forêt de q sous arbres. Cependant, le graphe peut lui rester connexe.

Dans cette sous section, nous proposons quelques solutions pour maintenir la structure d'adressage et pallier les minima locaux. En effet, le routage dans le plan hyperbolique

est robuste tant que l'intégrité de l'arbre est maintenue. Dans un environnement réseau réel, il est prévu que les défaillances des liens ou des pairs arrivent souvent. Clairement, l'inconvénient du routage glouton « classique » est la résistance aux pannes [CC09]. Nous devons considérer deux niveaux de pannes :

– Le premier niveau correspond aux pannes de l'arbre d'adressage q-régulier.
– Le second niveau correspond aux pannes dans le graphe du réseau recouvrant.

Restaurer un arbre brisé

Au premier niveau, si un lien dans l'arbre est rompu alors le routage glouton hyperbolique échouera pour les chemins traversant ce lien. Pis, si un pair autre qu'une feuille tombe en panne, l'arbre est brisé en une forêt d'au plus q arbres qui inévitablement perturbe la connectivité du graphe. Nous introduisons cette problématique afin de justifier les choix que nous faisons pour la gestion de la volatilité dans les réseaux dynamiques.

Si l'arbre d'adressage est brisé, nous pouvons alors utiliser des techniques de réparation ou de maintenance de forêt d'arbre telle que la marche aléatoire qui est une approche connue pour l'exploration des graphes. L'idée est attrayante mais dans notre cas nous ne pouvons pas garantir la consistance des adresses lorsque l'arbre fusionne, en d'autres termes, cela aurait plutôt tendance à créer d'autre minima locaux. En effet, la fusion sans mise à jour des coordonnées des pairs du sous-arbre n'est pas acceptable.

Une autre technique, pour les descendants perdants leur ancêtre, consiste à essayer d'établir une connexion avec leur premier ancêtre (i.e. grand-parent) aussi bien qu'avec leurs frères. S'ils réussissent alors le routage glouton hyperbolique sera garanti, à condition que le graphe soit connexe, sinon l'arbre d'adressage reste brisé et l'algorithme glouton échoue à cause du problème des minima locaux. La méthode de restauration maintient la validité du routage glouton en connectant les pairs fils avec leur grand-père dans l'arbre d'adressage. Cette solution est peu coûteuse car les adresses sont gardées et l'arbre est stabilisé. De plus, l'implantation est aisée car il suffit d'un échange de message entre les deux pairs et le routage glouton hyperbolique reste garanti. En effet, d'après la définition du plongement glouton, i.e. pour toutes paires distinctes de sommet $s, t \in V(G)$ il existe un pair u adjacent à s tel que $d(f(u), f(t)) < d(f(s), f(t))$. Alors, pour le pair u il existe un pair u' qui respecte la même propriété. De ce fait, si u disparaît nous connectons u' à s conservant ainsi le plongement glouton. Les pairs fils peuvent aussi être connectés entre eux et par la suite, le pair en panne pourrait être remplacé par un nouveau pair avec les mêmes connexions mais la mise en œuvre est contraignante. Dans un contexte distribué, il semble difficile pour un nouveau pair d'être configuré avec les mêmes connexions que celles du pair défaillant sans centraliser et mémoriser. Aussi cette solution engendre des problèmes complexes. Par exemple, si nous nommons T l'arbre initial contenant l'ensemble des sommets d'un graphe G sous-jacent. Tout arbre né de la disparition d'une arête ou d'un pair et ne contenant pas la racine, notée r, est nommé sous-arbre, quel que soit sa taille. L'arbre T est le seul arbre possédant la racine ou le seul arbre possédant r est T. T peut ne contenir que la racine, si la racine disparaît T disparaît à moins d'utiliser un algorithme d'élection mais aucun algorithme d'élection n'est considéré dans ces explications.

Dans un premier temps nous essayons de traiter les défaillances liées aux arêtes. Lorsqu'une arête casse, il y a une perte de la connexité, en d'autres termes la disparition d'une

arête crée un sous arbre. G est alors couvert par deux arbres. Un des deux arbres possède nécessairement la racine de l'arbre antérieur à la défaillance, c'est T. Lorsqu'une arête disparaît (quel que soit la raison) le pair appartenant au sous arbre résultant pourrait lancer un algorithme de maintenance afin de se raccrocher à T. Le pair exécutant l'algorithme est le pair ayant subi la déconnexion. Le but de l'algorithme est alors de regarder localement si dans le graphe sous-jacent un lien existe entre le pair où il s'exécute et l'autre arbre. L'inondation pourrait être utilisée.

- Si le lien sous-jacent existe alors les deux arbres peuvent fusionner (le sous arbre peut se raccrocher à T). Après la fusion l'algorithme se termine et T doit changer les adresses des pairs du sous arbre.
- Sinon nous propageons l'algorithme dans l'arbre à chaque fils (par un parcours en largeur ou profondeur ou autre) qui consiste à émettre un message.

En d'autres termes nous exécutons l'algorithme sur chaque pair de l'arbre l'un après l'autre. Cette approche l'un après l'autre permet d'éviter un problème décrit dans [PCGC10], la simultanéité. Si aucun lien sous-jacent n'existe alors l'algorithme se termine sans succès ou est relancé. Ensuite lorsqu'un pair perd l'arrête le liant à son père cela veut dire qu'il s'est détaché de T. C'est pourquoi, le pair « victime », s'il ne possède pas d'autre lien vers T, doit aussi propager l'information de non appartenance à T. Le coût de cette propagation est d'un booléen et d'un nombre de messages égal au degré pour chaque pair dans le sous arbre. Lorsque le sous arbre fusionne, il propage cette fois l'information d'appartenance à T. Donc c'est un mécanisme réalisable mais avec plus de sous arbres la solution devient impraticable. Nous venons seulement de décrire le cas « simple », c'est-à-dire mettant en jeu un arbre principal T et un sous arbre. En effet, le problème avec plus de sous arbre n'est pas trivial, nous ne le détaillerons donc pas plus puisque nous n'y apportons pas de solution particulière. En revanche, dans leurs travaux, Blin et al. fournissent un formalisme des mécanismes que nous venons de décrire [BGPB$^+$09, BDPBR10, BGPB$^+$10]. Malheureusement, la découverte de ce formalisme a eu lieu tardivement. Ils donnent des mécanismes, via un formalisme spécifique, pour permettre à un arbre couvrant de se maintenir en milieu dynamique. Toutefois la mise en œuvre ne semble pas si évidente. Enfin, dans la restauration avec l'ancêtre, il faudrait que dans une lignée seul un pair sur deux tombe en panne sous peine de créer quelques pairs avec des degrés très élevés (des *HUB*). En effet, suivant la position du pair par rapport à la racine, il peut devoir accueillir plusieurs générations de descendants.

L'article [PCGC10] présente une technique permettant l'émergence d'une nouvelle racine au sein d'une forêt d'arbres dynamique. Chaque nœud contient un jeton qu'il fait fusionner avec ses voisins. De fusion en fusion, le nombre de jetons converge vers 1 afin d'élire la nouvelle racine. Contrairement à nos explications précédentes, dans ce cas, chaque arbre (ou sous arbre) de la forêt contient une racine.

Cependant, le maintien de l'arbre est une chose mais garder la cohérence hiérarchique des adresses en est une autre. En effet, après la fusion d'un sous arbre, il est nécessaire de vider et de redonner l'ensemble des adresses en fonction du nouveau point d'attachement. Or ce nouveau point d'attachement peut être sous optimal. Alors que le fait de casser le sous arbre permet de répartir les pairs sur la topologie existante puisqu'il relance la procédure pour joindre le réseau (P2P ou *ad hoc*). Donc deux approches peuvent être utilisées pour restaurer la connectivité :

- Méthode de *purge* : consiste à purger les adresses des pairs à partir du lien ou du pair en panne et réaffecter les adresses à tout ces pairs.
- Méthode de *restauration* : consiste à restaurer l'arbre en remplaçant le lien défectueux par un nouveau lien identique ou le pair en panne par un nouveau pair avec les mêmes connections.

La première solution que nous appelons la méthode de *purge* peut être coûteuse si la taille du réseau recouvrant est très large et/ou si les pairs au-delà de la panne sont nombreux car cela peut mener à ré-adresser une vaste partie du réseau. La seconde solution que nous appelons la méthode de restauration est moins coûteuse car les adresses sont gardées mais l'implémentation est plus complexe dans le cas de la panne d'un pair. En effet, il est difficile pour un nouveau pair d'être configuré avec les mêmes connections que celles du pair défaillant qu'il remplace d'autant plus en environnement distribué. Mais quel que soit les solutions existantes, le talon d'Achille reste la racine. Pour les différentes raisons énoncées précédemment nous utiliserons la méthode de *purge*.

Au niveau du graphe

Au second niveau, si le lien du réseau recouvrant n'est pas touché par la panne dans l'arbre d'adressage ou si c'est un pair feuille alors le routage glouton hyperbolique s'effectuera sans erreur au travers des chemins pouvant être plus long. L'arbre d'adressage est embarqué dans le réseau recouvrant, cela signifie qu'il existe des liens tierces, potentiellement des raccourcis qui peuvent devenir la seule route possible. Au cours du temps, quand les solutions ci-dessus sont utilisées pour restaurer l'arbre d'adressage, d'autres techniques peuvent être utilisées pour garantir la réussite du routage. En effet, des minima locaux se créent donc pour surmonter ces contraintes, il est nécessaire d'utiliser une méthode de routage alternative.

La panne d'un pair entraîne la libération de son adresse par le pair ascendant. C'est pourquoi, lorsque l'arbre d'adressage est brisé, nous utiliserons les deux approches combinées suivantes :

1. Méthode de *renumérotation* : purger les adresses des pairs du sous arbre d'adressage enraciné dans le pair en panne et faire que ces pairs obtiennent de nouvelles adresses à partir de pair actif.
2. Méthode de *contournement / exploration* : explorer les alentours d'un pair en panne afin de pouvoir contourner les minima locaux éventuels créés par cette panne en attendant une réparation définitive par renumérotation ou restauration.

La première solution peut être coûteuse selon la taille du sous arbre car elle peut mener à réétiqueter une vaste partie du graphe. Néanmoins, la mise en œuvre est très simple. Il est aussi possible de ne pas utiliser de racine. Il suffit d'avoir un cœur de réseau fortement connecté (e.g. graphe complet) comme les AS. Finalement le cœur de réseau est distribué et correspond à la réalité technique déployée actuellement. Où une autre alternative, même si la racine est perdue, Interdire au premier fils de la racine de lancer une purge des adresses Pour la deuxième méthode, la *marche aléatoire* est une technique possible attrayante pour réparer une forêt d'arbre car c'est une approche naturelle pour l'exploration des graphes mais dans notre cas nous ne pouvons pas garantir la consistance des adresses lorsque l'arbre fusionne, en d'autres termes, cela créerait des minima locaux.

Dans le disque de Poincarè figure 1 une marche aléatoire n'est pas efficiente parce qu'elle tend vers l'infini [Ghy06]. Donc pas de *marche aléatoire*.

Malgré tout, ces méthodes peuvent ne pas suffire à garantir un taux de routage de 100%. C'est pourquoi, il est nécessaire d'utiliser une méthode de routage alternative par exemple *Gravity-Pressure*. Pour surmonter les minima locaux, l'heuristique appelé « *Gravity-Pressure* (GP) » est présenté dans [CC09] et aussi utilisé dans [PKBV10]. Lorsqu'un paquet arrive dans un pair qui est un minimum local il entre en mode « *pressure* ». Dans ce mode, le paquet maintient une liste des pairs qu'il traverse et le nombre de fois qu'il les traverse. Ce processus continue jusqu'à ce que le paquet trouve un pair dont la distance avec la destination est plus petite que la distance avec le minimum local. La solution présentée requiert le stockage de nombreuses informations dans l'en-tête du paquet laquelle peut être difficile et lourde à implanter.

Routage alternatif

Sur la figure 2.3 la destination est marquée par D et la source par S. Le cercle jaune entoure l'ensemble des points à une distance hyperbolique inférieure à la distance (D, S). Le cercle rouge entoure l'ensemble des points à une distance hyperbolique inférieure à la distance (D, W). Pour router de manière gloutonne la source doit avoir un voisin dans le cercle jaune (ici W via un lien du graphe en noir) puis W doit avoir un voisin dans le cercle rouge (ici T). Enfin, T atteint la destination.

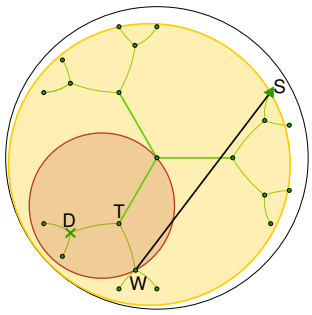

FIGURE 2.3 – Routage glouton et minimum local

Minima Locaux. Si l'algorithme de routage glouton ne trouve pas de meilleur solution que lui-même alors il échoue car le message a atteint un minimum local. Par exemple, si le pair W n'a pas de voisin dans le cercle rouge alors le routage glouton échoue, W est un minimum local. Pour s'extraire de ce type de pair nous utiliserons les solutions suivantes :

- *Glouton relâché* : transférer le message au voisin dont la distance hyperbolique avec la destination est la plus petite. En effet, l'exigence du plus court chemin est relaxée afin d'éviter les minima locaux.
- *Expulsion* : transférer le message au voisin dont la distance avec le minimum local est la plus grande. Lorsque le message arrive dans le nœud voisin il est traité avec la méthode gloutonne classique. Si le voisin est aussi un minima local alors la méthode expulsion est réutilisée.
- *Arbre* : remonter dans l'arbre jusqu'à trouver un pair ayant un voisin dont la distance avec la destination est inférieure à la distance (minimum local, destination). À la racine, rejeter le paquet.
- *Gravity-Pressure*.

Un autre phénomène complexe à résoudre est celui des circuits dans le routage. Une technique utilisée dans [KPBV09] consiste à rejeter le message après deux même passages

dans un pair mais cela demande de stocker l'information. Nous optons plutôt pour une durée de vie du message en fonction du nombre de sauts (TTL).

Métrique à évaluer

Dans ce mémoire, nous abordons le problème du routage glouton. Tout naturellement le premier paramètre que nous analyserons sera le taux de délivrance des messages, notamment en milieu dynamique. Lors du routage nous comptabilisons l'ensemble des paquets émis. En revanche nous ne prenons pas en compte les pertes de paquets dues à la disparition du pair destination. Autrement dit, nous n'évaluons que les défaillances provenant de la fonction de routage c'est-à-dire quand elle ne peut pas choisir un voisin suivant. Ensuite, nous étudions la longueur des chemins qui est notée par $d(s,t)$ où s et t sont deux pairs du réseau. Cette distance doit être distinguée de la distance du plus court chemin entre s et t dans le graphe G, notée par $d_G(s,t)$ et calculée avec l'algorithme de Dijkstra. Ces deux notations nous permettent de définir l'étirement du routage glouton comme étant égale à $E = d/d_G$. Un plongement avec un étirement $E = 1$ est appelé un plongement sans étirement. Enfin nous évaluerons en octet la quantité d'information globale consommée par le protocole des messages ALIVE nécessaires au maintien du voisinage d'un pair.

Sur le plan théorique, notre algorithme présente de très bonnes propriétés pour la complexité en temps et en espace. En termes de complexité notre algorithme dépend du degré des pairs. En effet, l'intérêt d'un algorithme de routage local est que les pairs stockent uniquement les coordonnées de leurs voisins. Concernant la mémoire, les coordonnées virtuelles sont de taille fixe 16 octets. De ce fait, la table de routage d'un pair occupe deg entrées $*$ 16 octets. Ensuite, la complexité en temps comprend le temps d'extraction de la structure de données utilisée pour stocker les voisins de $O(1)$ à $O(n)$ pour une table de hachage locale et le calcul de la distance hyperbolique en $O(1)$; Soit : $deg*(O(n)+O(1))$.

2.2 Simulations statiques

Dans cette section, nous présentons les résultats des simulations que nous avons menées pour évaluer l'utilisabilité de notre système d'adressage et de routage basé sur les coordonnées du plan hyperbolique. Nous utilisons un simulateur réseau à évènements discrets de niveau paquet appelé *nem* [Mag05] pour l'obtention de tous les résultats de simulations présentés pour les topologies statiques.

2.2.1 Configuration et paramétrage des simulations

Pour évaluer les algorithmes de notre réseau virtuel sur une topologie réelle, nous utilisons des cartes Internet provenant de données réelles collectées avec *nec*[1] et CAIDA[2]. Nous utilisons une carte IPv4 avec 75k nœuds de 2003, une carte BGP4 avec 34k nœuds de 2010 et une carte IPv6 avec 4k nœuds de 2004.

1. http ://www.labri.fr/perso/magoni/nec
2. Université de Californie, San Diego - The Cooperative Association for Internet Data Analysis : http ://www.caida.org/home

Nous évaluons notre algorithme de routage sur des topologies statiques en considérant que chaque pair de la carte est un pair membre du réseau recouvrant. De ce fait, la topologie du réseau recouvrant est égale à la topologie de la carte Internet et peut être considérée comme un morceau d'Internet. Les simulations sont définies comme statiques parce que les nœuds sont toujours opérationnels au cours du temps. L'avantage des simulations statiques est le faible coût des calculs. De plus, cela permet l'évaluation du passage à l'échelle puisque nous pouvons utiliser tous les nœuds de la carte comme membres du réseau. Au niveau des coordonnées hyperboliques la précision de calcul traitée au chapitre 1 est fixée à 10^{-9}. C'est la précision maximale de la virgule flottante pour toutes les simulations. Évidemment, dans le cas d'un graphe statique, le taux de réussite du routage glouton hyperbolique est égal à 100%. En revanche, l'étirement des chemins est influencé en fonction de la congruence entre le graphe d'origine et le graphe couvrant plonger. C'est pourquoi, afin d'évaluer au mieux le différentiel entre l'algorithme de plus court chemin de Dijkstra et notre algorithme de routage glouton, nous affectons les coordonnées aux pairs avec un parcours en largeur ainsi l'arbre d'adressage hyperbolique est un arbre couvrant du graphe sous-jacent. Le point de départ de l'algorithme est sélectionné au hasard à chaque exécution. L'ensemble des points montrés sur les graphes suivants sont calculés avec un niveau de confiance de 95% et une erreur statistique relative de 5%.

2.2.2 Simulations

Nous étudions maintenant l'influence du degré de l'arbre sur la profondeur de l'arbre d'adressage. Nous adressons tous les nœuds de la carte avec un algorithme de distribution en largeur qui commence par un nœud tiré aléatoirement dans la carte ; ce nœud est la racine de l'arbre d'adressage. Les adresses sont calculées avec l'algorithme vu en section 1.1. Chaque nœud donne ses adresses à ses pairs voisins jusqu'à avoir l'ensemble de ses nœuds voisins adressés.

