Maîtrise E.E.A

Mr. DERBALI Mohammed
Mr. EMBOUAZZA Fethi

ETUDE DES
RESEAUX PEER-TO-PEER

T.E.R proposé et encadré par Mr. Olivier FOURMAUX .

Année scolaire 2002-2003

Table des matières

I.Bibliographie sur les réseaux Peer-to-peer

1.1 Qu'est ce qu'un réseaux Peer-to-Peer ?

1.1.1 Introduction

L'acronyme P2P est apparue bien avant le phénomène marketing de l'apparition à tout va des acronymes liés aux " nouveaux " modèles économique / technologique du marché Internet : B2B, B2C, C2C, etc. Mais le réseau des réseaux, bien avant de représenter un marché avec toute la valeur ajoutée qu'il peut (difficilement) générer, désignait tout autre chose auparavant, à savoir essentiellement un outil de travail. Or, qui dit travail dans un environnement réseau, dit nécessairement collaboration, et donc partage. C'est pourquoi, le P2P, pour Peer-to-Peer (réseau d'égal à égal, mais on rencontre également pair to pair, de personne à personne), fait appel à un principe qui est au fondement même de tout dispositif informatique relié en réseau : celui du partage des ressources. Il existe toutefois une nuance de taille en ce qui concerne l'architecture réseau d'égal à égal, tel que nous l'entendons avec le phénomène P2P sur Internet. Il convient ici, avant de se lancer dans le coeur du sujet, de connaître quelques bases toutes simples directement tirées des technologies réseaux. En particulier, les principes fondamentaux de communication entre les clients et les serveurs.

1.1.2 Notion de Client/Serveur

Pour bien comprendre le Peer-to-Peer, il faut le distinguer de ce qu'il n'est pas. Sur le Web, et plus largement sur Internet, on trouve d'un côté les clients. Les clients désignent des applications que vous avez installées sur votre machine pour exploiter les services Internet ou ceux proposés sur un réseau privé : courrier électronique, Web, Newsgroups, etc. La plupart du temps, ce type de logiciels est gratuit mais l'accès aux ressources qu'ils permettent ne l'est pas toujours.
De l'autre côté, il y a les serveurs. Ils représentent des machines puissantes conçues pour fournir un service à plusieurs utilisateurs (ou clients) connectés simultanément. Vous l'avez compris, ces derniers partagent leurs ressources en ligne et, pour l'occasion, sont équipées de logiciels spécifiques (que l'on appelle également serveur), tout comme les utilisateurs sont équipés de programmes particuliers (les outils client) pour accéder à tous ses serveurs. Et dans la plupart des cas, à la différence des programmes client, les logiciels serveur ne sont pas gratuits. Bien entendu, un client précis ne pourra pas fonctionner avec tous les services proposés sur Internet. S'il a été développé pour accéder à une plate-forme de jeux en ligne propriétaire, par exemple, n'espérez pas alors l'exploiter de la même façon pour vous connecter sur le service d'un concurrent.
On parle alors d'architecture client / serveur.

A l'inverse, dans un environnement P2P tous les utilisateurs sont un peu clients et serveurs en même temps, à quelques exceptions près (réseau centralisé dans la lignée de Napster par exemple).

Concrètement, à travers une architecture P2P, chaque utilisateur peut partager et gérer des ressources comme il l'entend : définition des autorisations sur les fichiers et répertoires, élaboration des structures d'accès à l'information, gestion des ressources machines (publiques et privées), etc. Il n'existe plus de serveur centralisé pour stocker et gérer les données, mais l'information et la charge (connexion des utilisateurs pour récupérer l'information) sont réparties sur différentes machines, parfois de façon très subtile, toutes reliées entre elles grâce à un logiciel spécifique, mi-client mi-serveur.

1.1.3 *Avantages et failles des réseaux Peer-to-Peer*

Il s'avère assez difficile de définir les avantages et les failles de l'architecture réseau P2P. Tout dépend en fait du contexte dans lequel la technologie est exploitée. Ainsi, dans un cadre professionnel, les réseaux P2P n'ont pas une bonne presse. Le fait de ne pas pouvoir centraliser, en un seul endroit, toute l'information de l'entreprise, cela engendre des difficultés d'administration considérables. De plus, la faiblesse d'un tel système, si aucun dispositif de sauvegarde et de relais n'a pas été installé, apparaît dès qu'une ou plusieurs machines tombent en panne.
En outre, administration et sécurité vont toujours de pair, et là encore, dans ce domaine, les réseaux P2P ne brillent pas par leur qualité de protection et de confidentialité des données lorsqu'il s'agit de distribuer l'information.
D'un autre côté, les réseaux P2P emploient des programmes extrêmement simples à installer, à configurer et à exploiter, où qui tendent à le devenir. De plus, toujours dans le même esprit, l'économie financière du choix d'une telle architecture réseau se retrouve également au niveau des logiciels à installer. En effet, tout système d'exploitation digne de ce nom dispose d'outils intégrés pour participer à l'élaboration et au fonctionnement d'un réseau P2P.
Enfin, un réseau P2P n'entraîne pas les mêmes dépenses pour la location de bande passante si vous exploiter un tel service sur Internet. Mais, par voie de conséquence, la qualité et la rapidité des transferts ne sont jamais assurées.

En conclusion, le principal défaut des réseaux P2P pour les entreprises désigne l'absence d'administration (puisque chaque utilisateur est lui même administrateur de sa propre machine), mais il constitue également la principale qualité de cette architecture, particulièrement appréciée des milieux underground informatiques. Ainsi, dans un environnement de loisir et de divertissement sur Internet, le troque, l'échange et le libre téléchargement désignent toujours des plates-formes particulièrement fréquentées et qui remportent un large succès auprès des jeunes, principalement.
De telles architectures sont difficiles, voire impossible, à éliminer. Elles se créent de façon aléatoire et n'obéissent à aucune règle administrative, ou presque. Il est clair que nous sommes, actuellement, à l'an 1 du P2P grand public sur Internet. Il est évident que de nombreux modèles vont apparaître dans les mois et années à venir, ce qui permettra de faire un peu de ménage dans le paysage sulfureux du P2P, tel que nous le connaissons aujourd'hui et qui fait essentiellement parler de lui avec le célèbre format de compression et de diffusion audio : le MP3.

Cela dit, il est également évident que l'apparition d'un cadre juridique légal, pour les échanges de fichiers et le partage des ressources directement d'utilisateur à utilisateur, ne suffira pas pour éliminer toute la mauvaise graine. Comme dans tout autre domaine technologique, il existera toujours des gens pour détourner des outils de leur but principal. Et l'évolution de toutes ces technologies reste ainsi immuable, avec ses failles et ses parades. On notera que tout cela relève plus d'une course poursuite entre "hackers" et scientifiques, avec la mise en place progressive de nouvelles protections et l'apparition régulière de nouvelles "attaques". Dans une sorte de mouvement perpétuel...

Pour ce qui est de l'aspect technique des réseaux P2P, nous n'irons pas plus loin dans le cadre des généralités (nous y reviendrons néanmoins plus tard lorsque nous vous présenterons le fonctionnement précis des réseaux P2P). Dès lors, maintenant que vous connaissez les principales subtilités techniques nécessaires à la compréhension du sujet, passons à l'esprit véhiculé par ce phénomène. Car n'allez pas croire que le P2P, sur Internet, se réduit à une bande d'énergumènes qui s'échangent en cachette de la musique piratée. En fait, cela va beaucoup plus loin que ça car il est tout simplement impossible d'envisager Internet sans les activités de partage. De fait, l'acheminement de toutes les données repose sur des accords d'échange et de réciprocité plus ou moins formalisés entre les différents acteurs de ce marché. Cela mérite d'être regardé de plus près.

1.2 Architectures Peer-to-Peer

1.2.1 Architecture centralisée

Qui n'a pas déjà entendu parlé de Napster? Un service peer-to-peer spécialisé dans l'échange de fichiers son (au format MP3 et Windows Media) qui a largement contribué à développer la technologie P2P sur le marché grand public, et même professionnel. Or, toute l'originalité de ce réseau, devenu désormais commercial, consiste à avoir adopté une architecture centralisée.

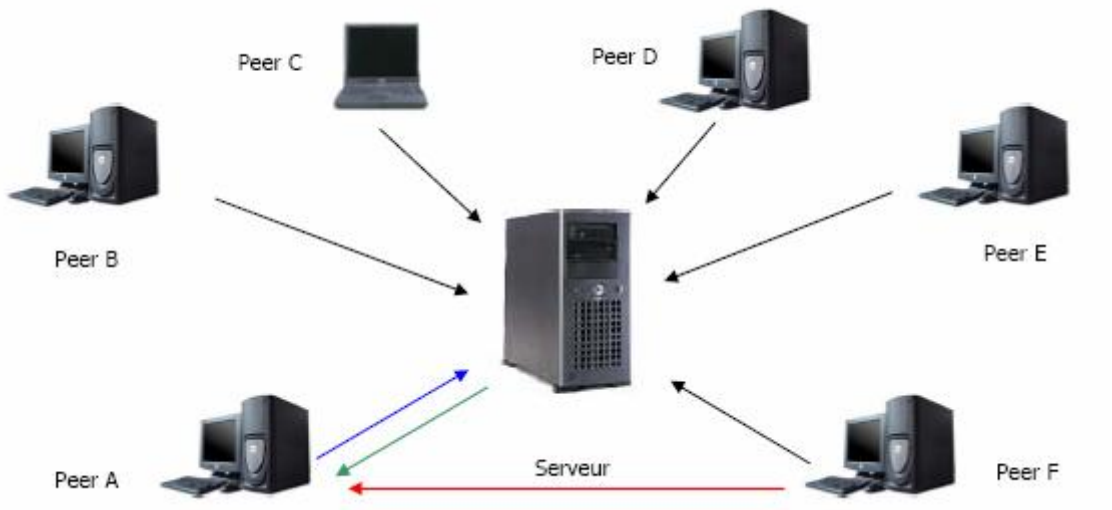
Sur le papier, un tel dispositif représente actuellement la solution la plus confortables pour échanger vos fichiers dans une communauté (musique, DVD...). Mais dans la réalité, ce type d'architecture exige un tel investissement en ressource que les services restent rarement de bonne qualité (lenteur, disponibilité...). Soit, ils sont saturés ; soit, ils sont limités en termes d'utilisateurs simultanés autorisés.