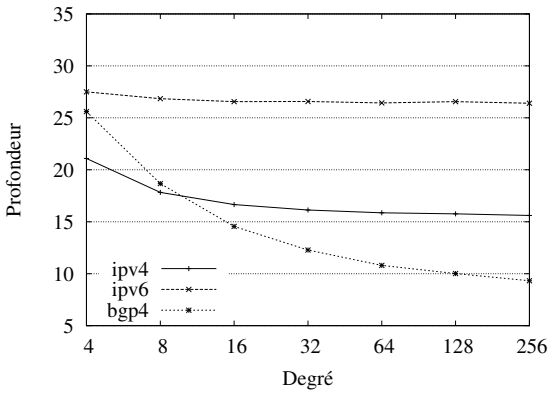

FIGURE 2.4 – Mesure de la profondeur maximale avec le plan d'adressage en arbre standard

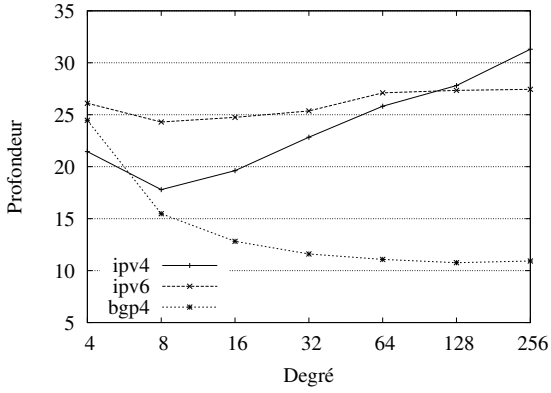

FIGURE 2.5 – Mesure de la profondeur maximale avec le plan d'adressage en arbre hyperbolique

Nous comparons deux plans d'adressage l'un en arbre standard tel que celui défini dans [SML06] et l'autre en arbre hyperbolique précédemment défini en section 1.1. Nous menons

des simulations pour les deux plans d'adressages et pour chacune des cartes Internet afin d'évaluer la profondeur maximale de l'arbre d'adressage, la longueur moyenne d'un chemin (mesuré en saut) et les métriques d'étirement et de congestion. Nous analysons ces métriques en fonction du degré de l'arbre d'adressage q.

Le plan d'adressage en arbre standard à une longueur d'adresse variable dépendante du rayon du graphe. En effet, le premier nœud est marqué 1, son quatrième descendant sera marqué 1.4, le sixième descendant de son descendant sera marqué 1.4.6 et ainsi de suite. Comme nous utilisons 1 octet pour chaque niveau de l'arbre standard, nous pouvons voir que le nombre d'octets dans une adresse dépendra de la distance maximale de la racine. Au contraire, dans le plan d'adressage en arbre hyperbolique chaque adresse a une longueur fixée de 16 octets (2∗8-octets ***double*** types). Cependant la profondeur de l'arbre d'adressage hyperbolique a une valeur maximale qui est atteinte lorsque les points sont trop proches du disque unité, du fait de la précision de calcul. La profondeur maximale observée dépend donc du degré choisi aussi bien que de la précision de calcul choisie. Notre but ici est d'essayer d'adresser tout les nœuds de la carte mais les simulations montrent que nous ne pouvons pas dépasser 90% environ.

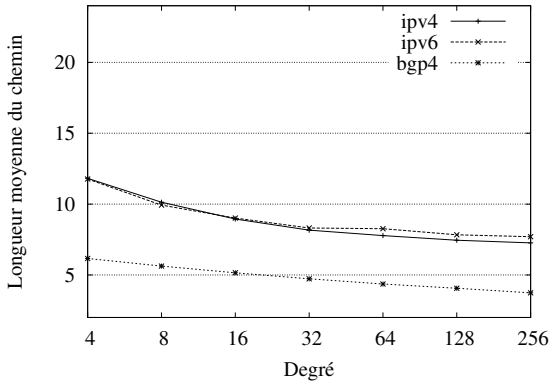

FIGURE 2.6 – Distance moyenne entre les pairs avec le routage lexicographique

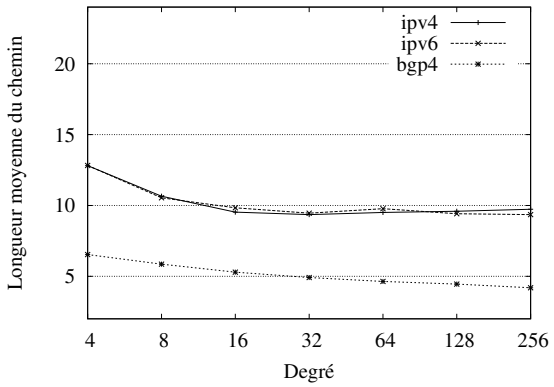

FIGURE 2.7 – Distance moyenne entre les pairs avec le routage glouton hyperbolique

Avec les figures 2.4 et 2.5, nous pouvons voir que dans l'arbre hyperbolique les résultats sont similaires à l'arbre standard excepté pour la carte IPv4 où la profondeur croît avec le

degré. Cependant cela n'a pas d'effet sur la taille des coordonnées hyperboliques puisque elles ont une longueur fixe. Cependant pour des topologies semblables l'arbre standard nécessite des adresses de longueur supérieur. Pour IPv6, 27 octets sont requis faisant donc de l'adressage hyperbolique un meilleur choix. Pour IPv4 et avec un degré supérieur à 16 les adresses ont une taille bornée entre 15 et 17 octets. Nous pouvons conclure de ces deux figures que la longueur fixe des adresses hyperboliques est un avantage. À noter une autre différence importante, l'adressage en arbre standard ne permet d'adresser que 256 nœuds à chaque niveau au maximum alors que le nombre d'adresses hyperboliques croît en fonction du degré avec la loi de puissance suivante : $\sum_{\substack{k=0 \\ q=degre}}^{n} q * (q-1)^k$.

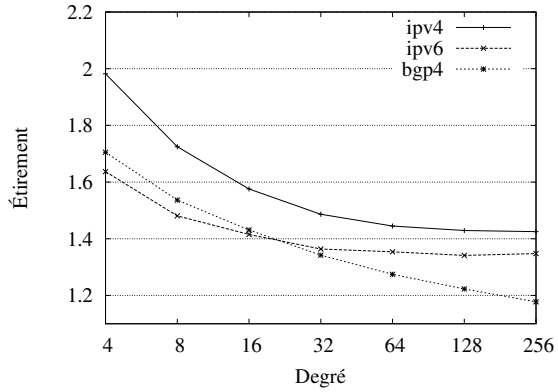

FIGURE 2.8 – Mesure de l'étirement moyen avec le routage lexicographique

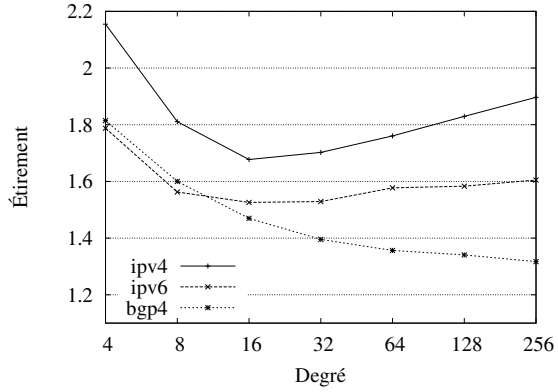

FIGURE 2.9 – Mesure de l'étirement moyen avec le routage hyperbolique

Nous étudions maintenant les métriques liées à l'évaluation du routage : la longueur moyenne d'un chemin, l'étirement et la congestion. Évidemment, nous n'étudions pas le taux de réussite des algorithmes de routage, car en configuration statique, ce taux est de 100%. La figure 2.6 montre la distance moyenne entre les nœuds avec l'utilisation du routage lexicographique (i.e. des adresses tel que 1.4.6 signifie que pour atteindre le nœud, la traverser de 1 et 1.4 est nécessaire). La distance est le nombre des sauts produits par ce schéma routage donc plus la distance est petite, meilleur est le routage. Nous voyons qu'avec un degré élevé les distances sont réduites. Les courbes sont approximativement

les mêmes figures 2.7, la différence est plus importante pour IPv4. Cela signifie que les chemins entre les nœuds sont plus éloignés dans le plan hyperbolique.

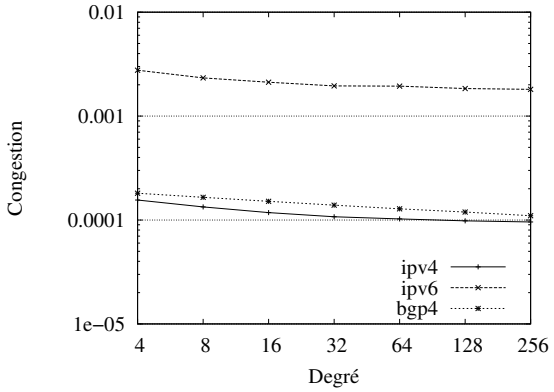

FIGURE 2.10 – Mesure de la congestion avec le routage lexicographique

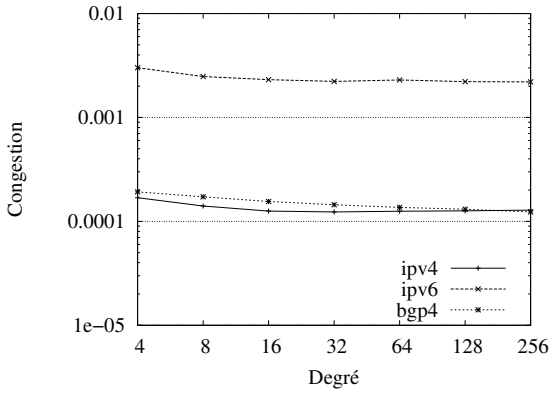

FIGURE 2.11 – Mesure de la congestion avec le routage hyperbolique

À présent, nous mesurons l'étirement du chemin pour les deux algorithmes de routage locaux. L'étirement est égal à la longueur du chemin avec le schéma de routage (standard ou hyperbolique) divisée par la longueur du plus court chemin global (i.e. le plus court chemin possible calculé d'une manière centralisée par l'algorithme de Dijkstra). Les figures 2.8 et 2.9 montrent que le degré a une influence sur l'étirement. L'étirement est meilleur quand nous routons dans l'arbre standard bien qu'il y ait un rendement décroissant pour les cartes IP quand le degré est supérieur à 16. Sur la figure 2.9 les meilleures valeurs d'étirement pour IP sont respectivement 1.7 avec un degré de 16 et 1.5 avec un degré de 32. La valeur de l'étirement la plus basse est 1.3 pour les cartes BGP pour un degré égal à 256. Dans l'algorithme de routage hyperbolique, quand le degré est supérieur à 32, l'étirement tend à se dégrader et croît encore pour les cartes IP. Cependant nous devons rappeler que l'arbre standard utilise un routage lexicographique lequel est moins robuste en présence de pair défaillant. De plus, le routage glouton hyperbolique est basé sur une vraie métrique (comme vu en section 1.1), cela permet en théorie aux paquets de toujours prendre le plus court chemin et d'être plus robuste en présence de pannes. Comme la topologie de notre réseau recouvrant est égale à la topologie de notre carte et

que la longueur du chemin entre les pairs est mesurée avec notre algorithme de routage, nous pouvons conclure que le chemin entre n'importe lesquels de ces pairs est plus éloigné dans le plan hyperbolique.

Nous remarquons aussi que les nœuds BGP ont un étirement plus faible entre eux, le résultat est attendu puisque nous mesurons ici des sauts de systèmes autonomes (AS) et non pas IP. Le nombre des sauts dans une carte d'AS est connu pour être beaucoup plus petit que dans une carte IP. Enfin, nous évaluons l'efficience de ces algorithmes de routage locaux en observant la congestion. Nous définissons la congestion moyenne d'un nœud comme le nombre des chemins le traversant divisé par le nombre total des chemins. Dans les figures 2.10 et 2.11, nous observons que la congestion reste très basse. De plus, elle est sensiblement la même entre les deux figures. Toutefois, nous notons que la carte IPv6 a une congestion plus élevée, cela est due à la petite taille de la carte par rapport aux deux autres. Finalement, contrairement à l'étirement, ces courbes montrent que le degré a une influence modérée sur la congestion.

2.3 Simulations dynamiques

Nous l'avons vu dans l'état de l'art le routage glouton sur des topologies dynamiques reste un challenge car jusqu'à présent les solutions présentées sont gourmandes en ressources réseau ou mémoire. Dans les simulations dynamiques, nous considérons qu'à un instant donné seulement quelques pairs sont actifs comme pairs du réseau recouvrant. Nous utilisons l'environnement de simulation réseau à évènement discret *OMNeT++* pour l'obtention de tous les résultats de simulations présentés pour les topologies dynamiques. Le simulateur gère un temps de simulation et chaque pair du réseau recouvrant démarre à un temps donné pour une durée donnée. Nous n'utiliserons plus de carte Internet comme topologie sous-jacente. Aussi, comme expliqué au chapitre 3, une topologie sous-jacente nécessite un algorithme de sélection de pair performant afin de rapprocher les pairs proche selon certaines caractéristiques.

2.3.1 Configuration et paramétrage des simulations

Le pair qui crée le réseau recouvrant reste actif pour toute la durée d'une simulation. Nous limitons aussi le nombre de coordonnées de la racine à 5. Car nous avons vu en section 1.1 que des degrés élevés n'améliorent pas les performances. Les paquets de données sont émis par chaque pair (ayant une adresse) à un régime de 1 toutes les 5 secondes.

Chaque pair a une durée de vie aléatoire fixée selon une probabilité décrivant une loi exponentielle avec $\lambda = 2.10^{-3}$ laquelle donne une valeur médiane de 350 secondes, une espérance de 500 secondes et une valeur au 90e percentile d'environ 1000 secondes. Le nombre des nouveaux pairs entrants par minute est dépendant de la taille du réseau évalué. Les pairs entrent avec un temps d'inter arrivé aléatoire fixé selon une probabilité décrivant une distribution exponentielle de $\lambda = 0.25$. Les pairs créent des liens dans le réseau recouvrant avec d'autres pairs en sélectionnant ceux qui ont l'adresse la plus proche de la racine. Nous collectons les données toutes les 100 secondes. Comme chacune des simulations dynamiques dure 7200 secondes, soit 2 heures, cette distribution des durées

de sessions des pairs produit de nombreux mouvements d'apparitions et de disparitions des pairs que nous qualifions de remous. Les remous représentent la volatilité inhérente aux participants des réseaux P2P ou *ad hoc*.

Deux moments sont intéressant dans la vie du RR. Premièrement la naissance du RR, les pairs entrent dans le réseau avec une certaine fréquence et le font grossir jusqu'à un niveau stable. Deuxièmement, quand le RR est sur le déclin ou subit de nombreux remous engendré par la volatilité des utilisateurs. Dans le premier cas, il est nécessaire de créer le RR et l'arbre d'adressage en même temps. Dans le second cas, comme dans un contexte de pannes, le challenge est de garantir la consistance de l'arbre d'adressage et le taux de délivrance. De plus, le RR peut s'étendre de manière irrégulière. Au commencement des simulations dynamiques, il y a une phase de démarrage et une phase de dissipation à la fin qui doivent être considérées toutes les deux comme des régimes transitoires. Dans nos simulations, nous stoppons les simulations durant le régime stable, c'est pourquoi nos graphiques ne montrent pas des courbes avec une phase de dissipation. Les mesures les plus significatives sont localisées après la phase de démarrage qui est d'environ 30 minutes. C'est alors que commence le régime stable.

2.3.2 Simulations

Préliminaires : réseau de 250 pairs

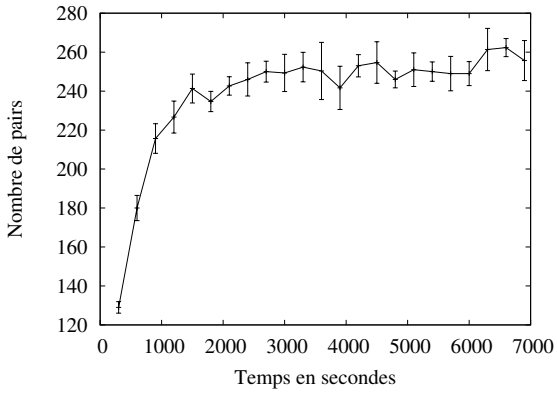

FIGURE 2.12 – Évolution de la taille pour un réseau de 250 pairs

La figure 2.12 montre le motif que suivent tous nos réseaux. Nous observons que le réseau atteint un régime stable autour de 1500 secondes. L'écart type indique la variation de la taille du réseau sur les dix exécutions. La taille du réseau est de 250 pairs. Le taux de remous implique que toutes les 60 secondes 12% de la population quittent le réseau. De même, toutes les 60 secondes, nous créons N pairs. N représente 12% de la population finale souhaitée. Ainsi le réseau stabilise sa taille afin d'atteindre un régime stationnaire. Nous rappelons que la durée de vie médiane d'un pair est de 350 secondes et qu'un pair émet un paquet toutes les 5 secondes.

Choix des paramètres de mises à jour avec un réseau de 500 pairs

La taille du réseau est de 500 pairs. Le réseau évolue de la même manière que pour un réseau de 250 pairs, le nombre de pairs introduit par minute est simplement plus important. Pour chaque courbe, nous avons réalisé 10 exécutions possédant chacune un jeu de données différent. L'écart type étant faible, nous n'avons pas augmenté le nombre d'exécutions.

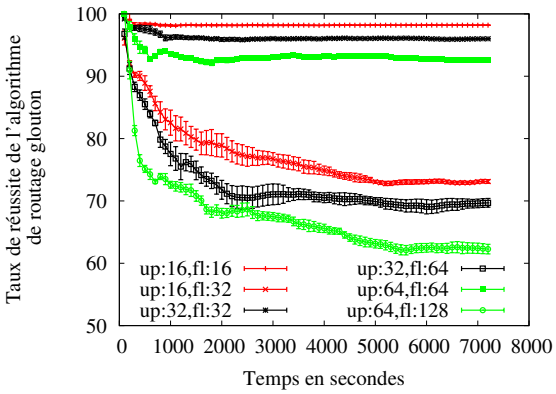

FIGURE 2.13 – Taux de délivrance du routage glouton vs. intervalle de temps des mises à jour et de la maintenance

La figure suivante 2.13 nous montre l'influence des mises à jour sur le taux de délivrance de l'algorithme de routage glouton dit classique.

- *up* représente l'intervalle de temps des mises à jour pour les tables de routage. Envoi des messages ALIVE à ses voisins.
- fl représente l'intervalle de temps avec lequel un pair vérifie la structure locale de l'arbre. Autrement dit, la présence des ancêtres et des descendants. Suite à la vérification, l'algorithme de maintenance est lancé ou non.

L'intervalle de temps des mises à jour est spécifié par la variable *up* de la figure 2.13. Nous observons que la fréquence des *purges*, représentée par la variable fl, a aussi un impact mais plus faible. En effet, cela représente la réactivité avec laquelle les pairs voisins à la panne répercutent l'événement. Lorsque cela n'est pas fait, cela signifie que les pairs descendants et le pair ascendant sont les seuls informés. De ce fait, les chemins de routage ne sont pas modifiés puisque l'information de panne reste locale. Les pairs deviennent alors des minima locaux (ou des puits).

La figure 2.13 nous permet d'observer que c'est en synchronisant les valeurs de *up* et de fl que nous obtenons les meilleurs résultats. Donc pour nos futures simulations nous prendrons $up = 16$ et $fl = 16$. En effet, la seule structure de donnée qui permet à un pair de disposer d'information sur son voisinage est la table de routage. Lorsque les valeurs *up* et fl ne sont pas égales cela signifie que la vérification du voisinage local est reportée repoussant le mécanisme de maintenance destiné à pallier la panne. Donc plus de paquets sont perdus. En revanche, choisir une valeur de fl plus faible que *up* n'a pas de sens car fl utilise la table de routage pour effectuer sa vérification donc si la table de routage n'est pas encore à jour, il est inutile de la vérifier.

Les paquets perdus

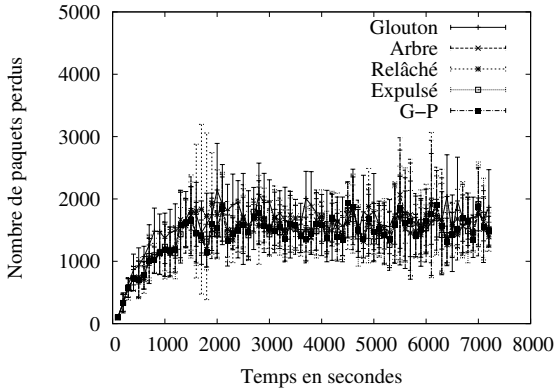

FIGURE 2.14 – Cumul des paquets perdus par unité de temps dans un réseau de 500 pairs

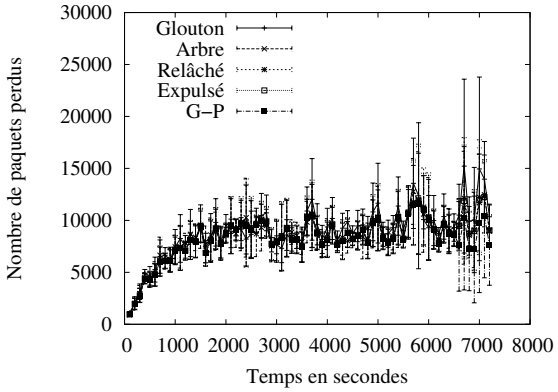

FIGURE 2.15 – Cumul des paquets perdus par unité de temps dans un réseau de 2000 pairs

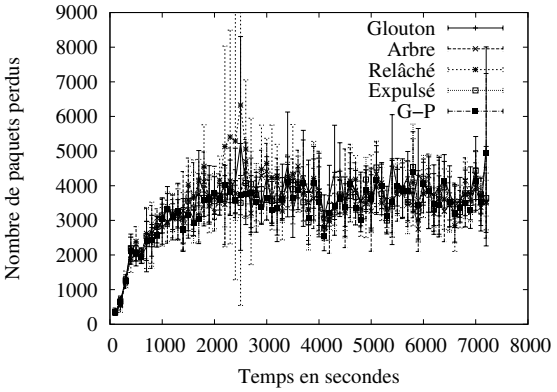

FIGURE 2.16 – Cumul des paquets perdus par unité de temps dans un réseau de 1000 pairs

Un paquet perdu signifie que le paquet est rejeté par une couche inférieure à la couche du RR. Cela n'est en aucun cas l'algorithme de routage qui est mis en défaut. Un paquet peut être correctement routé au niveau du RR mais sans pour autant pouvoir atteindre sa destination, à cause de la topologie dynamique. Les paquets sont alors comptés perdus. Nous verrons par la suite que le nombre des paquets perdus peut avoir de l'importance pour l'interprétation des résultats de réussite. En effet, si l'écart entre les courbes est petit cela signifie que le taux de délivrance des paquets est parfaitement représentatif dans le sens où il a été évalué avec plus de paquets.

Des écarts importants sur les courbes de la figure 2.17 pourraient remettre en question les résultats visibles sur la figure 2.23. En effet, si une courbe diverge alors il faut relativiser le taux de réussite associé. Si cela diverge à gauche c'est que moins de paquets sont pris en compte (car perdus). Inversement, si cela diverge à droite. Globalement, pour toutes nos

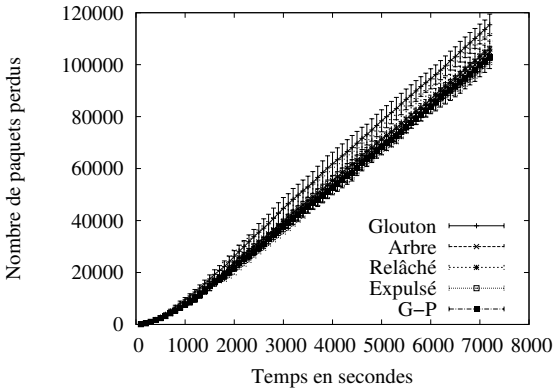

FIGURE 2.17 – Cumul des paquets perdus dans un réseau de 500 pairs

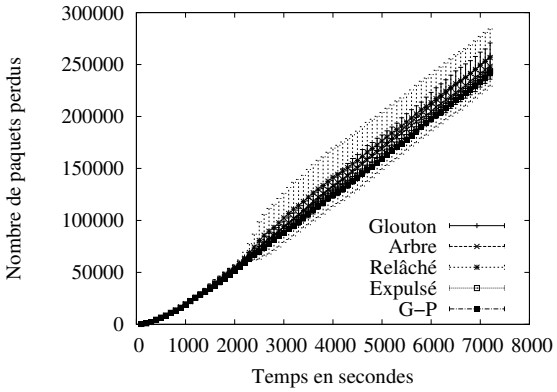

FIGURE 2.18 – Cumul des paquets perdus dans un réseau de 1000 pairs

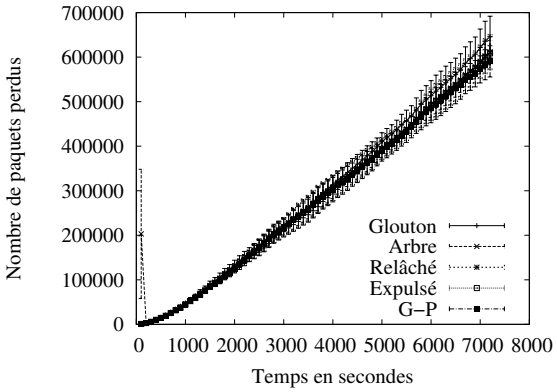

FIGURE 2.19 – Cumul des paquets perdus dans un réseau de 2000 pairs

simulations les divergences sont faibles. La figure 2.14 montre les paquets perdus entre chaque collecte d'information.

La figure 2.18 et la figure 2.19 montrent des écarts types en relation avec la figure 2.27 qui restent cohérents. L'algorithme *relâché* se démarque encore puisque il affiche un écart type plus important.

Test des paramètres de mises à jour avec 1000 pairs

Pour les résultats de simulations présentés par la figure 2.20, nous avons essayé de modifier la fréquence de l'algorithme de maintenance dans le cas des réseaux de 1000 pairs. Nous avons fixé $up = 16s$ et $fl = 32s$. Pour ces résultats nous n'avons effectué qu'une exécution. Nous pouvons observer que les courbes se séparent plus nettement et se regroupent. Pour un réseau de 500 pairs nous avions vu l'effet pour un même algorithme, ici nous observons l'effet sur les différents algorithmes alternatifs.

Il semble que l'absence de mise à jour influence moins l'algorithme qui utilise le mode alternatif de l'arbre. Paradoxalement, c'est l'algorithme de routage alternatif utilisant

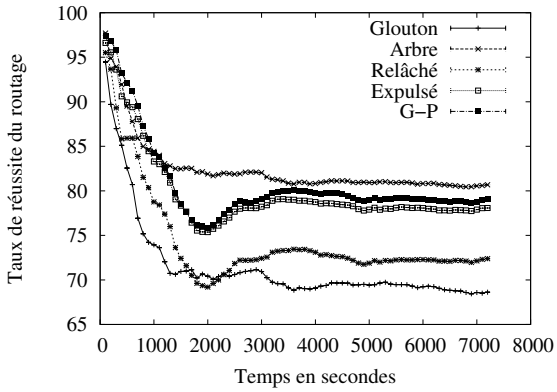

FIGURE 2.20 – Taux de délivrance des différents algorithmes de routage dans un réseau de 1000 pairs avec $up = 16$ et $fl = 32$

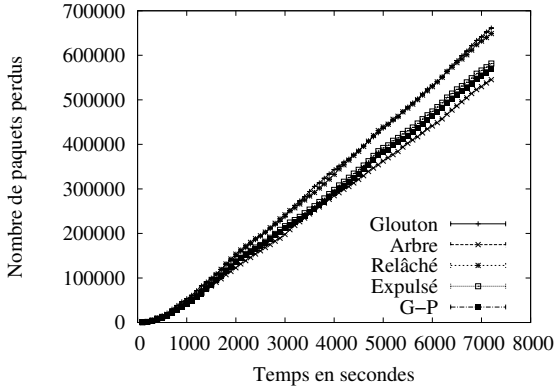

FIGURE 2.21 – Cumul des paquets perdus dans un réseau de 1000 pairs

l'arbre qui obtient le meilleur taux de délivrance alors que justement il n'est plus possible, du moins pour les pairs fils à la panne de transmettre un message au pair qui a disparu. De plus, on observe que c'est l'algorithme qui a le moins perdu de paquets.

Nous observons la divergence et observons les mêmes regroupements que pour le taux de réussite. Lorsque $fl \neq up$, seuls les pairs directement concernés sont à jour puisque ils ne reçoivent plus de message ALIVE. Du coup leur table de routage est à jour. En revanche, la *purge* n'est pas mise en œuvre de suite.

Le pair en panne implique que l'arbre d'adressage est cassé mais le routage continue de fonctionner correctement dans l'ensemble des pairs puisque l'information n'est pas répercutée. De ce fait, tous les chemins qui traversent le pair en panne continuent d'être utilisés. Ce n'est qu'au moment où le paquet atteint les pairs voisins à la panne que ce produit le changement de mode de routage, si nécessaire, car les pairs voisins à la panne peuvent devenir des minima locaux. Peuvent car ils disposent certainement d'autres liens.

Enfin les figures 2.22 et 2.21 montrent la divergence la plus importante, notamment pour l'algorithme utilisant l'arbre qui, de ce fait, comptabilise moins de paquets.

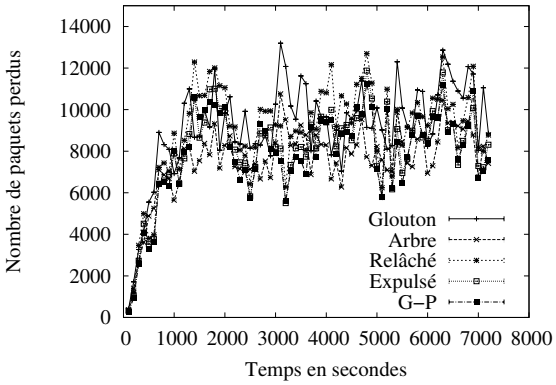

FIGURE 2.22 – Cumul des paquets perdus par unité de temps dans un réseau de 1000 pairs

Taux de délivrance des paquets pour toutes les tailles de réseaux

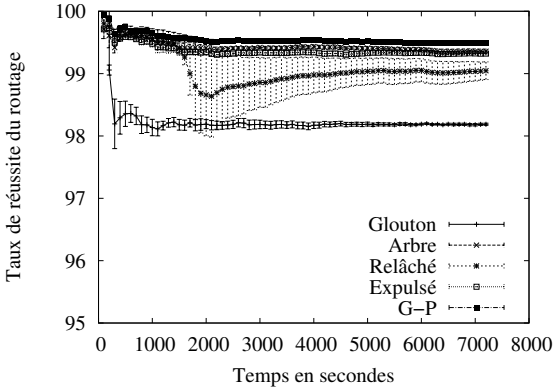

FIGURE 2.23 – Taux de délivrance des algorithmes de routage pour 500 pairs

Le figure 2.23 nous montre les résultats des algorithmes de routage alternatifs. Chaque exécution utilise un jeu de données différent mais chaque algorithme utilise les mêmes jeux de données globaux. La courbe de l'algorithme *relâché* est marqué par un écart type plus grand et une variation significative. Vraisemblablement, l'algorithme a moins bien réagi à un événement que les autres. Nous avons alors étudié le détail des exécutions que nous retrouverons ensuite à la figure 2.28 laquelle nous montre que le motif suivi est la conséquence d'une exécution. De plus, l'événement se reproduit pour un réseau de 1000 pairs mais il est difficile de fournir plus d'explications car reproduire cette exécution est possible mais ne nous donnera pas plus d'information. Un monitoring avec une granularité plus fine pourrait peut-être nous éclairer plus.

Pour des tailles de réseaux de 1000 et 2000 pairs, nous procédons de la même manière que pour les réseaux précédents. Cependant, pour chaque courbe nous avons réalisé 5 exécutions avec un jeu de données différent au lieu de 10. Le jeu de données des 5 exé-

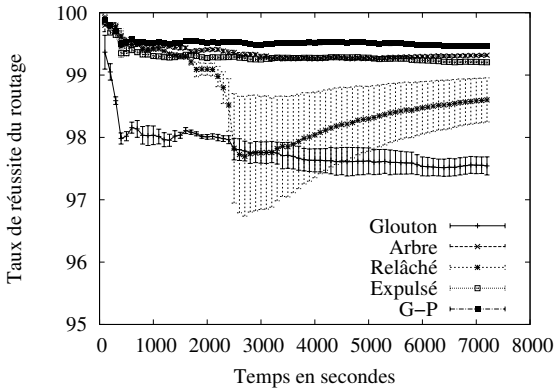

FIGURE 2.24 – Taux de délivrance des algorithmes de routage pour 1000 pairs

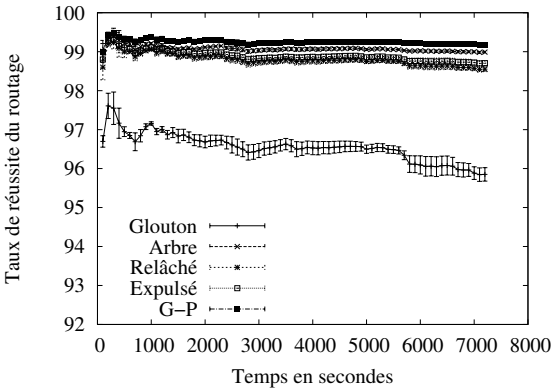

FIGURE 2.25 – Taux de délivrance des algorithmes de routage pour 2000 pairs

cutions est commun au 5 premières exécutions des simulations précédentes. La raison est qu'au delà de 1000 pairs le coût des simulations devient important en temps ainsi qu'en mémoire réduisant les possibilités d'exécutions simultanées.

Dans le cas de 1000 pairs le taux de délivrance des paquets reste très semblable à celui obtenu avec 500 pairs. La courbe de l'algorithme *relâché* a le même comportement. Comme vu dans la figure 2.27 un jeu d'exécution en est la cause. Aussi, pour le réseau de 2000 pairs l'événement ne s'est pas reproduit. Enfin, la diminution du nombre d'exécutions donne des courbes moins lissées. Toutefois, le taux de délivrance reste supérieur à 99%.

Quel que soit la taille du réseau l'algorithme *Gravity-Pressure* continue d'obtenir le meilleur taux de délivrance. Nous notons une faible diminution du taux de réussite qui baisse à 99% pour un réseau de 2000 pairs.

Détail du taux de délivrance pour l'algorithme *Gravity-Pressure*

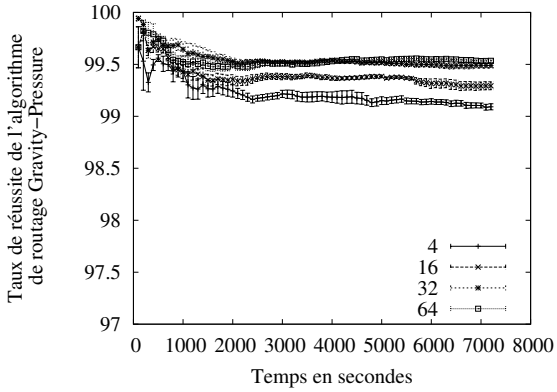

FIGURE 2.26 – Taux de délivrance de l'algorithme G-P en fonction de la taille des entêtes du paquet pour 500 pairs

La figure 2.26 présente, sur le même type de réseau que précédemment, l'influence de

la taille des entêtes des paquets pour le taux de délivrance de l'algorithme de routage *Gravity-Pressure* (G-P). Chaque courbe est la moyenne de 5 exécutions. Nous observons que la taille des entêtes ne fait pas baisser significativement le taux de délivrance. À partir des entêtes de taille 32, les courbes ne montrent pas de gain significatif. En revanche, la diminution de la taille des entêtes à un impact immédiat bien que faible.

Détail du taux de délivrance de l'algorithme *relâché* avec des réseaux de 500 et 1000 pairs

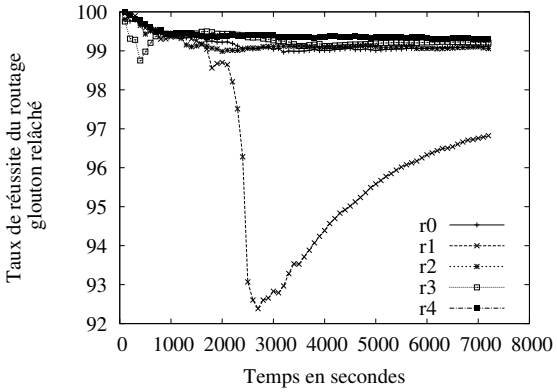

FIGURE 2.27 – Détail des exécutions de l'algorithme de routage glouton *relâché* pour 1000 pairs

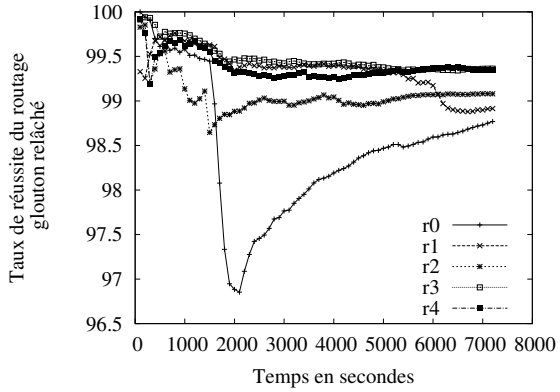

FIGURE 2.28 – Détail des exécutions de l'algorithme de routage glouton *relâché* pour 500 pairs

La figure 2.27 montre le détail des exécutions pour l'algorithme de routage glouton *relâché*. Comme annoncé, c'est une exécution qui influence très nettement la courbe et son écart type. Chaque exécution utilise un jeu de données différent mais les algorithmes de routage sont évalués avec les mêmes jeux de données.