Concrètement, dans toute architecture centralisée, un dispositif exclusivement serveur se charge de mettre en relation directe tous les utilisateurs connectés. L'intérêt de cette technique réside dans l'indexation centralisée de tous les répertoires et intitulés de fichiers partagés par les abonnés sur le réseau. En général, la mise à jour de cette base s'effectue en temps réel, dès qu'un nouvel utilisateur se connecte ou quitte le service.

Comment cela fonctionne pour les clients ? :

Le plus simplement du monde, comme avec un moteur de recherche classique : vous lancez une requête en inscrivant un mot-clé. Vous obtiendrez une liste d'utilisateurs actuellement connectés au service et dont les fichiers partagés correspondent au terme recherché. Dès lors, il suffit de cliquer sur un des intitulés de lien pour vous connecter directement sur la machine correspondante et entamer le transfert. Dans ces conditions, à aucun moment les fichiers se retrouvent stockés sur le serveur central.

Schéma : Architecture centralisée



Légende :

- Un utilisateur recherche un fichier ressource en envoyant une requête au serveur central.
- Le serveur central répond et transmet la liste des ordinateurs utilisateurs proposant le fichier demandé.
- L'utilisateur télécharge le fichier directement à partir d'un des ordinateurs renseignés par le serveur.

Comme nous l'avons dit, le principal atout de cette méthode réside dans le confort et l'efficacité des recherches, à condition que le service ne soit pas surchargé et qu'il dispose d'une communauté suffisamment nombreuse pour le rendre intéressant. Ce qui n'est pas toujours le cas...

Enfin, nous terminerons par les principaux défauts qui reviennent lorsqu'on étudie de plus près le fonctionnement de ce type d'architecture. On en relèvera surtout trois :

- Au niveau de la sécurité, une architecture P2P centralisée s'avère particulièrement vulnérable. Elle ne propose qu'une seule porte d'entrée, son serveur centralisé, ce qui constitue le talon d'Achille de tout le réseau. Il suffirait effectivement de bloquer ce serveur pour déconnecter tous les utilisateurs et stopper le fonctionnement de l'ensemble du réseau.

- Autre chose, le fait de passer à travers une architecture centralisée, où il faut s'enregistrer pour pouvoir y accéder, ne garantit bien évidemment aucun anonymat. Le service connaît l'adresse IP de votre machine et le type de fichiers que vous téléchargez. Il peut facilement élaborer des profils d'utilisateurs et qui ne le fait pas ? De tels fichiers clients représentent pour le moment le principal fond de commerce des services exclusivement on-line. Vous êtes prévenu.

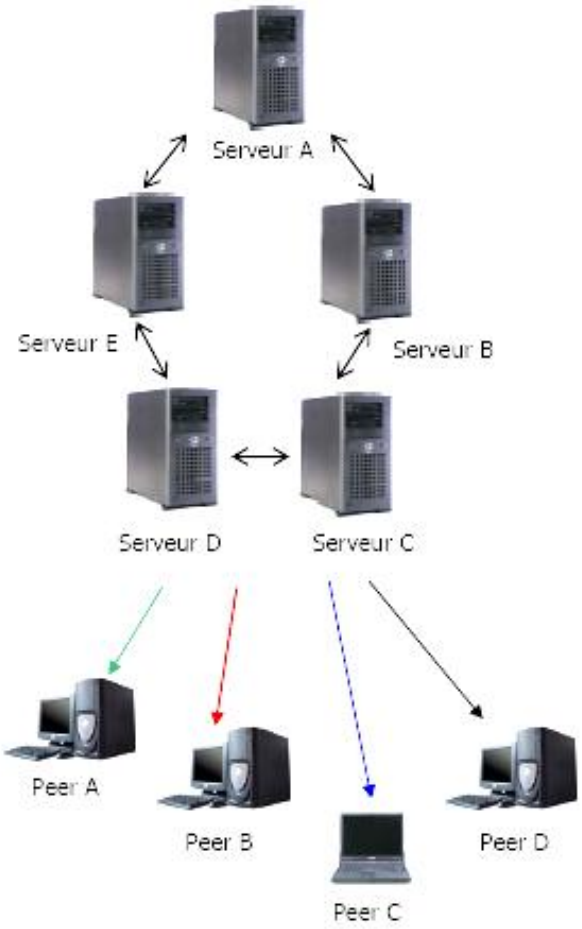
- Enfin, l'échange de fichiers numériques à grande échelle à travers Internet (musique, vidéo, photo, etc.) entraîne bien souvent le non respect des protections intellectuelles. Les oeuvres placées sous copyright qui circulent sur les réseaux P2P sont légion et on se retrouve ainsi en face d'une gigantesque organisation de piratage, et non plus à un petit réseau d'échange entre amis.