Dans ce cas, les courbes de la figure 2.28 sont les 5 exécutions de l'algorithme *relâché* avec 500 pairs afin de comparer les deux figures sur le même nombre d'exécutions. La figure 2.28 montre que le taux de réussite pour un réseau de 500 pairs présente le même motif de courbe que pour un réseau de 1000 pairs. On remarque que l'augmentation du nombre d'exécutions influence seulement le routage glouton *relâché* alors que les autres algorithmes sont plus stables.

Taille des tables de routage pour toutes les tailles de réseaux

Ensuite la figure 2.29 nous montre une caractéristique importante pour un algorithme de routage, la taille des tables de routage en moyenne. Comme espéré la taille des tables de routage est minime. Elle est en moyenne de 5.

La figure 2.30 et la figure 2.31 montrent des tables de routage qui en moyenne gardent la même taille. La taille des tables est donc indépendante de la taille du réseau, condition sine qua non pour un passage à l'échelle. De manière logique les figures 2.32 et 2.33 montrent des longueurs de chemins qui augmentent.

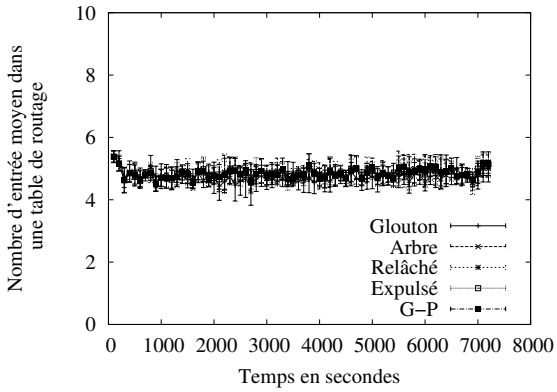

FIGURE 2.29 – Taille des tables de routage pour 500 pairs

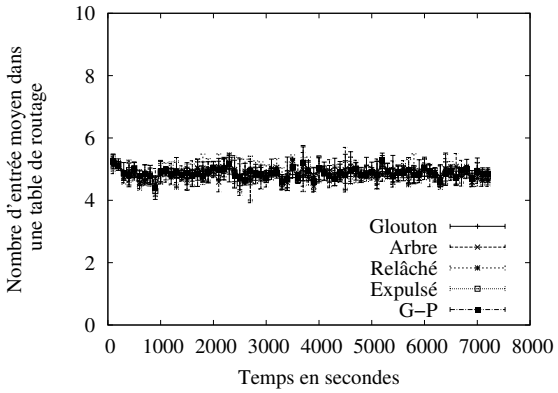

FIGURE 2.30 – Taille des tables de routage pour 1000 pairs

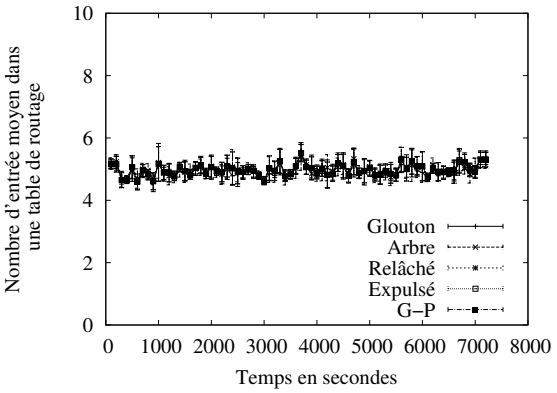

FIGURE 2.31 – Taille des tables de routage pour 2000 pairs

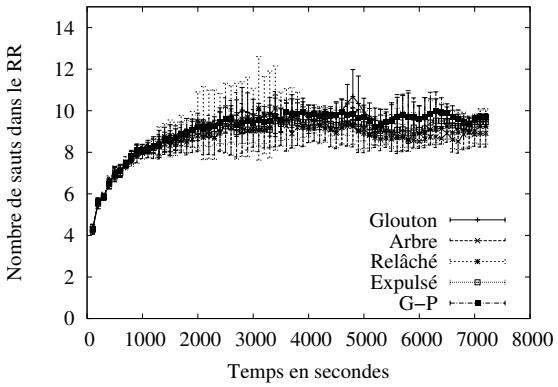

FIGURE 2.32 – Longueur des chemins de routage empruntés par les paquets pour 1000 pairs

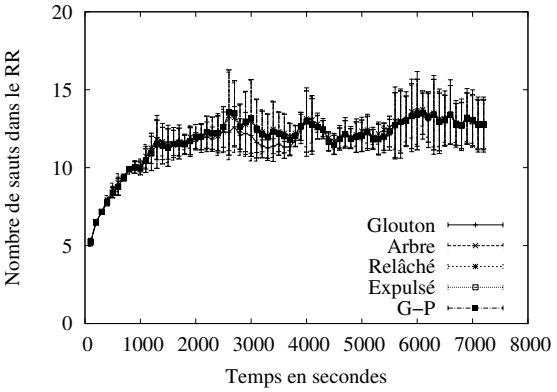

FIGURE 2.33 – Longueur des chemins de routage empruntés par les paquets pour 2000 pairs

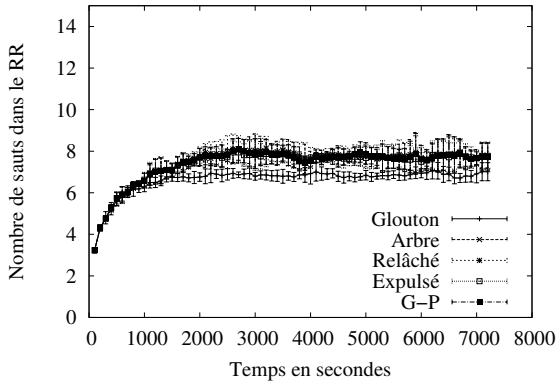

FIGURE 2.34 – Longueur des chemins de routage empruntés par les paquets pour 500 pairs

La figure 2.34 nous montre la longueur des chemins sur lesquels nous évaluons le taux de délivrance des paquets. La longueur des chemins représente le nombre de sauts fait par les paquets. Les algorithmes permettant aux paquets de s'extraire des minima locaux ont bien des chemins plus long que le routage glouton classique.

Étirement des chemins pour toutes les tailles de réseaux

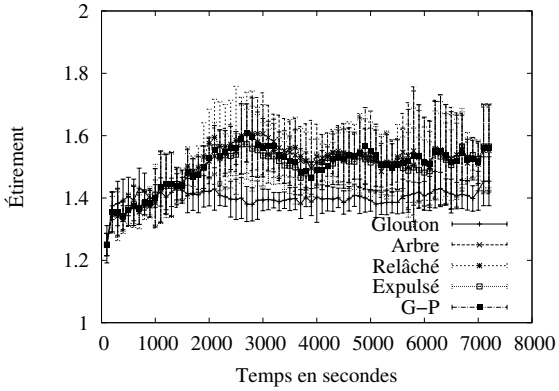

FIGURE 2.35 – Étirement des chemins empruntés par les paquets dans un réseau de 500 pairs

Les étirements obtenus par les différents algorithmes de routage sont inférieurs à 2 mais malheureusement pas de 1 (figure 2.35). Le routage glouton classique obtient l'étirement le plus bas contrairement à l'algorithme G-P, cela semble cohérent étant donné que les algorithmes que nous disons alternatifs doivent s'extraire de minima locaux donc faire des détours.

Les étirements présentés par les figures 2.36 et 2.37 restent raisonnable. Malheureusement, pour 2000 pairs l'étirement semble tendre vers 2. Aussi, pour 1000 pairs, nous

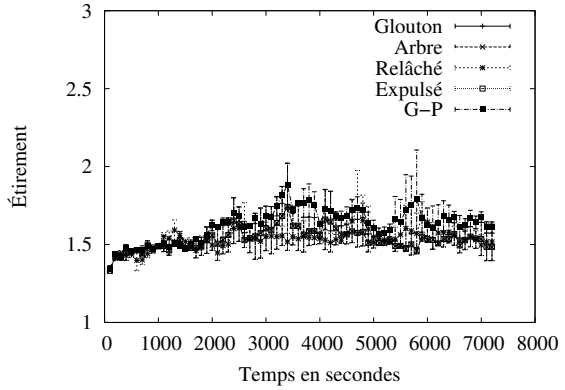

FIGURE 2.36 – Étirement des chemins empruntés par les paquets dans un réseau de 1000 pairs

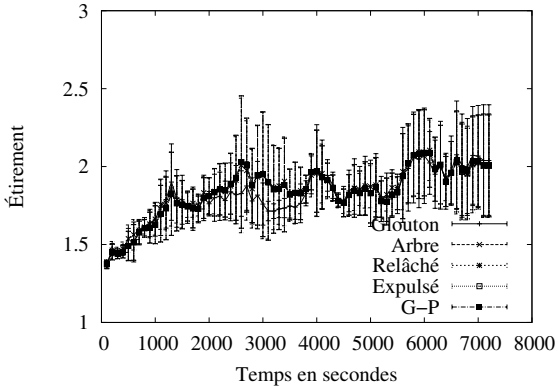

FIGURE 2.37 – Étirement des chemins empruntés par les paquets dans un réseau de 2000 pairs

observons de fortes oscillations, évidemment plus d'exécutions lisseraient les courbes. Enfin, c'est l'algorithme *relâché* qui présente l'étirement le plus faible.

Nombre d'octets en transit pour le trafic de contrôle pour toutes les tailles de réseaux

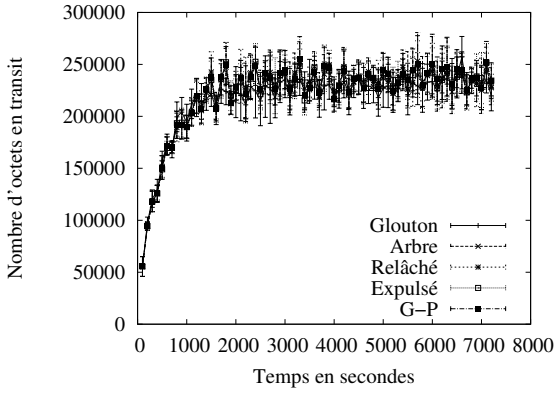

FIGURE 2.38 – Nombre d'octets en transit pour le trafic de contrôle dans un réseau de 500 pairs

La figure 2.38 présente le nombre d'octets échangé dans le réseau au cours du temps. Pour calculer ce nombre d'octets, nous multiplions le degré moyen des pairs à un instant t avec le nombre des pairs actifs dans le réseau au même instant t. Nous avons ensuite choisi une taille de message arbitraire, ici 100 octets, afin d'obtenir les courbes de la figure 2.38. Le nombre de messages en transit pour le trafic de contrôle à un instant t varie entre 2000 et 2500.

Avec la figure 2.39 et la figure 2.40 nous voyons le nombre d'octets que représente le trafic de contrôle entre les pairs. Clairement, il augmente en fonction du nombre de pairs.

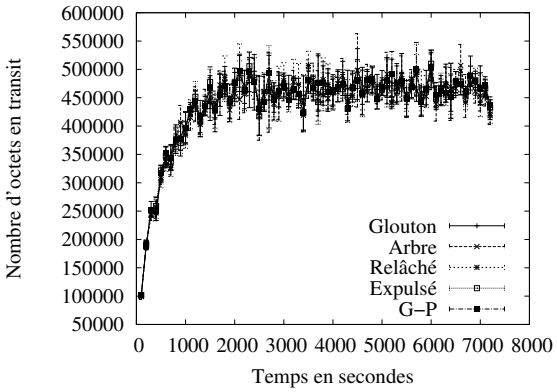

FIGURE 2.39 – Nombre d'octets en transit pour le trafic de contrôle dans un réseau de 1000 pairs

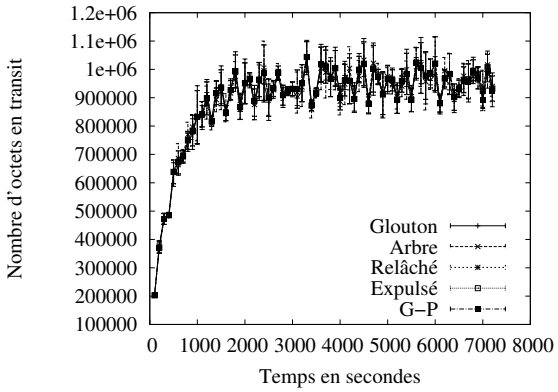

FIGURE 2.40 – Nombre d'octets en transit pour le trafic de contrôle dans un réseau de 2000 pairs

Pour 1000 pairs il requiert en moyenne entre 4500 et 5000 messages. Pour 2000 pairs environ 9000 messages sont émis pour le trafic de contrôle. Il suffit de multiplier par la taille d'un paquet de contrôle pour obtenir le nombre d'octets.

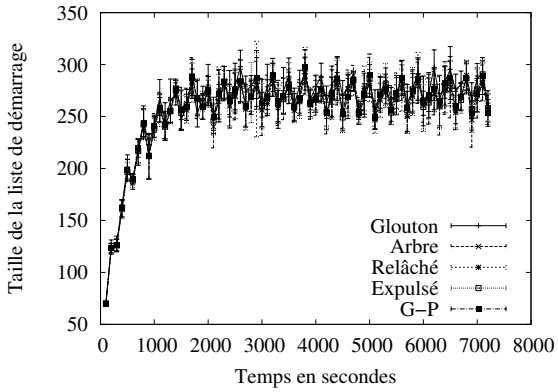

FIGURE 2.41 – Évolution de la liste de démarrage pour les pairs entrants avec un réseau de 500 pairs

La figure 2.41 montre le nombre de pairs contenu dans notre service de démarrage. Cette liste contient principalement les feuilles de l'arbre d'adressage.

2.4 L'infini en pratique

2.4.1 Problème de la précision en virgule flottante

D'un point de vue théorique, le plan hyperbolique est illimité. Néanmoins, il est nécessaire d'utiliser une représentation de ce plan (i.e une modélisation). Une propriété du

modèle de Poincaré est manquante : les distances ne sont pas préservées. Si nous observons le modèle de Poincaré d'un point de vu extérieur, les distances sont plus petites que la réalité (à l'intérieur du plan). La raison est que le modèle est une représentation du plan hyperbolique dans le plan euclidien. En effet, plus les points sont proches de la limite euclidienne du cercle unité, représenté par le disque de rayon 1, plus ils sont éloignés en réalité. Le disque unité ouvert autour du centre $(0, 0)$ est l'ensemble des points dont le module complexe est inférieur à 1.

En informatique, la représentation d'un tel espace mathématique est contraint par la précision du type flottant utilisé, typiquement un *double*. Ce qui impose implicitement un niveau de précision pour les calculs à effectuer. Alors, comment pouvons nous déterminer le nombre maximal de régions que nous pouvons créer afin d'affecter aux nœuds un couple de coordonnées uniques ? C'est un problème de précision arithmétique lorsque nous atteignons la précision maximale permise par le calcul en virgule flottante. Les calculs usuels obéissent aux standards IEEE 754 lesquels déterminent la représentation binaire de la virgule flottante. Néanmoins, l'arithmétique en virgule flottante peut aussi être implémenté avec une longueur de chiffre significatif ajustée en fonction de ces besoins, c'est l'*Arbitrary Precision Arithmetic (APA)*. Comme la capacité d'adressage est suffisante avec la précision en virgule flottante standard et que l'utilisation de l'APA est complexe, nous gardons la représentation classique du type *double* pour distinguer les points.

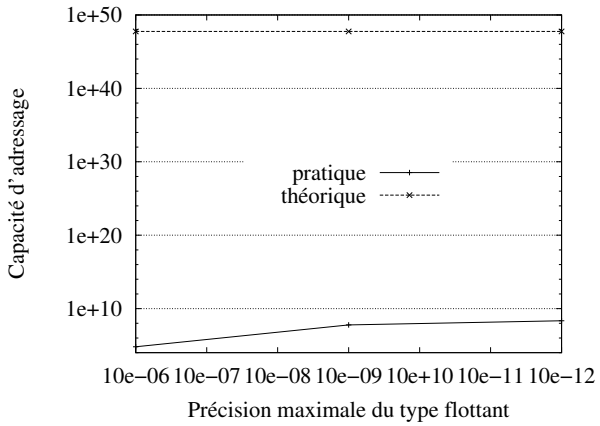

FIGURE 2.42 – Capacité d'adressage en fonction du seuil de précision de la virgule flottante

Pour déterminer le nombre maximal d'adresses calculables, nous plongeons un arbre de degré 32 et d'une profondeur égale à 32 sauts. Pour le calcul des adresses des nœuds, la précision maximale varie entre 10^{-6} et 10^{-12}. La figure 2.42 montre l'écart entre le nombre d'adresses en théorie et en pratique. L'augmentation de la capacité d'adressage *pratique* se réduit entre 9 et 12 décimales parce que moins de points peuvent être créés. En outre, nous avons observé qu'après 12 décimales les chiffres ne sont plus significatifs à cause des erreurs d'arrondies des fonctions de la librairie. Le seuil atteint par la capacité d'adressage est égale à 2.246E+08 avec les paramètres ci-dessus. Cette limite supérieure est induite par les restrictions mémoires des logiciels et par le matériel faisant les calculs (i.e. 4GB de RAM et autant pour le fichier de *swap*).

2.4.2 Influence du degré sur le nombre de coordonnées

La précision est fixée à 10^{-12} et la profondeur à 32 sauts. Le degré de l'arbre évolue de 4 à 256. La figure 2.43 montre que la capacité d'adressage théorique augmente linéairement en fonction du degré, ce qui est attendu. En revanche, la capacité d'adressage *pratique* montrée sur la figure 2.44 ne suit pas la même tendance et reste entre 2 fois 10^8 et 3 fois 10^8.