1.2.2 Amélioration du Peer-to-Peer centralisé

Pour résoudre les problèmes de robustesse et améliorer la qualité de connexion avec le serveur, le serveur central de l'architecture centralisée est remplacé par un anneau de serveur. Ceci permet d'éviter la chute du réseau si une panne se produit sur un serveur, car il y a toujours un point de connexion valide aux serveurs.

De plus l'utilisation de plusieurs serveurs permet de mieux répartir les demandes de connexions et donc de limiter la chute de bande passante.
Chaque serveur peut avoir accès aux informations des clients connectés sur les autres. L'accès aux données partagées est donc totalement transparent pour les utilisateurs.

Schéma : Amélioration de l'architecture Peer-to-Peer

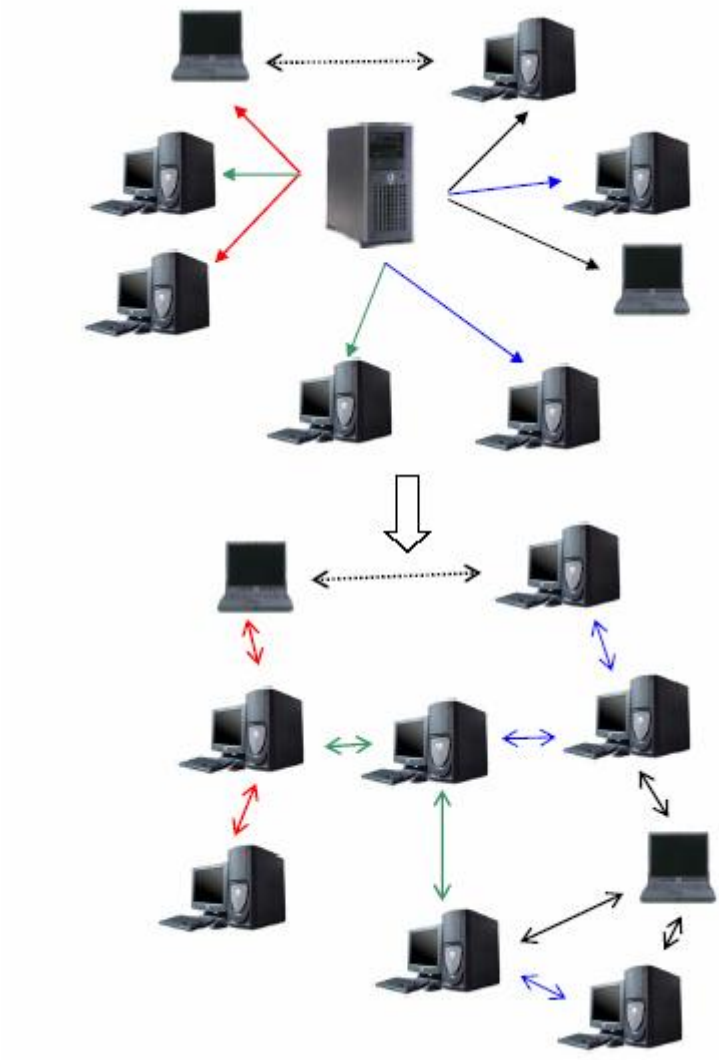


1.2.3 Architecture centralisée vers décentralisée

Nous avons vu que l'architecture centralisée pose des problèmes de sécurité, robustesse, et de limitation de la bande passante. Les problèmes sont directement issus de l'utilisation de serveurs dont le seul but est de posséder l'annuaire des clients.

Si on désire supprimer les serveurs centraux il faut donc trouver le moyen de constituer un annuaire sur chaque client, puis de les faire communiquer. C'est sur ces mécanismes que sont basés les réseaux Peer-to-Peer décentralisés. Il n'y a donc plus de serveurs centraux, ce sont tous les éléments du réseau qui vont jouer ce rôle. Chaque machine dans ses rôles est identique à une autre, c'est pour cela que l'on appelle ces types de réseaux Peer-to-Peer pur.
Un grand avantage de ce nouveau type de réseaux, est en théorie le total anonymat qu'il procure. En effet en évitant de communiquer avec une machine centralisant les demandes et les annuaires, on évite les problèmes de récupération des données utilisateurs.