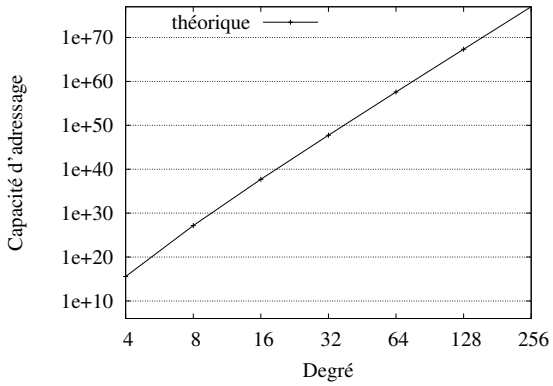

FIGURE 2.43 – Influence du degré sur le nombre d'adresses théoriques

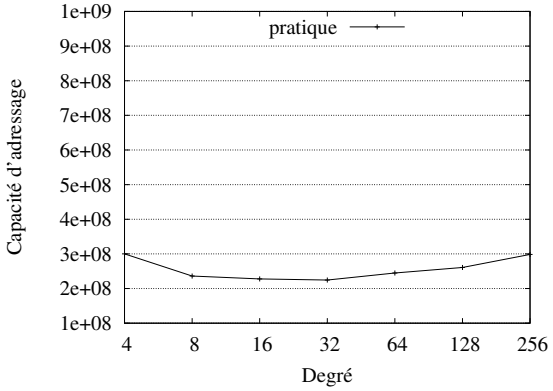

FIGURE 2.44 – Influence du degré sur le nombre d'adresses pratiques

Pour un degré supérieur à 32 le gain est faible comparé à l'ordre de grandeur observé figure 2.43. Comme nous créons un réseau recouvrant, il est possible de régler finement le degré pour améliorer la capacité d'adressage. Aussi, nous garantissons au moins 200 millions d'adresses utilisateurs quel que soit le degré q choisi et malgré le problème de la virgule flottante.

2.4.3 Influence de la profondeur sur le nombre de coordonnées

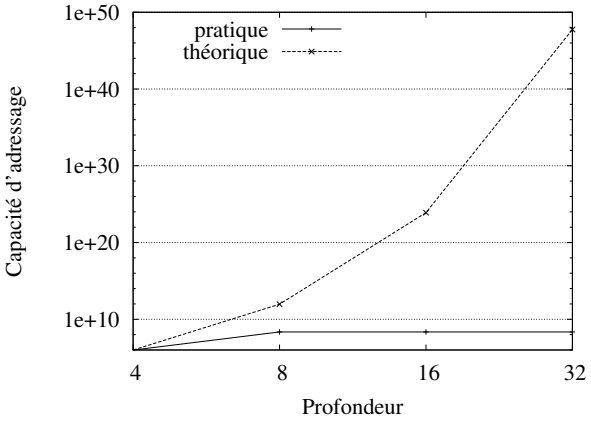

FIGURE 2.45 – Influence du degré sur le nombre d'adresses

Pour cette analyse, la précision reste inchangée, le degré de l'arbre est fixé à 32 et la profondeur évolue de 4 à 32. Sur la figure 2.45, la capacité d'adressage théorique croît exponentiellement quand la profondeur augmente. Cela correspond aux caractéristiques normales de $\mathbb{H}^2$ mais du fait de cette relation entre la profondeur et le nombre d'adresses, en pratique, la capacité d'adressage atteint les limites de la machine à 2.246E+08 avec une profondeur de seulement 8. En effet, la limite du disque est rapidement atteinte. Nous rappelons que le pavage est construit à l'infini avec un $p - gones$ régulier. Or dans $\mathbb{H}^2$, il existe une distance d à partir de laquelle les demi-plans sont deux à deux disjoints (i.e les côtés du q-gones sont tangents, voir les lignes rouges de la figure 1.5). De ce fait et en considérant les paramètres précédents, les feuilles de l'arbre peuvent atteindre la limite du disque après seulement 7 sauts (i.e. une profondeur de 7).

2.5 Conclusion

Dans le cas d'un réseau dynamique, nous proposons un algorithme distribué d'adressage et des solutions pour maintenir le routage. Pour ce faire, nous donnons des solutions permettant aux pairs de surmonter les minima locaux. Les solutions évaluées présentent de bonnes performances. Le taux de délivrance des paquets est supérieur à 99%, les tables de routage pour un réseau de 2000 pairs n'ont pas changé avec la taille du réseau et l'étirement reste inférieur à 2 pour toutes les tailles de réseaux.

Nous avons abordé la problématique du routage point-à-point issu d'un plongement glouton dans un contexte distribué et dynamique et proposons des mécanismes pour faire face aux défaillances des pairs engendrant des défaillances du routage. La solution que nous présentons est intéressante pour quelques raisons simples :

- La taille des adresses est constante (les coordonnées des pairs dans l'espace hyperbolique). Dans le cas du routage compact la taille est poly-logarithmique ;
- La taille des tables de routages est proportionnelle aux degrés des pairs (chaque pair stocke ses coordonnées et celles de ses voisins). Avec le routage compact et dans le cas d'un adressage dépendant de la topologie ou indépendant, la taille est de poly-logarithmique à polynomial ;
- Le taux de délivrance des paquets est égal à 100% avec une faible dégradation dans les réseaux dynamiques (avec une technique de maintien) contre, théoriquement, 100% pour le routage compact ;
- L'étirement est équivalent avec le routage compact.

L'algorithme de routage distribué que nous présentons n'a pas de temps de convergence si ce n'est celui de l'algorithme de maintenance. Durant ce laps de temps, se sont les algorithmes de routage alternatifs qui prennent le relais. Alors que des algorithmes de planarisation requièrent le maintien de structure couvrante engendrant de nombreux échanges de message et un temps de convergence. D'autres solutions consistent à stocker les informations dans des pairs spéciaux mais leurs mises à jour régulière a aussi un coût. De plus, la centralisation diminue la robustesse du réseau.

Un des points faibles de la méthode présentée dans [Kle07] réside dans le parcours global du réseau pour trouver le plus haut degré. Au contraire nous ne présupposons aucune connaissance sur le réseau existant. D'autant qu'un parcours global en milieu dynamique

est rapidement obsolète. C'est pourquoi, dans le cadre de ce mémoire nous proposons des adresses à usage unique. Un espace de nommage plat dont les noms sont indépendants de la localisation physique. En revanche, nous prenons l'hypothèse que la racine n'est jamais détruite. Cette hypothèse peut être considérée comme très contraignante cependant un algorithme permettant l'émergence d'une nouvelle racine n'est pas nécessairement adapté ou plus efficace. Mais nous y reviendrons par la suite car ces considérations rentrent dans le cadre de l'étude d'un algorithme de maintenance.

Enfin, le coût de notre trafic de contrôle est simple à évaluer. Chaque pair présent dans le réseau émet à intervalle régulier un nombre de paquets égal à son degré. En moyenne le degré est égal à 5 pour un réseau de 500,1000 ou 2000 pairs, multiplié par le nombre de pairs moyen au même instant cela nous donne le nombre de messages en transit dans le réseau P2P. La bande passante consommée sera donc égale au nombre de messages en transit multiplié par leur fréquence d'envoi puis multiplié par leur taille.

Chapitre 3

Réseaux virtuels et leurs Applications

Sommaire

Deux mots caractérisent l'environnement dans lequel nous travaillons : dynamique et hétérogène. Ces contraintes sont inhérentes aux appareils nomades. De ce fait, le choix de l'architecture P2P sur laquelle repose le RR est important (pair central ou pair léger et lien permanent ou potentiel).

Les réseaux recouvrants sont adaptés pour tester de nouvelles idées s'appuyant sur le paradigme pair-à-pair. L'intérêt de ce paradigme est le comportement global du réseau qui émerge des interactions locales entre les pairs. Chaque pair peut se connecter à un sous ensemble d'autre pais de manière aléatoire et peuvent ensuite changer de voisin. Cela engendre des phénomènes difficilement prédictibles, c'est pour cette raison que ce modèle de réseau est étudié au travers de simulation plutôt qu'au travers d'un formalisme qui à notre connaissance n'existe pas.

Le reste du chapitre est organisé comme suit. La section 3.1 rappel quelques concepts bien connus des réseaux P2P. La section 3.2 discute du passage à l'échelle dans le cadre de la redistribution de vidéo dans les réseaux P2PTV. La section 3.3 propose des mécanismes basés sur l'algorithme de sélection de pair pour améliorer la croissance du réseaux P2PTV. La section 3.4 présente des résultats d'évaluation obtenus en simulant l'architecture d'un des plus grands producteurs de P2PTV en Europe Zattoo.

3.1 Réseaux virtuels ou recouvrants (RR)

Dans l'introduction, nous avons introduit les deux termes qui permettent de catégoriser les utilisateurs. En effet, certains participants au réseau P2P éteignent leur client une fois qu'ils ont obtenu le contenu recherché. Ces utilisateurs sont nommés les *free-riders*. Si le réseau de distribution de contenu survit c'est grâce aux *seeders* qui sont en quel que sorte les serveurs du réseau. La volatilité des *free-riders* créée des perturbations plus ou moins importantes selon les applications. Pour de gros fichiers les pairs vont rester connectés plus longtemps. C'est d'ailleurs pour ce type d'échange que l'aspect collaboration est le plus important. C'est pourquoi, dans des protocoles comme Bitorrent les participants sont mis à contribution dès les premiers octets reçus. Alors que pour des flux vidéo comme la télévision, l'aspect collaboration est nécessaire non pas pour trouver un contenu mais plutôt pour satisfaire l'ensemble des participants car ils sont interdépendants. Or les participants ont tendance à changer régulièrement de flux vidéo créant des perturbations pour l'ensemble du réseau de diffusion [CMC+11].

Quel que soit la partie applicative certains problèmes sont communs à l'ensemble des réseaux recouvrants. La coordination de l'interconnexion des pairs n'est pas aisée puisque distribuée. Cette coordination intervient à deux moments lorsque les pairs entre dans le réseau (démarrage) ou lorsque des voisins du pair disparaissent (maintenance). L'interconnexion des pairs influence la structure du réseau (la topologie) et la topologie permet, lorsqu'elle est connue, de choisir la meilleure stratégie pour transférer l'information. En revanche, la topologie ne fournit pas d'information sur les caractéristiques des liens entre les pairs. La structure permet seulement de garantir de borner le nombre de sauts (e.g. CHORD). Ainsi les réseaux structurés permettent de maîtriser la topologie du réseau virtuel afin de garantir un acheminement efficient. De plus, ces systèmes permettent d'utiliser un routage proche de celui du graphe dont ils s'inspirent, diminuant ainsi le trajet des messages donc le transite dans le réseau. Cependant, les architectures structurées sont coûteuses à maintenir et résistent mal à la volatilité.

Toutefois, les architectures P2P non structurées tel que Gnutella ont aussi un désavantage majeur puisque le nombre de messages qu'elles utilisent pour une recherche augmente en suivant une loi de puissance, c'est le prix de l'inondation. C'est pourquoi, ce mode de recherche ne passe pas à l'échelle.

3.1.1 Architecture P2P : rappel

Architecture centralisée : Dans cette architecture un serveur central gère les contenus, la recherche, l'insertion d'informations mais le transfert de contenu effectué de ma-

nière direct entre les utilisateurs. Le serveur central est indispensable au réseau (e.g. Napster). C'est pourquoi c'est son talon d'Achille.

Architecture décentralisée : L'ensemble des pairs qui forment l'application distribuée ont un rôle et une importance identique dans le réseau. Ce type d'architecture inclut deux catégories ou stratégies pour la gestion des connexions. Les réseaux structurés comme CAN, Chord, Tapestry, Pastry, Symphony, etc. adoptent une stratégie d'organisation des pairs afin d'assurer et de trouver les services et contenus. La plupart de ces systèmes utilisent une table de hachage distribué (THD ou en anglais DHT) pour indexer les données. L'intérêt de cette méthode est de borner le nombre de messages nécessaires à une recherche quel que soit le nombre de pairs présents dans le réseau. Le nombre de messages doit croître de manière logarithmique en la taille du système. Pour les réseaux non structurés comme Gnutella le nombre de messages nécessaires à une recherche peut croître de manière exponentielle car le nombre de messages est proportionnel au nombre de pairs dans le réseau. La méthode de transmission des messages pour Gnutella est l'inondation or ce n'est pas une technique passant à l'échelle. Bien que plus robuste aux disparitions de pairs les architectures distribuées requiert des protocoles et des algorithmes plus complexe à mettre en œuvre pour une recherche d'information efficace et/ou une maintenance de la structure.

Architecture décentralisée hiérarchique (ADH) : Étant donnée la difficulté de mise en œuvre des architectures totalement décentralisé une dernière solution a émergé. Une ADH comme eDonkey ou KaZaA (système FastTrack) est un hybride des solutions précédentes où certains pairs retrouvent un rôle clef. Ces pairs disposent normalement d'une puissance de calcul et d'une bande passante plus importante afin de jouer le même rôle qu'un serveur central mais de manière locale. En général, ces super-pairs sont interconnectés et reprennent des protocoles et des algorithmes employés dans les architectures décentralisées. Cette architecture hybride reste assez robuste puisque c'est un compromis entre les deux solutions précédentes. En fait, tout dépend de la gestion des super-pairs (i.e. fixe et/ou figé ou souple) ; autrement dit si les pairs du réseau peuvent, selon les besoins du système, devenir des super-pairs et inversement alors la robustesse et la flexibilité du réseau est accrue.

En conclusion, les architectures centralisées dépendent d'un centre névralgique alors que plus de décentralisation apporte plus de robustesse à la perte des pairs. Cependant, nous n'avons pas traité de sécurité (réputation, authentification, etc.).

3.1.2 Problème du démarrage

Ce processus modélise la naissance et la croissance du réseau virtuel car au départ seul le créateur du réseau existe. Ensuite d'autres participants viennent joindre le réseau. Le problème du démarrage consiste à trouver le meilleur mécanisme pour intégrer les pairs entrants au réseau car intégrer les pairs signifie qu'ils sont connectés. Or pour pouvoir se connecter ils doivent échanger des messages, c'est tout l'intérêt de la couche transport mais surtout les pairs entrant doivent avoir au moins une adresse à contacter. La solution la

plus simple est de disposer d'un service centralisé. Ce service peut aussi être hiérarchique sur le modèle de DNS.

Le challenge dans de tels systèmes est de déterminer comment découvrir d'autres pairs dans le réseau de manière distribuée. Car une fois qu'un pair entrant dispose d'un point d'entrée, il peut potentiellement parcourir tout le réseau. Décentraliser cette étape est insoluble avec l'architecture Internet actuelle. En effet, un pair entrant requiert une adresse de la couche transport et, à notre connaissance, aucune fonction n'est capable de générer la plage d'adresse des participants (pairs actifs). Autrement dit, il faut un point d'entrée quel que part (e.g. avec un traqueur à la Bitorrent). Nous pouvons déplacer le problème dans les couches du modèle OSI jusqu'aux réseaux sociaux mais il reste le même, trouver un point d'entrée. Cependant une méthode alternative est de permettre au pair entrant de laisser une adresse IP sur un serveur sécurisé afin qu'ils soient par la suite contactés par des membres du réseau.

3.2 Redistribution dans un réseau recouvrant P2PTV

Les applications de diffusion de flux vidéo permettent d'aborder un grand nombre de problématiques liées aux déploiements de réseaux P2P dans l'Internet. Pour les flux réseaux, le but est de maximiser la capacité des liens par lesquels transit le flux vidéo.

La télévision transmise par le protocole Internet (IPTV) présente de nombreuses possibilités tant pour les utilisateurs que pour les fournisseurs de services. Elle a attiré de manière significative les entreprises et depuis quelques années, la communauté de la recherche. Parmi les architectures de diffusion de l'IPTV, le mécanisme de diffusion TV pair-à-pair est intéressant du fait de sa facilité à être déployé. Cependant, la question de comment une application P2PTV peut supporter un nombre croissant d'utilisateurs n'a pas été étudié en profondeur. Dans ce chapitre, nous essayons d'aborder ces questions en étudiant la possibilité du passage à l'échelle et le facteur efficacité d'un réseau P2P diffusant de la vidéo temps réel. À travers l'utilisation de données fournie par un système de production P2PTV, nous avons fait des simulations dont les résultats montrent qu'il y a encore des difficultés à surmonter avant qu'un réseau P2PTV puisse devenir largement utilisé.

Avec l'accroissement de la largeur de bande passante et les continuelles améliorations dans les technologies de compression vidéo, le service de télévision par internet (IPTV) connaît ces derniers temps, un maintien de sa croissance. Pour réaliser le service IPTV dans l'internet d'aujourd'hui, le mécanisme de diffusion basé sur le pair-à-pair (P2P) est considéré comme une option intéressante de part sa facilité à être déployé et pour le faible besoin de bande passante.

Dans un réseau P2P construit pour diffuser de la TV sur IP, les clients accèdent aux flux vidéos en se connectant au serveur de diffusion ou à un autre client présent sur le réseau. Le serveur de diffusion génère un flux vidéo depuis l'encodage du signal TV capturé par le satellite. Après avoir rejoint le réseau, les clients peuvent contribuer avec leur débit ascendant (*upload*) en transférant le flux vidéo entrant aux clients se joignant au réseau par la suite. Pour permettre une utilisation plus efficace de la capacité d'envoi des clients, les flux vidéos sont distribués en morceaux (*chunks*) (e.g. [BPL+08]) ou en sous-flux (e.g.

[XLKZ07, CJW09]) via un réseau recouvrant. Les *chunks* sont des segments vidéo divisés en fonction du temps, tandis que les sous-flux sont issus de la division spatiale du flux originel (e.g., couches en H.264 SVC). Les *chunks* ou les sous-flux sont soit poussés (*push*) par les clients amonts pour le transfert aux autres clients avals, soit tirés *pull* par les clients en aval depuis les clients en amonts. Cela dépend du protocole de partage utilisé dans le réseau P2P. Dans la distribution *pull-driven*, les clients cherchent et récupèrent un flux individuel, ils sont dans une approche opportuniste, tandis que dans l'approche *push-driven*, un client établit une connexion virtuelle de transmission vers un client, et continue à recevoir les données tirées depuis un transmetteur jusqu'à la terminaison de la connexion. Dans des travaux récents [ZZY07], la distribution avec la technique du *pull-push* est présentée comme plus efficace que son homologue *pull-based*.

Comparé au P2P traditionnel partageant des données ou à la diffusion (*streaming*) progressive pour la vidéo à la demande (VoD), l'expérience d'optimisation de l'utilisateur final dans un environnement P2PTV est une tâche non-triviale à cause des contraintes de délai et de mémoire tampon. De plus, les contraintes de capacité d'envoi et les comportements d'attrition inhérents aux clients participants peuvent s'ajouter à la difficulté de réaliser un système de diffusion extensible. Motivé par notre étude précédente sur l'adaptabilité opérationnelle du P2PTV [SMC+09], nous explorons l'impact de la distribution sur la bande passante d'envoi et les algorithmes de sélection d'un pair sur la performance du système. Dans notre étude, nous nous concentrons sur une architecture de *streaming* utilisant le *push-driven* et des sous-flux. Nous exécutons des simulations avec les données fournies par un système de production implémentant une telle architecture.

Dans cette section, nous étudions l'impact de la bande passante du client sur le passage à l'échelle d'un système P2P (i.e, le nombre maximal des utilisateurs qui peuvent être supportés par le système). Nous ne considérons pas de pair volatile. Nous ajouterons des pairs dynamiques dans les sections suivantes. Nous définissons le ratio du flux entrant qui peut être redistribué à d'autres pairs comme un *facteur de redistribution* et le nommons k partout dans ce chapitre. Le facteur de redistribution k peut varier de 0 à l'infini et peut être fractionné, selon la capacité de connexion du pair. Si $k = 1$, cela signifie que le pair peut redistribuer le flux complet. Si $k = 2$, cela signifie que le pair redistribue deux copies du flux. Si $k = 0,5$, cela signifie que le pair redistribue seulement la moitié du flux en raison des contraintes de la bande passante de sa connexion. Des valeurs fractionnaires sont possibles parce qu'un flux est divisé dans des sous-flux multiples, qui permettent à un pair de redistribuer seulement un sous-ensemble du flux. Techniquement, le facteur de redistribution peut être évalué comme le débit ascendant (*upload bitrate*) divisé par le débit descendant (*download bitrate*) à n'importe quel instant donné. Dans cette section, nous supposons que le facteur de redistribution reste constant dans le temps pour n'importe quel pair.

3.2.1 Analyse théorique

D'abord, nous étudions d'un point de vue théorique l'impact du facteur de redistribution lors du passage à l'échelle. Comme le réseau recouvrant est un graphe acyclique orienté, nous pouvons définir la profondeur d'un pair dans le réseau recouvrant comme le nombre des liens du réseau recouvrant entre lui-même et les pairs connectés à la source.

Si nous supposons que tous les pairs ont un même k, nous pouvons déterminer le nombre maximal de pairs seulement en utilisant le facteur de redistribution k, la profondeur du graphe n et la capacité de la source C car le nombre total de pairs est égal à la somme d'une suite géométrique. En effet, si U_n représente le nombre de pairs à la profondeur n et si $U_0 = C$, alors nous avons $U_n = k \times U_{n-1}$. Le nombre total de pairs est ainsi égal à :

$$S_n = \sum_{i=0}^{n} U_i = C \times \frac{1 - k^{n+1}}{1 - k} \tag{3.1}$$

Le nombre maximum de pair pouvant se connecter au système dépendront de la valeur de k :

- Si $k < 1$ alors le nombre de pairs convergera vers $\sum U_n \to \frac{C}{1-k}$ et donc le système ne passera pas à l'échelle en taille.
- Si $k = 1$ alors le nombre de pairs divergera de manière linéaire ; $\sum U_n \to +\infty$, et donc le système passera à l'échelle en taille.
- Si $k > 1$ alors le nombre de pairs divergera de manière exponentielle ; $\sum U_n \to +\infty$, et donc le système passera à l'échelle en taille.

Considérons un cas plus général où un niveau l est donné, chaque pair i a une capacité d'envoi (ul_i) et de téléchargement (dl_i) alors le facteur k pour ce niveau l contenant p pairs serait :

$$k_l = \frac{\sum_{i=1}^{p} ul_i}{\sum_{i=1}^{p} dl_i} \tag{3.2}$$

Nous définissons un pair inactif qui ne téléchargera pas la totalité du flux. Nous définissons un pair hyperactif comme ayant $k > 1$. Dans le cas le plus général, le facteur k de redistribution est exprimé comme ceci :

$$k = \frac{U_a \times (1 - R_i - R_h) + U_i \times R_i + U_h \times R_h}{D_a \times (1 - R_i - R_h) + D_i \times R_i + D_a \times R_h} \tag{3.3}$$

- D_a : nombre de téléchargements de sous-flux par pair actif
- U_a : nombre d'envois de sous-flux par pair actif
- D_i : nombre de téléchargements de sous-flux par pair inactif
- U_i : nombre d'envois de sous-flux par pair inactif
- U_h : nombre d'envois de sous-flux par pair hyperactif
- R_i : ratio de pairs inactifs vs. pairs totaux
- R_h : ratio de pairs hyperactif vs. pairs totaux

Dans un monde idéal, nous aurions $k \geq 1$ pour tous les pairs et donc nous n'aurions pas de réseau recouvrant de taille limitée (bien que cela puisse créer un réseau recouvrant trop profond si k est à peine plus grand que 1). En réalité, nous avons un ensemble de facteurs de redistribution assez variés.

- k peut être inférieur à 1
  - Profiteurs $(k = 0)$
  - Pairs bloqué par sécurité $(k = 0)$
  - Pairs avec une connexion peu performante $(k < 1)$
- k peut être supérieur à 1
  - Sources auxiliaires, i.e., super pairs $(k >> 1)$

– Pairs avec une connexion performante ($k > 1$)
– Pair inactif, i.e., pair qui ne regarde pas le flux mais redistribue un sous ensemble du sous-flux du flux ($k > 1$ pour plusieurs sous-flux).

3.2.2 Les pairs inactifs

Pour rehausser un facteur k bas, une solution est de tirer partie des pairs inoccupés qui peuvent redistribuer le flux en arrière plan. Ne regardant pas le flux activement, les pairs inactifs pourraient recevoir seulement une fraction du flux complet et redistribuer les copies de cette fraction, agissant ainsi comme un multiplicateur. La figure 3.1 montre le réseau P2PTV en fonction du pourcentage de pairs inactifs. Le nombre de sous-flux contenus dans le flux complet est mis à 16 comme dans le réseau recouvrant de Zattoo [SMC+09]. Il est présumé que des pairs actifs ont $k = 0.5$ et que des pairs inactifs redistribuent un sous-flux (1/16e du flux) quatre fois (ainsi $k = 4$). Pour permettre une même disponibilité des différents sous-flux dans le réseau, chaque pair actif/inactif choisit aléatoirement le(s) sous-flux à redistribuer. Les courbes de la figure 3.1 sont obtenues en utilisant la formule 3.3.

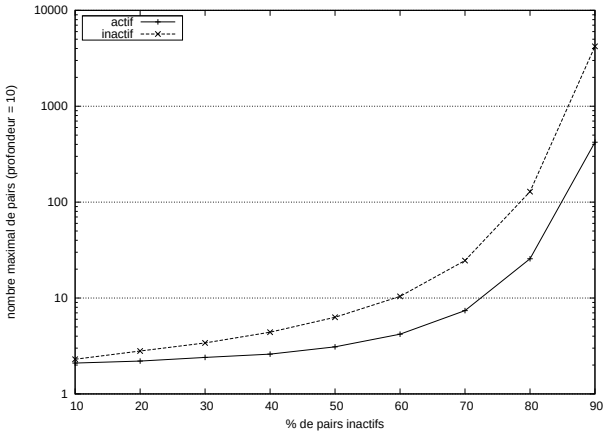

FIGURE 3.1 – Capacité du réseau vs. quantité de pairs inactifs

La capacité du réseau est égale au nombre maximal de pairs du réseau recouvrant divisé par le nombre maximal des pairs qui peuvent se connecter à la source directement. Si la capacité du réseau est égale à 1, cela signifie qu'un pair peut seulement se connecter à la source, mais pas aux autres pairs. Si la capacité du réseau est égale à 10, cela signifie que le réseau recouvrant peut supporter dix fois plus de pairs que la capacité de la source. La figure montre à fois les capacités du réseau actif et du réseau inactif. La courbe étiquetée "actif" est le nombre de pairs dans le réseau avec $k = 0.5$. La courbe étiquetée "inactif" est la somme du nombre de pairs inactifs et de pairs actifs. Elle montre que des pairs inactifs aident très peu à améliorer la capacité du réseau quand des valeurs réalistes du nombre de pairs inactif sont prises (<30%). Les pairs actifs d'un réseau recouvrant ont besoin d'un grand nombre de pairs inactifs pour s'étendre (c'est-à-dire avoir une capacité > 10). Ainsi l'utilisation de pairs inactifs n'aide pas le système à passer à l'échelle de manière significative.

3.2.3 Les pairs hyperactifs

Une autre solution pour rehausser un facteur k bas est d'utiliser des pairs hyperactifs pour redistribuer des flux multiples. Un pair hyperactif est un pair qui regarde un flux et peut redistribuer des copies multiples du flux, ou au moins un flux complet plus quelques sous-flux supplémentaires. La figure 3.2 montre la capacité du réseau de *streaming* en fonction du pourcentage de pairs hyperactifs. Semblable au cas précédent, un flux complet consiste en 16 sous-flux. Il est présumé que des pairs actifs régulièrement ont un $k = 0.5$ et que des pairs hyperactifs ont un $k = 1.5$. Les courbes de la figure 3.2 sont aussi obtenues par utilisation des formules 3.3. La courbe étiquetée "hyperactif" est le nombre de pairs dans le réseau avec $k = 1.5$. La courbe étiquetée "actif" est la somme des nombres de pairs hyperactifs et de pairs actifs. Nous pouvons voir qu'il y a une divergence à 60% des pairs hyperactifs. Ainsi l'utilisation de pairs hyperactifs n'aide pas le système à passer à l'échelle de manière significative. Avec des pairs exclusivement hyperactifs la capacité du réseau croît exponentiellement alors que s'il y a des pairs moins hyperactifs, ils doivent avoir des valeurs de k plus grande que 1.5 pour atteindre la divergence.

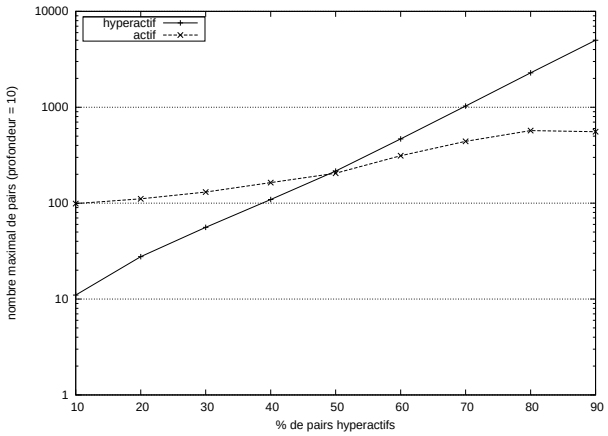

FIGURE 3.2 – Capacité du réseau vs. quantité de pairs hyperactifs

3.3 Efficacité de l'algorithme de sélection d'un pair dans un réseau P2PTV

Quand un nouveau pair se connecte au réseau recouvrant, il exécute son propre algorithme de sélection pour choisir les pairs auxquels se connecter. Si la sélection du pair est faite aléatoirement, le réseau résultant peut devenir inefficace pour deux raisons. D'abord, le réseau peut devenir trop profond et non assez large, engendrant ainsi de gros retards de lecture. Deuxièmement, le reseau peut subir beaucoup d'attrition, entraînant de nombreuses interruptions pour les utilisateurs.

3.3.1 Les effets sur le réseau recouvrant

Nous voudrions voir s'il est possible d'améliorer l'efficacité d'un réseau recouvrant P2PTV en utilisant des algorithmes de sélection de pairs qui fusionnent des paramètres

dynamiques tels que la capacité d'envoi, des temps de sessions, des distances entre des pairs et des positions de profondeur dans le réseau recouvrant. L'idée derrière cette proposition est qu'en plaçant des pairs plus stables et avec une bande passante à forte capacité tout près de la source, on puisse rendre le réseau recouvrant plus efficace. L'algorithme de sélection de pair est primordial pour influencer l'évolution du réseau car nous pouvons contrôler où les pairs se connecteront dans le réseau. Si l'algorithme de sélection de pair réussit à placer des pairs stables près de la source, cela doit réduire les réorganisations complètes du réseau recouvrant. Aussi, si cet algorithme réussit à placer des pairs avec une bande passante à forte capacité près de la source, cela devrait augmenter la capacité du réseau recouvrant en gardant une profondeur raisonnable. Nous étudierons cette approche par la simulation dans la sous-section 3.4.

3.3.2 Les effets sur un pair

En réalité, les pairs changent de canaux et redistribuent seulement un minimum de flux dans le but d'économiser leur bande passante d'envoi. De tels comportements sont nuisibles à la croissance d'un réseau recouvrant et donc au passage à l'échelle. Ainsi quelle peut être la motivation pour le pair d'un P2PTV de se comporter autrement ? Une solution est d'employer le mécanisme "d'un prêté pour un rendu" avec des vidéos de qualités différentes comme proposé dans [LSP+07]. Si différentes qualités de vidéos ne sont pas disponibles, une autre solution que nous proposons permet de négocier la position de profondeur dans le réseau recouvrant. Si le pair a une bonne connexion et reste dans un même canal, il deviendra progressivement plus proche de la source (mieux servi). À la fin, les pairs avec une connexion stable et performante seraient exposés à moins de variations de leur connectivité et jouiraient de la TV en temps réel. Ils seraient donc naturellement incités à rester dans un même canal. Il a été montré dans [HLL+07] que le retard de lecture dans un système P2PTV à grande échelle varie entre 30 et 90 secondes. Ainsi, la position d'un pair pourrait être une bonne motivation pour inciter un pair stable et performant à rester sur un même canal. Dans les résultats de simulation montrés dans la sous-section 3.4, nous utilisons ces critères (e.g., le temps de session et la capacité d'envoi) pour voir si le réseau fonctionne mieux quand les pairs sont choisis en fonction de tels critères.

3.4 Simulation d'un réseau recouvrant P2PTV

Après avoir étudié quelques aspects théoriques d'un réseau recouvrant P2PTV et proposé des mécanismes d'amélioration, nous présentons maintenant des résultats obtenus par simulation et mettons en évidence l'impact des algorithmes de sélection de pair sur la performance d'un système de P2PTV.

3.4.1 Paramètres et métriques de la simulation

Notre code de simulation met en œuvre la technique des sous-flux *push-driven* de Zattoo basé sur une architecture de P2PTV [CJW09] et a été exécuté sur le logiciel de

simulation réseau *nem*[1]. Une carte Internet de 4.2k-nœuds a été utilisée comme topologie sous-jacente. Nous supposons qu'un réseau donné distribue une chaîne de télévision, et nous ne prenons pas en considération les variations quotidiennes de la taille du canal. Chaque test dure 12 heures et seulement les 6 dernières heures sont analysées. Après la 6e heure, nous sommes dans un régime stable d'état. Chaque valeur de résultat est la moyenne de 30 tests et les valeurs d'écart type sont fournies.

Pour toutes nos simulations, nous utilisons les mêmes paramètres que dans notre travail précédent [SMC+09], qui était basé sur l'analyse de 9.6M de sessions enregistrées par le P2PTV Zattoo pendant une période de 2 semaines. Selon les données de session, la fonction de distribution cumulative (CDF) du facteur de redistribution k suit une distribution exponentielle. 50% des pairs peuvent redistribuer moins de 50% du flux complet (.i.e., $k < 0.5$). 82% des pairs peuvent redistribuer moins que le flux complet (i.e., $k < 1$). La capacité de la connexion des pairs individuels est assignée pour que la distribution résultante devienne la même que la distribution empirique. Le type de NAT d'un pair, qui détermine son accessibilité au réseau, est aussi considéré dans la distribution empirique des types de NAT énoncés dans [SMC+09]. Finalement, le temps d'inter arrivée dans la session et le temps de session sont tous deux des distributions exponentielles équivalentes énoncées dans [SMC+09]. Nous faisons varier la taille du canal du mode famine (proche de la capacité source) au mode profusion (plusieurs fois supérieur à la capacité source). La table 3.1 montre les paramètres de maintien saisie.

TABLE 3.1 – paramètres de simulations

Paramètres	Valeurs
Nombre de sous-flux par flux	16
Capacité de la source	50 clients
Période de recherche	2 sec
Maximum de tentatives de recherche	2
Taille maximale de la liste de pairs candidats	40 pairs
Nombre de simulations par scénario	30

Pour évaluer le fonctionnement d'un système de P2PTV, nous étudions les métriques suivantes :

1. Ratio du temps de visualisation (en % = 100 x temps de visualisation du pair / durée de vie du pair) : le temps de visualisation moyen des pairs qui ont terminé pendant cette période.
2. Pourcentage de pairs rejetés : nombre total des pairs qui ne pouvaient pas se connecter au réseau recouvrant P2PTV pendant la période donnée divisé par le nombre de nouveaux pairs par période.

1. http ://www.labri.fr/perso/magoni/nem/

3. Nombre moyen d'interruptions par pair : nombre total d'interruptions vidéo vu pour tous les pairs dans la période donnée divisé par le nombre de nouveaux pairs par période.

Nous analysons ces métriques de production en variant le nombre de nouveaux pairs arrivant par heure, qui définie la charge mis sur la source et le réseau P2P.

3.4.2 Résultats de simulation

Cette section présente nos résultats de simulation. Un pair essayant de se connecter à d'autres pairs pour obtenir tous les sous-flux nécessaires est appelé un pair orphelin. Il envoie des messages de recherche pour découvrir d'autres pairs, envoie des messages d'appairage pour se connecter aux pairs disponibles et obtient finalement le flux complet de leur part. Un pair qui est capable, et disposé à offrir une partie de tous les sous-flux demandés par un pair orphelin est appelé pair adoptif. Un pair adoptif est un pair qui a positivement répondu au message de recherche d'un orphelin. Après avoir eu de multiples réponses positives des pairs adoptifs candidats, les orphelins doivent choisir le pair auquel ils devraient envoyer un message pour se connecter. Nous évaluons les algorithmes de sélection des pairs suivants, décrit dans la sous-section 3.3 :

- **Aléatoire** : Un pair orphelin essaie de se connecter à un pair adoptif sélectionné aléatoirement
- **Local** : Un pair orphelin essaie de se connecter à un pair adoptif proche (la distance étant mesurée en sauts)
- **Débit ascendant (*upload*)** : Un pair orphelin essaie de se connecter à un pair adoptif proposant le meilleur envoi en amont (mesuré sur le nombre de sous-flux)
- **Durée d'activité** : Un pair orphelin essaie de se connecter à un pair adoptif ayant la plus longue durée d'activité.