Schéma : Architecture centralisée vers décentralisée



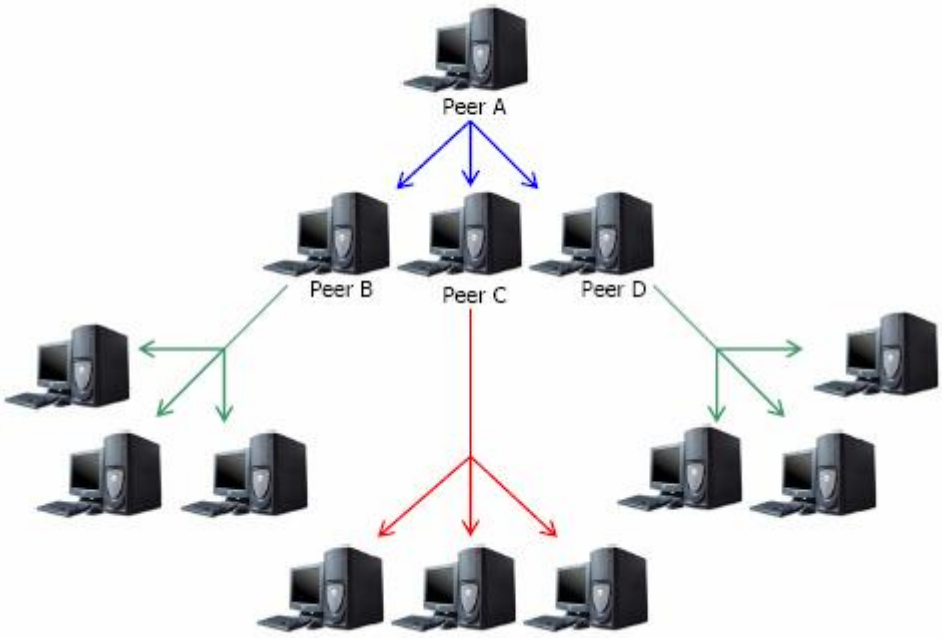
1.2.4 Architecture décentralisée

Les abonnés des réseaux P2P marquent une nette préférence pour les architectures décentralisées. Cela s'explique par l'absence d'anonymat dans les dispositifs centralisés et cela fait suite également aux difficultés judiciaires rencontrées par Napster et Scour, principalement. C'est pourquoi, très vite, sont apparues des solutions alternatives. On pense plus particulièrement à Gnutella qui constitue incontestablement le fer de lance des réseaux décentralisés.

Quoi qu'il en soit, en vous connectant à de tels réseaux, vous aurez toujours besoin d'un programme mi-client mi-serveur pour établir une connexion sur une ou plusieurs autres machines équipées, comme la vôtre, du même logiciel. Contrairement aux réseaux centralisés, ou il suffisait de se connecter au serveur pour avoir accès aux informations, il faut pour avoir accès à une information en décentralisé :

- Apprendre la topologie du réseau sur lequel le client est connecté.
- Rechercher l'information sur tous les noeuds.
- Recevoir une réponse d'un noeud répondant aux critères.

Schéma : Création d'un annuaire



Légende :

• Le client A se connecte sur le réseau, mais il ne connaît pas la topologie du réseau (A est totalement aveugle).

• Pour connaître les autres membres du réseau, A va "broadcaster" une demande d'identification des noeuds du réseau.

• Les noeuds recevant la demande vont à leur tour la répercuter sur tous les noeuds voisins et ainsi de suite (comme les noeuds B, C et D).

• Lorsque que la trame est reçue et identifiée par un autre client, le noeud renvoi une trame d'identification à A.

• Ainsi A va peu à peu pouvoir identifier tous les noeuds du réseau et se créer un annuaire. De cette façon, la taille d'une telle architecture réseau est théoriquement infinie.

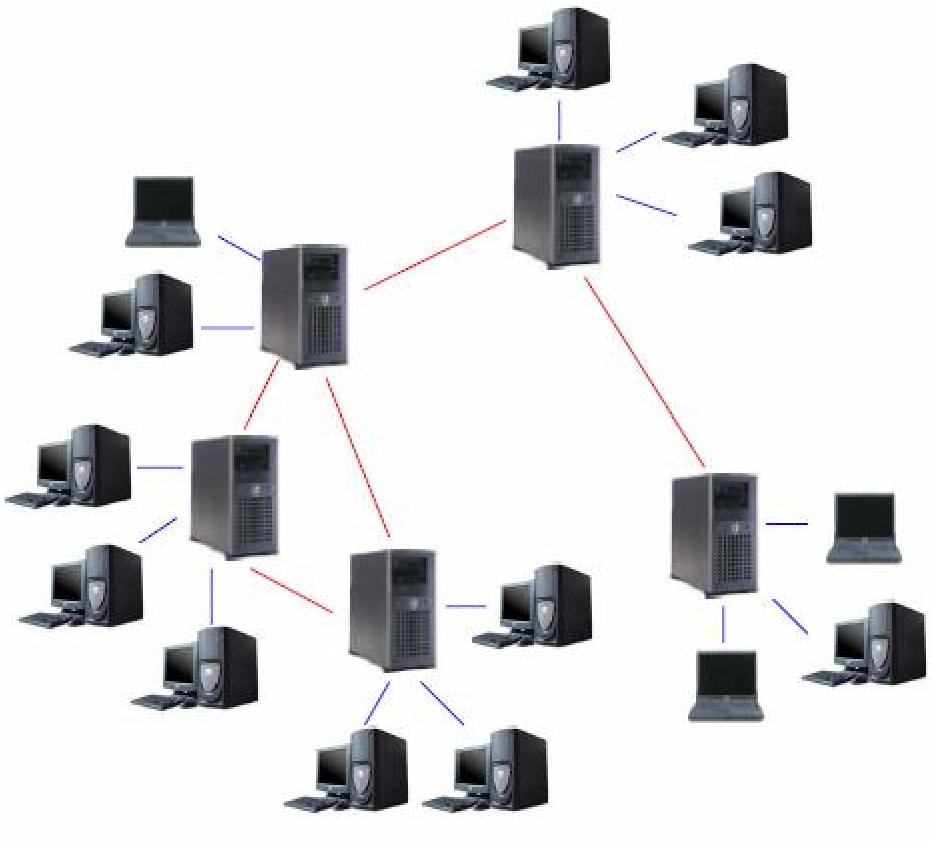
Une fois que votre machine fait partie intégrante du réseau, vous pouvez lancer une recherche à partir d'un ou plusieurs mots-clés, à l'instar de tout bon moteur de recherche que vous connaissez déjà. Mais on notera toutefois une différence de taille : lorsque vous validez une requête, celle-ci reste toujours active et son traitement ne s'arrête jamais, sauf si vous en établissez une autre. Cela s'explique par ce qu'on appelle l'horizon, c'est-à-dire toutes les machines du réseau auxquelles vous avez accès pour effectuer vos recherches. Dès lors, si de nouveaux ordinateurs apparaissent dans votre horizon, leurs ressources vous seront immédiatement accessibles. Concrètement, la liste des résultats se génère progressivement dans une fenêtre particulière de votre programme et dès qu'un fichier correspond au critère de recherche, il s'affiche dans la liste. Pour télécharger l'information correspondante, il suffit de cliquer dessus.