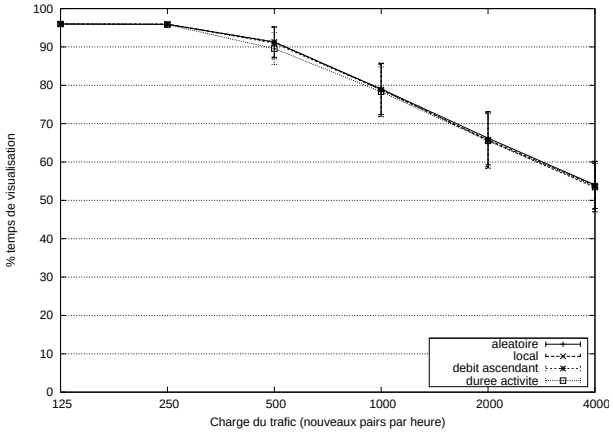

FIGURE 3.3 – Temps de visualisation moyen vs. charge de trafic

La figure 3.3 montre le temps moyen de visualisation d'un pair en fonction de la charge du trafic. Nous pouvons voir que les effets des divers algorithmes sur le temps d'observation ne font pas beaucoup de différence comparé à une sélection aléatoire. Bien qu'il y ait 7% de différence entre le plus mauvais et les meilleurs algorithmes pour une

charge de trafic de 2000 nouveaux pairs par heure et 5% de différence pour une charge de 4000, ces valeurs ne sont pas énormes. Ce résultat quelque peu inattendu induit la relative importance des paramètres de niveau utilisateur telle que la capacité d'envoi et le temps de session sur les paramètres systèmes tels que les algorithmes de sélection de pair.

La figure 3.4 montre que le pourcentage moyen de pairs rejeté est fonction de la charge de trafic. En comparant avec les résultats précédents, la diminution en temps de visualisation est principalement due aux pairs étant rejetés du réseau. Seulement un petit pourcentage est causé par les interruptions dues aux déconnexions et reconnexions de pairs.

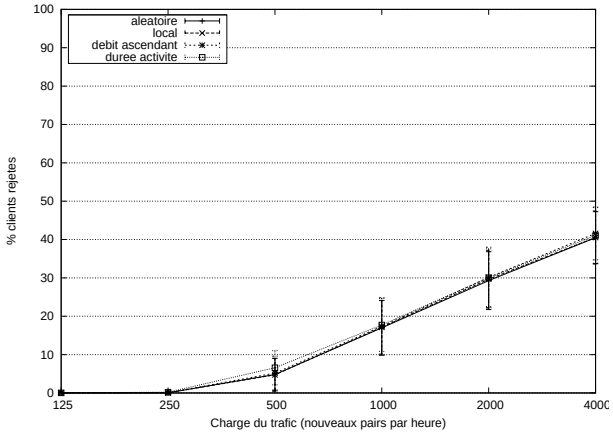

FIGURE 3.4 – Pourcentage moyen de pairs rejetés vs. charge de trafic

La figure 3.5 montre le nombre moyen d'interruptions par pair en fonction de la charge de trafic. Nous observons que le nombre moyen d'interruptions par pair augmente progressivement quand le nombre de pairs augmente. Quand le nombre de nouveaux pairs augmente, le réseau recouvrant grandit et le taux d'attrition moyen devient plus grand et ainsi mène à plus de connexions et reconnexions. Cependant, quand le réseau recouvrant devient saturé par les nouveaux pairs, ces nouveaux pairs ne peuvent pas réussir à se joindre au réseau et sont rejetés. Ainsi, le nombre de connexions et de reconnexions ne grandit pas tant, car les pairs rejetés ne contribuent pas significativement à ce nombre. Cependant, le nombre total de pairs augmente toujours, et donc la proportion n'augmente plus désormais.

Deux remarques peuvent expliquer les résultats des algorithmes alternatifs par rapport à l'algorithme de sélection aléatoire. D'abord, les sessions sont de durée limitée. Environ 50% des sessions sont plus brèves que 1,5 minute. Cela crée beaucoup d'attrition qui rendent l'évolution du réseau recouvrant difficile à contrôler au cours du temps. Deuxièmement, le facteur de redistribution tend fortement vers de petites valeurs. 50% des pairs ont une capacité d'envoi inférieure à 50%. Ainsi, les pairs avec de long temps session peuvent avoir une faible capacité d'envoi et ne pas être trop utilisés. En somme, ces deux facteurs pèsent beaucoup plus lourdement sur le temps de visualisation que les algorithmes de sélection divers. Quand la charge de trafic est grande, l'algorithme qui fonctionne mieux que la sélection aléatoire est l'algorithme « local ». Le résultat des simulations avec chacune des métriques définies prouve que l'algorithme de sélection n'affecte

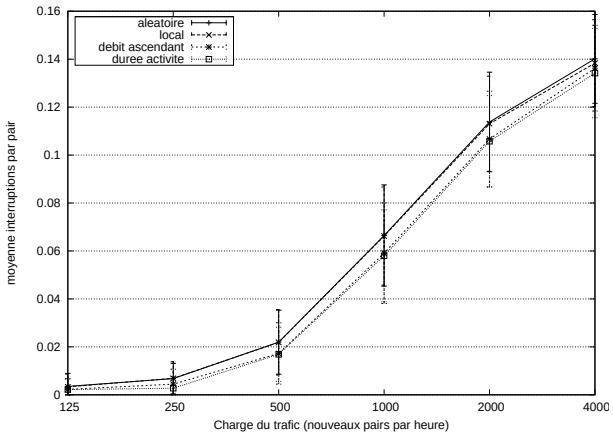

FIGURE 3.5 – Nombre moyen d'interruptions par pair vs. charge de trafic

pas beaucoup le système.

3.5 Conclusion

Dans cette section nous avons étudié le comportement d'un réseau P2PTV *push-driven* par sous-flux. Nous avons conduit une étude théorique concernant l'influence du facteur de redistribution et nous avons montré que le passage à l'échelle du système dépend fortement de la capacité d'envoi disponible au sein des pairs. Nous avons aussi montré que l'utilisation de pairs inactifs aussi bien que celle de pairs hyperactifs n'est pas une solution fondamentale pour que le système passe à l'échelle. Étant donné que le processus de sélection a une influence directe sur la structure et l'efficacité du réseau recouvrant, nous avons proposé une amélioration possible du processus de sélection des pairs. Pour évaluer son impact potentiel, nous avons effectué des expériences de simulation afin de mesurer l'efficacité des divers algorithmes de sélection des pairs dans un système P2PTV soumis une forte charge. Nous avons effectué notre simulation avec des paramètres réalistes tirés des données de 9.8M de sessions rassemblées par le système de P2PTV Zattoo. Nos résultats montrent que le facteur de redistribution et le temps de session ont un effet profond sur la capacité maximale du réseau recouvrant P2PTV et que les divers algorithmes de sélection jouent un rôle relativement marginal pour le passage à l'échelle du système. Nos travaux futurs seront destinés à l'étude d'autres algorithmes de sélection de pair ainsi qu'à l'étude de l'impact de la taille des mémoires tampons et des paramètres de recherche de pairs sur le rendement global d'un système P2PTV.

Conclusion

Les réseaux recouvrants basés sur le paradigme P2P et construits au-dessus des protocoles de la couche transport sont des solutions idéales pour tester de nouvelles applications. Chaque pair du réseau recouvrant exécute les mêmes algorithmes et participe au bon fonctionnement du réseau. Pour communiquer, ces pairs peuvent utiliser une information de localisation telle que la position géographique ou géométrique dans un espace comme cela est proposé dans les travaux présentés au chapitre 1.

La contribution de ce mémoire est de répondre au besoin croissant d'un algorithme qui avec une quantité d'information réduite, soit capable d'acheminer l'information de manière décentralisée dans les réseaux filaires ou non filaires, d'Internet aux réseaux *ad hoc*. En effet, de nombreux schémas de routage distribué existants reposent sur des algorithmes gloutons comme expliqué au chapitre 1. La plus simple des techniques de routage basée sur des coordonnées virtuelles est gloutonne, dans le sens où les nœuds transfèrent toujours le paquet au voisin qui est le plus proche de la destination finale en utilisant, par exemple, la métrique hyperbolique. Cependant, le routage dans les graphes dynamiques reste un enjeu de part la volatilité et l'hétérogénéité des participants car les solutions sont souvent trop complexes pour être déployées (i.e. complexe en termes de coûts de calcul, de stockage et de bande passante). Notre solution a le mérite d'être simple et de présenter des performances encourageantes pour un déploiement. En effet, l'étirement des chemins est inférieur à 2, le taux de réussite est supérieur à 99% et la taille des tables de routages reste minime comparativement à la taille du réseau. Les performances ont été évaluées selon les critères définis pour les schémas de routage compact. En résumé, notre solution permet de répondre aux besoins suivants :

- Elle permet de construire incrémentalement des tables de routage compact en utilisant uniquement les informations locales.
- Elle garantit que la maintenance des tables de routage cause peu de trafic de contrôle.
- Elle fournit des mécanismes pour s'adapter aux changements topologiques et donc pallie les erreurs de routage temporaire.

Enfin, nous avons analysé les performances des routes obtenues à partir de notre schéma de routage sur plusieurs topologies (e.g. de type Internet ou aléatoire)

Dans ce mémoire, nous avons aussi présenté dans l'état de l'art quelques propriétés du plan hyperbolique que nous utilisons pour créer notre algorithme distribué d'adressage et de routage glouton présenté au chapitre 2. En effet, nous proposons un algorithme de plongement donnant accès à un routage glouton basé sur des coordonnées virtuelles prises dans le plan hyperbolique. Notre algorithme distribué permet d'étiqueter n'importe quel type d'entité et de router entre ces entités à l'aide de ces étiquettes. De plus, il respecte les bornes théoriques définies par le routage compact qui permet un passage à l'échelle,

c'est-à-dire la taille des tables de routage qui ne varie pas en fonction de la taille du réseau. Enfin, la taille des adresses utilisées est fixe.

Pour répondre aux contraintes introduites par la mobilité nous proposons des mécanismes afin de pallier les défaillances du routage en environnement dynamique. Nous évaluons, avec les métriques habituelles du routage, l'efficacité du routage glouton dans des réseaux statiques. Puis, dans le cas dynamique, nous évaluons les différentes méthodes qui permettent d'assurer la délivrance des messages. Nous montrons par les simulations, au chapitre 2, que notre service de routage présente des performances encourageantes en atteignant un taux de délivrance supérieur à 99% et un étirement inférieur à 2. De plus, les simulations montrent que notre système P2P nécessite peu de trafic de contrôle. Enfin, dans le chapitre 3, nous avons étudié l'influence de l'algorithme de sélection des pairs au travers d'un cas particulier, celui des réseaux P2P destinés à distribuer des flux vidéos. Nous avons vu avec étonnement que dans les situations basées sur des données d'un réseau P2PTV réel (Zattoo), l'influence de l'algorithme de sélection de pairs n'est que marginale par rapport à la durée de session et la capacité du débit montant.

3.6 Perspectives de recherche

Dans l'Internet le problème de la racine de l'arbre d'adressage peut trouver une solution technique. Par exemple, nous pouvons imaginer un cœur de réseau qui se substitue à la racine pour plus de robustesse mais cela demanderait un consensus général sur l'architecture Internet. C'est pourquoi, une telle conception demanderait une stratégie de déploiement en accord avec les opérateurs. Cependant, ce n'est pas la première raison qui nous a poussé à étudier un nouveau service d'adressage et de routage.

Table de hachage distribuée

De manière générale, une architecture P2P structurée basée sur le concept de DHT peut être divisée en trois couches : l'application utilisant la DHT, la DHT elle-même et le routage par clef. Les solutions existantes tels que Chord, CAN, Pastry, Tapestry, Kademlia, Koorde, Viceroy, etc, appelées architectures P2P structurées, sont nombreuses [LCP+05]. En effet, les réseaux recouvrants P2P structurés par une table de hachage répartie sont connus et étudiés depuis plus d'une décennie. Cependant, ces solutions offrent un service de routage peu performant dans les réseaux dynamiques. Et lorsque le service de routage est performant la topologie du réseau doit suivre un certain motif engendrant d'autres contraintes liées à la maintenance et à la croissance du réseau.

Si nous avons mené l'étude, au travers d'un réseau virtuel P2P, d'un nouveau service d'adressage et de routage pour les membres d'un réseau P2P dans un environnement dynamique, c'est pour un objectif précis. Simplement, la finalité de ce service est de permettre la mise en œuvre d'une table de hachage distribué destinée à gérer des sessions utilisateur. En effet, notre architecture finale vise à supporter la gestion de sessions virtuelles afin de favoriser la flexibilité des communications dans l'Internet. L'application P2P sera capable de lier de manière souple, les utilisateurs, les applications et les périphériques, lors d'une communication grâce à la gestion / restauration des sessions. Donc notre routage basé

sur les coordonnées virtuelles du plan hyperbolique peut être aussi utilisé comme couche de routage pour une table de hachage distribuée.

Dans notre cas, chaque pair a un nom qui est un identifiant au niveau applicatif lequel permet aux membres de se contacter. C'est pourquoi, les correspondances nom-adresse des pairs doivent être stockées afin de lier l'espace de nommage des pairs (considéré comme l'espace des clefs) et l'espace d'adressage (considéré comme l'espace des adresses). Le premier espace d'adressage est plutôt destiné aux utilisateurs alors que le deuxième est plutôt destiné aux machines. Avec ces éléments, nous pouvons construire une couche de routage par clef pour lier le nom et l'adresse d'un nœud membre afin de fournir une flexibilité des communications, comme expliqué dans des travaux antérieurs [CBM10].

Développement de l'outil de simulation

Les premières simulations ont été réalisées avec le simulateur *nem*. Afin de définitivement valider nos résultats nous avons décidé d'implanter notre système P2P à l'aide de l'environnement de simulation *OMNeT++*. L'environnement d'*OMNeT++* nous a permis de créer un outil pour l'évaluation d'algorithmes distribués avec le modèle pair-à-pair. Ainsi, il est possible de simuler, dans un environnement dynamique régi par des lois statistiques, des algorithmes et des protocoles sur des groupes de périphériques hétérogènes. De plus, le grain pour la représentation des équipements permet de spécifier des piles de protocoles différentes, si nécessaire. Dans notre cas, nous utilisons deux couches. La couche de plus haut niveau qui exécute les algorithmes du RR et la couche inférieure qui offre un routage basé sur l'algorithme de Dijkstra.

Cependant, après quelques années consacrées à l'étude des réseaux P2P, le verdict n'est pas en faveur des simulateurs. En effet, le problème de la simulation pour ce type de système de façon générale, est qu'elle ne rend compte du fonctionnement que d'une implantation particulière. Or une erreur s'immisce très facilement. Donc comment ensuite savoir si le comportement observé provient d'une erreur, du comportement normal ou de l'interaction entre les pairs (émergence). Toutefois, les simulateurs sont indispensables mais ils ne permettent pas de généraliser les conclusions, sauf si comme nous l'avons fait les résultats sont associés à des résultats théoriques. Dans notre cas, nous utilisons les travaux réalisés dans le cadre du routage compact. De plus, nous n'avons pas trouvé de formalisme permettant d'exprimer les protocoles nécessaires au bon fonctionnement d'un système P2P. D'autant plus, qu'un formalisme permettrait la vérification et la description claire des différents protocoles.

Vers l'IoT et les DTN

Ensuite une autre étude intéressante sera de pousser la volatilité des participants à son maximum, à la limite entre les réseaux tolérants aux délais (DTN) et les réseaux dits dynamiques. En effet, l'accroissement de la mobilité mène dans le pire des cas à un réseau tolérant aux délais où des communications flexibles sont indispensables. En outre, le comportement nomade des utilisateurs reliés à Internet nous annonce que la flexibilité des communications n'est plus une option. Enfin, dans la continuité de l'Internet des objets certains travaux, ayant déjà quelques années, ne traitent plus des réseaux en

terme d'interconnexion de nœuds physiques mais ne considèrent plus qu'un réseau virtuel d'Objet-à-Objet [BKMR07].

Bibliographie

[AGM+04] Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Noam Nisan, and Mikkel Thorup. Compact name-independent routing with minimum stretch. *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 20–24, 2004.

[AK08] Bruno Abrahao and Robert Kleinberg. On the internet delay space dimensionality. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 157–168, 2008.

[AT08] Moitra Ankur and Leighton Tom. Some results on greedy embeddings in metric spaces. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 337–346, Washington, DC, USA, 2008. IEEE Computer Society.

[Bar09] Albert-László Barabási. Scale-free networks : A decade and beyond. *Science Magazine*, 325 :412–413, 2009.

[BB03] Albert-László Barabási and Eric Bonabeau. Scale-free networks. *Scientific American*, 288 :60–69, 2003.

[BDPBR10] Lélia Blin, Shlomi Dolev, Maria Potop-Butucaru, and Stephane Rovedakis. Fast self-stabilizing minimum spanning tree construction using compact nearest common ancestor labeling scheme. In *Proceedings of 24th International Symposium on Distributed Computing (DISC)*, pages 480–494, 2010.

[BFCW09] Hitesh Ballani, Paul Francis, Tuan Cao, and Jia Wang. Making routers last longer with viaggre. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, pages 453–466. ACM, 2009.

[BGPB+09] Lelia Blin, Maria Gradinariu, Potop-Butucaru, Stephane Rovedakis, and Sébastien Tixeuil. A new self-stabilizing minimum spanning tree construction with loop-free property. In *Proceedings of 23rd International Symposium on Distributed Computing (DISC)*, pages 407–422, 2009.

[BGPB+10] Lelia Blin, Maria Gradinariu, Potop-Butucaru, Stephane Rovedakis, and Sébastien Tixeuil. Loop-free super-stabilizing spanning tree construction. In *Proceedings of 12th International Symposium (SSS)*, pages 50–64, 2010.

[Bia08] Samuel Bianco. Quelques aspects de géométrie hyperbolique plane. Master's thesis, École Polytechnique Fédérale de Lausanne (EPFL), semestre de printemps 2008.

[BKMR07] Olivier Beaumont, Anne-Marie Kermarrec, Loris Marchal, and Etienne Rivière. Voronet : A scalable object network based on voronoi tessellations. In *Parallel and Distributed Processing Symposium, IPDPS*, pages 1–10, 2007.

[BM06] Alan F. Beardon and David Minda. The hyperbolic metric and geometric function theory. In *International Workshop on Quasiconformal Mappings And Their Applications*, 2006.

[Bon09] Olivier Bonaventure. Scaling the internet with lisp. Trilogy Future Internet summer school, 08 2009. Tutorials.

[BPL+08] L. Bracciale, F. Lo Piccolo, D. Luzzi, S. Salsano, G. Bianchi, and N. Blefari-Melazzi. A Push-based Scheduling Algorithm for Large Scale P2P Live Streaming. In *Proceedings of the 4th International Telecommunication Networking Workshop on QoS in Multiservice IP Networks*, pages 1–7, 2008.

[CBM10] Cyril Cassagnes, David Bromberg, and Damien Magoni. An overlay architecture for achieving total flexibility in internet communications. In *International Conference on Advanced Information Technologies for Management*, 2010.

[CC09] Andrej Cvetkovski and Mark Crovella. Hyperbolic embedding and routing for dynamic graphs. In *Proceedings of Infocom*, 2009.

[CCK+06] Matthew Caesar, Tyson Condie, Jayanthkumar Kannan, Karthik Lakshminarayanan, and Ion Stoica. Rofl : routing on flat labels. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications*, 2006.

[CCN+06] Matthew Caesar, Miguel Castro, Edmund B. Nightingale, Greg O'Shea, and Antony Rowstron. Virtual ring routing : network routing inspired by dhts. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications*, pages 351–360, 2006.

[Che08] William Chen. *Introduction to Complex Analysis*. 2008.

[Cis12] Cisco. Visual networking index : Forecast and methodology. White papers, May 2012.

[CJW09] Hyunseok Chang, Sugih Jamin, and Wenjie Wang. Live Streaming Performance of the Zattoo Network. In *Proceedings of Internet Measurement Conference*, November 2009.

[CMC+11] Cyril Cassagnes, Damien Magoni, Hyunseok Chang, Wenjie Wang, and Sugih Jamin. Scalability and efficiency of push-driven p2ptv systems. *Journal of Communications Software and Systems*, 7(3) :93–103, 2011.

[CTBM11] Cyril Cassagnes, Telesphore Tiendrebeogo, David Bromberg, and Damien Magoni. Overlay addressing and routing system based on hyperbolic geometry. In *IEEE symposium on Computers and Communications (ISCC)*, pages 294–301, 2011.

[DC09] Amit Dvir and Niklas Carlsson. Power-aware recovery for geographic routing. In *Proceedings of the conference on Wireless Communications and Networking*, pages 2851–2856. IEEE Press, 2009.

[DGK10] Benoit Donnet, Bamba Gueye, and Mohamed Ali Kaafar. A survey on network coordinates systems, design, and security. *IEEE Communication Surveys and Tutorial*, 12 :488–503, 2010.

[DMM08] John Day, Ibrahim Matta, and Karim Mattar. Networking is ipc : a guiding principle to a better internet. In *Proceedings of the 2008 ACM CoNEXT Conference*, pages 67 :1–67 :6, 2008.

[Emm08] Philippe Emmanuel. *Géométrie des surfaces hyperboliques*. PhD thesis, Institut de Mathématiques de Toulouse, 2008.

[Fin87] Gregory G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical report, University of Southern California, 1987.

[FLV08] Pierre Fraigniaud, Emmanuelle Lebhar, and Laurent Viennot. The inframetric model for the internet. In *27th IEEE International Conference on Computer Communications (INFOCOM)*, 2008.

[FPW09] Roland Flury, Sriram Pemmaraju, and Roger Wattenhofer. Greedy routing with bounded stretch. In *28th Annual IEEE Conference on Computer Communications (INFOCOM), Rio de Janeiro, Brazil*, April 2009.

[Fra94] Paul Francis. *Addressing in Internetwork Protocols*. PhD thesis, University College London, september 1994.

[GGK+04] Ramakrishna Gummadi, Ramesh Govindan, Nupur Kothari, Young Jin Kim, Brad Karp, and Scott Shenker. Reduced state routing in the internet. In *Proceedings of the Third ACM Workshop on Hot Topics in Networks (HotNets-III)*, 2004.

[Ghy06] Étienne Ghys. Poincaré et son disque. *L'héritage scientifique de Henri Poincaré*, Belin, 2006.

[Goo08] Michael T. Goodrich. Succinct greedy graph drawing in the hyperbolic plane. In *In Proc. 16th Int. Symp. Graph Drawing*, 2008.

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

[JCAC+10] Loránd Jakab, Albert Cabellos-Aparicio, Florin Coras, Damien Saucez, and Olivier Bonaventure. Lisp-tree : A dns hierarchy to support the lisp mapping system. *IEEE Journal on Selected Areas in Communications*, 2010.

[jKGkS05a] Young jin Kim, Ramesh Govindan, Brad karp, and Scott Shenker. Geographic routing made practical. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, 2005.

[jKGKS05b] Young jin Kim, Ramesh Govindan, Brad Karp, and Scott Shenker. On the pitfalls of geographic face routing. In *Proceedings of DIALM-POMC*, 2005.

[JST+09] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies (CoNEXT)*, pages 1–12. ACM, 2009.

[Kar01] Brad Karp. Challenges in geographic routing : Sparse networks, obstacles, and traffic provisioning. In *the DIMACS Workshop on Pervasive Networking*, 2001.

[KBB+04] Thomas Karagiannis, Andre Broido, Nevil Brownlee, Kimberly Claffy, and Michalis Faloutsos. Is p2p dying or just hiding ? In *Proceedings of Globecom - Global Internet and Next Generation Networks*. IEEE, 2004.

[KCFB07] Dmitri Krioukov, Kimberly Claffy, Kevin Fall, and Arthur Brady. On compact routing for the internet. *SIGCOMM Computer Communication Review*, 37 :41–52, 2007.

[KK77] Leonard Kleinrock and Farok Kamoun. Hierarchical routing for large networks. *Performance Evaluation and Optimization Computer Networks*, 1(3) :155–174, 1977.

[Kle00] Jon Kleinberg. The small-world phenomenon : An algorithmic perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*. Also published a short version in Nature, 406 :845, 2000.

[Kle07] Robert Kleinberg. Geographic routing using hyperbolic space. In *Proceedings of the 26th Annual Joint Conference of the INFOCOM*, pages 1902–1909. IEEE Computer and Communications Societies, 2007.

[KPBV09] Dmitri Krioukov, Fragkiskos Papadopoulos, Marián Boguñá, and Amin Vahdat. Greedy forwarding in scale-free networks embedded in hyperbolic metric spaces. *SIGMETRICS Performance Evaluation Review - workshop on MAthematical performance Modeling and Analysis*, 37(2) :15–17, 2009.

[KR09] James Francis Kurose and Keith W. Ross. *Computer Networking : A Top-Down Approach*. Addison-Wesley, 2009.

[KSU99] Evangelos Kranakis, Harvinder Singh, and Jorge Urrutia. Compass routing on geometric networks. In *In Proceedings of the 11th Canadian Conference on Computational Geometry*, 1999.

[KWZ03] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 267–278, 2003.

[KWZZ03] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing : of theory and practice. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 63–72, 2003.

[LBSB09] Cristian Lumezanu, Randy Baden, Neil Spring, and Bobby Bhattacharjee. Triangle inequality variations in the internet. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, pages 177–183, New York, NY, USA, 2009. ACM.

[LCP+05] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys & Tutorials, IEEE*, 7 :72–93, 2005.

[LLM07] Ben Leong, Liskov, and Morris. Greedy virtual coordinates for geographic routing. In *Proceedings of the IEEE International Conference on Network Protocols*, pages 71–80, 2007.

[LPMS09] Lukic, Pavkovic, Mitton, and Stojmenovic. Greedy geographic routing algorithms in real environment. In *Proceedings of the 5th International Conference on Mobile Ad-hoc and Sensor Networks*, pages 86–93, 2009.

[LRP95] John Lamping, Ramana Rao, and Peter Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '95, pages 401–408. ACM Press/Addison-Wesley Publishing Co., 1995.

[LSP+07] Zhengye Liu, Yanming Shen, Shivendra S. Panwar, Keith W. Ross, and Yao Wang. Using Layered Video to Provide Incentives in P2P Live Streaming. In *Proceedings of the ACM Workshop on Peer-to-peer streaming and IP-TV*, pages 311–316, 2007.

[LW06] Cong Liu and Jie Wu. Swing : Small world iterative navigation greedy routing protocol in manets. In *Proceedings of the 15th International Conference on Computer Communications and Networks*, pages 339–350, 2006.

[Mag05] Damien Magoni. Network topology analysis and internet modelling with nem. *International Journal of Computers and Applications*, 27(4) :252–259, 2005.

[May06] Petar Maymounkov. Greedy embeddings, trees, and euclidean vs. lobachevsky geometry, 2006.

[MCK11] Victoria Manfredi, Mark Crovella, and Jim Kurose. Understanding stateful vs stateless communication strategies for ad hoc networks. In *Proceedings of the 17th annual international conference on Mobile computing and networking*, pages 313–324, 2011.

[Mey08] David Meyer. The locator identifier separation protocol (lisp). *The Internet Protocol Journal*, 11 :0, 2008.

[MHL+11] Péter Mátray, Péter Hága, Sándor Laki, István Csabai, and Gábor Vattay. On the network geography of the internet. In *Proceedings of the 30th Annual Joint Conference of the INFOCOM*, pages 126–130, 2011.

[Miq00] Alexandre Miquel. Un afficheur générique d'arbres à l'aide de la géométrie hyperbolique. In *Journées francophones des langages applicatifs (JFLA)*, pages 49–62, 2000.

[MM06] William Ma and David Minda. Geometric properties of hyperbolic geodesics. In *International Workshop on Quasiconformal Mappings And Their Applications*, 2006.

[MRSR10] Nathalie Mitton, Tahiry Razafindralambo, and David Simplot-Ryl. Position-based routing in wireless ad hoc and sensor networks. In *Theoretical Aspects of Distributed Computing in Sensor Networks*, chapter 15. Springer-Verlag, 2010.

[NMF+07] An Nguyen, Milosavljevic, Qing Fang, Jie Gao, and Guibas. Landmark selection and greedy landmark-descent routing for sensor networks. In *Proceedings of the 26th IEEE International Conference on Computer Communications*, pages 661–669, 2007.

[Pan05] Pierre Pansu. Master de mathématiques fondamentales et appliquées - 2e année : Analyse, arithmétique, géométrie. Cours de Géométrie Différentielle, 2005. Chapitre 5.

[PCGC10] Yoann Pigné, Arnaud Casteigts, Frédéric Guinand, and Serge Chaumette. Construction et maintien d'une forêt couvrante dans un réseau dynamique. In *12èmes Rencontres Francophones sur les Aspects Algorithmiques de Télécommunications (ALGOTEL)*, 2010.

[PKBV10] Fragkiskos Papadopoulos, Dmitri Krioukov, Marián Boguñá, and Amin Vahdat. Greedy forwarding in dynamic scale-free networks embedded in hyperbolic metric spaces. In *Proceedings of the 29th Annual Conference on Computer Communications*, INFOCOM, pages 1–9. IEEE, March 2010.

[PR05] Christos H. Papadimitriou and David Ratajczak. On a conjecture related to geometric routing. *Theor. Comput. Sci.*, 344(1) :3–14, 2005.

[RMK+09] Venugopalan Ramasubramanian, Dahlia Malkhi, Fabian Kuhn, Mahesh Balakrishnan, and Aditya Akella. On the treeness of internet latency and bandwidth. In *Proceedings of the 11th international joint conference on Measurement and modeling of computer systems*, 2009.

[RP07] Céline Laroche (Traduction) Roger Penrose. *À la découverte des lois de l'univers - La prodigieuse histoire des mathématiques et de la physique*. Odile Jacob, 2007.

[RRP+03] Ananth Rao, Sylvia Ratnasamy, Christos Papadimitriou, Scott Shenker, and Ion Stoica. Geographic routing without location information. In *Proceedings of the 9th annual international conference on Mobile computing and networking (MobiCom)*, pages 96–108. ACM, 2003.

[Sal82] Jerome H. Saltzer. On the naming and binding of network destinations. In *International Symposium on Local Computer Networks*, pages 311–317. Also published as Internet RFC 1498, transcribed August 1993 by J. Noel Chiappa., 1982.

[Sch09] Sebastian Schnettler. A structured overview of 50 years of small-world research. *Social Networks*, 31 :165–178, July 2009.

[SCIE10] Athanassopoulos Stavros, Kaklamanis Christos, Laftsidis Ilias, and Papaioannou Evi. An experimental study of greedy routing algorithms. In *Proceedings of the International Conference on High Performance Computing and Simulation*, pages 150–156, 2010.

[SGF+10] Ankit Singla, Philip Brighten Godfrey, Kevin Fall, Gianluca Iannaccone, and Sylvia Ratnasamy. Scalable routing on flat names. In *Proceedings of the 6th International COnference (CoNEXT)*, pages 20 :1–20 :12, 2010.

[Sho78] John F. Shoch. Inter-network naming, addressing, and routing. In *Proceedings of the 17th IEEE Conference on Computer Communication Networks*, pages 72–79, 1978.

[SMC+09] Khaldoon Shami, Damien Magoni, Hyunseok Chang, Wenjie Wang, and Sugih Jamin. Impacts of Peer Characteristics on P2PTV Networks Scalability. In *Proceedings of the 28th IEEE Conference on Computer Communications – Mini-Conference*, April 2009.

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

[ST04] Yuval Shavitt and Tomer Tankel. On the curvature of the internet and its usage for overlay construction and distance estimation. In *Proceedings of the 23th Annual Joint Conference of the INFOCOM*, 2004.

[ST08] Yuval Shavitt and Tomer Tankel. Hyperbolic embedding of internet graph for distance estimation and overlay construction. *IEEE/ACM Trans. Netw.*, 16 :25–36, February 2008.

[Sti92] John Stillwell. *Geometry of surfaces*. springer-verlag, 1992.

[TAC08] Shao Tao, A.L. Ananda, and Mun Choon Chan. Greedy hop distance routing using tree recovery on wireless ad hoc and sensor networks. In *Proceedings of the International Conference on Communications*, pages 2712–2716, 2008.

[Tan04] Andrew Stuart Tanenbaum. *Les Réseaux*. Broché, 2004.

[TBD+11] Joseph Touch, Ilia Baldine, Rudra Dutta, Gregory G. Finn, Bryan Ford, Scott Jordan, Dan Massey, Abraham Matta, Christos Papadopoulos, Peter Reiher, and George Rouskas. A dynamic recursive unified internet design (druid). *Computer Networks*, 55 :919–935, 2011.

[TK84] Hideaki Takagi and Leonard Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Transactions on Communications*, 32 :246–257, 1984.

[Tut63] William T. Tutte. How to draw a graph. In *Proceedings of the London Mathematical Society*, pages 743–767, 1963.

[TZ01] Mikkel Thorup and Uri Zwick. Compact routing schemes. In *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*, pages 1–10, 2001.

[VF07] Vivek Vishnumurthy and Paul Francis. A comparison of structured and unstructured p2p approaches to heterogeneous random peer selection. In *In Proceedings of the Usenix Annual Technical Conference*, Santa Clara, CA, 2007.

[Wei99] Mark Weiser. The computer for the 21st century. *SIGMOBILE Mobile Computing and Communications Review - Special issue dedicated to Mark Weiser*, 3 :3–11, July 1999.

[WP09] Cedric Westphal and Guanhong Pei. Scalable routing via greedy embedding. In *Proceedings of the 28th Conference INFOCOM*, pages 2826–2830, 2009.

[XLKZ07] S. Xie, B. Li, G. Y. Keung, and X. Zhang. CoolStreaming : Design, Theory, and Practice. *IEEE Trans. on Multimedia*, 9(8), December 2007.

[XLPH06] Xing, Lu, Pless, and Huang. Impact of sensing coverage on greedy geographic routing algorithms. *IEEE Transactions on Parallel and Distributed Systems*, 17(4) :348–360, 2006.

[ZCZ+10] Yibo Zhu, Yang Chen, Zengbin Zhang, Xiaoming Fu, Dan Li, Beixing Deng, and Xing Li. Taming the triangle inequality violations with network coordinate system on real internet. In *Proceedings of the Re-Architecting the Internet Workshop*, 2010.

[ZZY07] Meng Zhang, Qian Zhang, and Shi-Qiang Yang. Understanding the Power of Pull-based Streaming Protocol : Can We Do Better ? *IEEE JSAC*, 25(9) :1678–1694, 2007.

ARCHITECTURE AUTONOME ET DISTRIBUÉE D'ADRESSAGE ET DE ROUTAGE POUR LA FLEXIBILITÉ DES COMMUNICATIONS DANS L'INTERNET

Résumé :

Les schémas de routage locaux basés sur des coordonnées prises dans le plan hyperbolique ont attiré un intérêt croissant depuis quelques années. Cependant, les solutions proposées sont toutes appliquées à des réseaux aux topologies aléatoires et au nombre de nœuds limité. Dans le même temps, plusieurs travaux se sont concentrés sur la création de modèle topologique basé sur les lois de la géométrie hyperbolique. Dans ce cas, Il est montré que les graphes ont des topologies semblables à Internet et qu'un routage local hyperbolique atteint une efficacité proche de la perfection. Cependant, ces graphes ne garantissent pas le taux de réussite du routage même si aucune panne ne se produit. Dans cette thèse, l'objectif est de construire un système passant à l'échelle pour la création de réseau recouvrant capable de fournir à ses membres un service d'adressage et de routage résilient dans un environnement dynamique. Ensuite, nous étudions de quelle manière les réseaux P2PTV pourraient supporter un nombre d'utilisateur croissant. Dans cette thèse, nous essayons de répondre à cette question en étudiant les facteurs de l'efficacité et du passage à l'échelle dans un système de diffusion vidéo P2P typique. Au travers des données fournies par Zattoo, producteur de réseau P2PTV, nous réalisons des simulations dont les résultats montrent qu'il y a encore des obstacles à surmonter avant que les réseaux P2P de diffusion vidéo puissent dépendre uniquement de leurs utilisateurs.

Mots-clés : Graphe Dynamique, Plongement Glouton, Pair-à-Pair, P2PTV

AUTONOMOUS AND DISTRIBUTED ARCHITECTURE FOR ADDRESSING AND ROUTING TO IMPROVE THE FLEXIBILITY OF COMMUNICATIONS IN INTERNET

Abstract :

Local routing schemes based on virtual coordinates taken from the hyperbolic plane have attracted considerable interest in recent years. However, solutions have been applied to ad-hoc and sensor networks having a random topology and a limited number of nodes. In other hand, some research has focused on the creation of network topology models based on hyperbolic geometric laws. In this case, it has been shown that these graphs have an Internet-like topology and that local hyperbolic routing achieves a near perfect efficiency. However, with these graphs, routing success is not guaranteed even if no failures happen. In this thesis, we aim at building a scalable system for creating overlay networks on top of the Internet that would provide reliable addressing and routing service to its members in a dynamic environment. Next, we investigate how well P2PTV networks would support a growing number of users. In this thesis, we try to address this question by studying scalability and efficiency factors in a typical P2P based live streaming network. Through the use of the data provided by Zattoo a production P2PTV network, we carry out simulations whose results show that there are still hurdles to overcome before P2P based live streaming could depend uniquely of their users.

Keywords : Dynamic Graph, Greedy Embedding, Peer-To-Peer, P2PTV