Le principal inconvénient de cette méthode est les séries de broadcast qui sont diffusées sur le réseau. Cela a pour conséquence de polluer et donc de ralentir les échanges de données entre les machines.

1.2.5 Modèle hybride : les réseaux Super Noeuds

Le modèle super noeud a pour but d'utiliser les avantages des 2 types de réseaux (centralisé et décentralisé). En effet sa structure permet de diminuer le nombres de connexions sur chaque serveur, et ainsi d'éviter les problèmes de bandes passantes. D'autre part le réseau de serveurs utilise un mécanisme issu des réseaux décentralisés pour tenir à jour un annuaire client et un index des fichiers à partir des informations provenant des autres serveurs. Un serveur peut donc proposer à n'importe quel client toutes les informations contenues sur le réseau.
Le réseau n'est plus pollué par les trames de broadcast. Mais la contrepartie est que l'anonymat n'est plus assuré.

Schéma : Les Super Noeuds



1.3 Liste des réseaux Peer-to-Peer

❖ Napster

La naissance de Napster

Fin 1998, Shawn Fanning, un étudiant américain passionné d'informatique alors âgé de 19 ans vient bouleverser le monde bien établi du client / serveur. Il décide de quitter l'université et se lance dans l'écriture d'un logiciel pour permettre l'échange de fichiers musicaux. La raison d'être de ce logiciel repose sur le constat suivant : rechercher des MP3 sur les moteurs de recherche habituels conduit à une perte de temps énorme et les réponses sont souvent inappropriées. Après quelques mois de travail acharné, une première version du logiciel est disponible. Fanning décide de tester une première version le 1er juin 1999 et appelle son logiciel Napster (son pseudo sur Internet). Le logiciel qui ne devait être testé que par quelques-uns de ses amis remporte un succès des plus rapides. Il conquiert notamment les universités. Shawn Fanning se retrouve propulsé à la tête d'une start-up pleine d'avenir. Lui qui déclare à propos de Napster qu'il n'avait "aucune envie d'en faire un business" voit les utilisateurs arriver en masse.
En septembre 2000, Napster atteint un nombre de téléchargement record. 1,39 milliard serait le nombre de chansons échangées par ses utilisateurs (source : Webnoize, Cabinet d'études américain).

La fin de Napster

Tombent alors les premières interdictions de la part des universités : les étudiants l'utilisent tellement qu'ils saturent les bandes passantes. Les groupes de musique demandent à ce qu'on protège leurs droits. Le groupe Metallica ouvre le bal et entame un procès. En décembre 1999 le RIAA intente également un procès à Napster.
En novembre 2000, il est prévu que Napster intègre un système anti-piratage mais en janvier 2001, le verdict de la 9 cour d'appel de San Francisco tombe : Napster viole la loi sur les droits d'auteurs et devra cesser dans un bref délai l'échange gratuit de fichiers musicaux MP3. C'est une victoire importante pour les maisons de disques, même si elles n'obtiennent pas la fermeture immédiate de Napster. En février 2001, un système de cotisations devait est mis en place, Napster échappe aux procès des maisons de disques mais les utilisateurs le délaissent.

OpenNap prolonge le protocole de Napster pour permettre le partage de n'importe quel type de supports, et il a la capacité de lier des serveurs ensemble. L'OpenNap est un serveur pour relier les clients ensemble, et n'est pas un client lui-même.

❖ Gnutella

Gnutella est né en mars 2000 des mains de Justin Frankel et Tom Pepper, deux programmeurs sans aucun papier universitaire, en seulement quatorze jours. A l'origine, le projet devait servir à s'échanger des recettes de cuisine et sa première apparition sur Internet n'a duré que quelques heures.
La raison ? Les directeurs d'America Online (propriétaires de Nullsoft, l'entreprise ayant développé Winamp et Gnutella) ne voyaient pas l'intérêt d'améliorer le partage de recettes. Gnutella a été déclaré « unauthorized freelance project » et « (..) put out to pasture like a car that goes a hundred miles on a gallon ».

Mais comment, me direz-vous, est-ce que Gnutella a vu le jour dans ces conditions ?!? En fait, un certain Bryan Mayland a fait du « reverse engineering » afin de trouver le langage de communication utilisé par l'application et l'a tout simplement publié à l'époque sur http://gnutella.nerdherd.net. Ensuite, Ian Hall-Beyer et Nathan Moinvarizi ont rassemblé toute une série de développeurs intéressés à la création d'applications basées sur Gnutella.

Gnutella est l'après Napster dans le sens où beaucoup d'utilisateurs se sont tourné vers cette solution suite à la fermeture des services de la start-up de Redwood, mais également dans le sens où c'est une étape de plus dans l'architecture. En effet, Gnutella ne possède aucun serveur central vu que sont application est à la fois client et serveur

Gnutella se distingue des autres réseaux peer-to-peer en proposant un réseau entièrement libre. Exemple pour enregistrer un client sur le réseau FastTrack, il faut avoir une licence. Or le réseau Gnutella comme son nom l'indique est sous licence Gnu (Gnu est l'abréviation de Gnu's Not Unix, le cri de la nouvelle génération de développeurs qui donne un accès gratuit aux sources de leurs programmes) et donc il est libre de droit. Il s'agit d'un peer-to-peer décentralisé qui a évolué au fil du temps. Son architecture était comparable à Edonkey avec des fonctionnalités en moins (pas de recherches sur les serveurs voisins etc.).

Suite à l'explosion du nombre de clients (de 75.000 à 300.000), le réseau a évolué vers le pur peer-to-peer. Pour se connecter l'utilisateur fait une sorte de "broadcast" vers l'Internet afin de trouver les utilisateurs les plus proches.

Actuellement, il existe sur la toile une multitude de clients se servant de cette technologie pour partager des fichiers. On peut citer Bearshare, Gnotella, Limewire, Phex, etc. De plus, les échanges peuvent s'effectuer quelque soit le client ou la plate-forme (Windows, Linux/Unix, Macintosh, etc.) sur lequel il se trouve.

Une plus ample analyse de Gnutella sera faite ultérieurement.

❖ eDonkey

Le réseau eDonkey est né en septembre 2000 mais c'est au cours de l'année 2001 qu'il a connu son véritable essor.
eDonkey représente un réseau particulièrement animé dont le client/serveur existe en deux versions : Windows et Linux. Dans tous les cas, le programme est gratuit et il vous permettra d'accéder à différentes ressources P2P pour télécharger vos fichiers.
Sa particularité ? Son efficacité. Par exemple, vous pouvez transférer un fichier à partir de la machine d'un client qui, lui aussi, transfère au même moment le même fichier à partir d'une autre machine client. Notez enfin que Edonkey fait partie des communautés P2P les plus importantes en ce moment.
Comme plusieurs autres, ce produit est hybride, il est décentralisé et l'architecture est proche du super nœud. Il ne fonctionne pas à partir d'un cluster centralisé de type Napster qu'on peut facilement identifier et désactiver. Mais son application en elle-même fonctionne selon un principe centralisé (j'entends par là qu'il existe sur son réseau des serveurs qui centralisent sous forme de pointeurs toutes les ressources disponibles sur chaque poste client). Il n'y a pas ici d'horizon (au sens de Gnutella) et de relais de requêtes.

Le plus gros avantage d'eDonkey est son protocole MFTP (Multisource File Transfert Protocol) Derrière cette abréviation se cache un système ingénieux pour optimiser les temps de téléchargement de fichiers. Sur un système classique, le client ne pourrait télécharger des fichiers qu'à partir de Peer sources disposant du fichier complet. Grâce au

MFTP, un client peut à la fois télécharger une partie d'un fichier à partir de plusieurs sources et aussi partager les parties déjà téléchargées.
Les auteurs d'eDonkey et de son réseau travaillent actuellement sur OverNet, un nouveau réseau pur peer-to-peer cette fois, un peu à la manière de FastTrack. Il élimine les serveurs et certains clients sont compatible eDonkey et OverNet.

❖ FastTrack

Le réseau FastTrack, dont le client le plus connu du grand public est Kazaa peut être comparé à eDonkey sur plusieurs points. Tout comme lui il permet le partage de fichiers de tous types et emploie le protocole MFTP.
Il connaît un énorme succès dans le monde, sauf en France. Plusieurs raisons à cela :

- La présence d'un spyware dans la version grand public du client Kazaa
- Le peu de fichiers disponibles en version française (la tendance s'inverse peu à peu).

Sa force réside dans la facilité d'utilisation puisque son interface est très intuitive et comporte un lecteur multimédia et une bibliothèque pour organiser ses fichiers. Autre avantage, beaucoup de peer de Kazaa dispose de liaison T1 et ainsi il n'est pas rare d'atteindre des débits supérieurs à 50ko/s.

Ils fonctionnent une subtilité supplémentaire : les super noeuds. Les utilisateurs disposant d'une connexion très rapide sont immédiatement considérée comme des super noeuds et jouent alors le rôle de serveur. C'est eux qui hébergent la liste des fichiers partages par les clients. Les serveurs principaux ne gèrent donc que les connections et la liste des super noeuds. Cette méthode permet au réseau FastTrack de disposer d'une quantité croissante de serveur de recherche avec le nombre de clients présents

Les super noeuds restent des Peer, ils partagent et téléchargent aussi des fichiers. Le temps CPU consommé par la fonction de super noeuds est de l'ordre de 10% et le client peut refuser de devenir super noeuds, mais dans cas, c'est la mort du réseau FastTrack.
Les super noeuds communiquent entre eux pour les recherches. En fonction du TTL (Time To Live) des paquets, la recherche est plus au moins profonde sur le réseau. Les échanges entre clients se font via le protocole HTTP 1.1.

Kazaa est le client le plus connu du réseau. Il est aussi le plus complet puisqu'il a été conçu par les auteurs du réseau. Une version "Spyware Free" est disponible et s'appelle Kazaa Lite.
Aucun client connu ne permet de se connecter avec une autre plate-forme que Windows. Il faut également disposer d'Internet Explorer et de Windows Média Player.

Le réseau FastTrack va faire l'objet d'une analyse détaillé par la suite.

❖ Freenet

Freenet est un réseau p2p à une large échelle, qui réunit la puissance des ordinateurs membres du monde entier pour créer une mémoire massive et virtuelle d'informations, ouverte à quiconque pour publier librement, ou accéder à, des informations de toutes
Le projet Freenet se trouve toujours, à l'heure actuelle, en phase de développement mais ses sources sont d'ores et déjà disponibles sur Internet. La définition de son protocole est

basée sur les idées de Ian Clarke alors étudiant à l'Université d'Edinburgh en Ecosse. Le principe de ce protocole est d'offrir un réseau dont les propriétés sont les suivantes :

- Anonymat pour les producteurs et les consommateurs des informations partagées sur le réseau.
- Déresponsabilisation des conservateurs d'informations.
- Résistance du réseau à diverses attaques menées par des membres tiers.
- Archivage et routage dynamiques des informations.
- Décentralisation de toutes les fonctions du réseau.

Si certaines de ces propriétés, comme la décentralisation des fonctionnalités, sont inhérentes à la majorité des réseaux p2p, les autres découlent des particularités même du réseau Freenet.

L'anonymat, qui est l'un des points clés de la philosophie de Freenet, est essentiellement assuré par les systèmes de sécurité – tel que le chiffrement des données qui transitent entre les nœuds – et par le fait que la connaissance locale est privilégiée sur la connaissance générale. Chaque nœud ne peut en effet communiquer qu'avec ses voisins immédiats et il n'est généralement pas possible de déterminer avec certitude l'origine exacte d'un message. Ceci est du au fait qu'en faisant suivre un message, le nœud qui réalise cette opération peut se déclarer comme étant la source de ce message.

Le réseau Freenet est l'un des trois réseaux dont nous allons étudier le fonctionnement dans le chapitre suivant

❖ DirectConnect

DirectConnect est original de par son fonctionnement. Les serveurs sont ici appelés des hubs (une traduction possible est "salon"). Chaque hub est créé pour accueillir une catégorie de fichiers ainsi qu'une véritable communauté autour d'un sujet précis (films, musique, animations, logiciels, jeux, …). Si on se connecte à ces hubs pour partager des fichiers ne correspondant pas à la catégorie du hubs, on est alors banni. L'interface et le fonctionnement du client peuvent s'apparenter à un IRC avec possibilité de partage de fichiers en plus. Le programme a un fonctionnement similaire à eDonkey (liste d'attente, décentralisé) mais ne possède ni le MFTP, ni la communication entre les serveurs lors des recherches.
Autre différence, il faut parfois partager plusieurs Go avant de pouvoir accéder aux hubs les plus intéressants du réseau. L'impossibilité de connaître la liste de tous les hubs en fait une communauté un peu fermée face aux autres peer-to-peer. De fait de sa technologie moins évoluée, il est également moins performant.

❖ OverNet

L'intérêt d'un réseau décentralisé comme OverNet est que les sources ne sont plus seulement propagées aux clients connectés au même serveur mais à l'ensemble du réseau

❖ Groove

Groove networks a été fondé en 1997 par l'inventeur de Lotus Notes ! Groove Networks distribue gratuitement Groove, son logiciel.
Groove est une application P2P pour le travail en groupes « flexibles ».
Il permet le dialogue en temps réel, le partage de documents ou d'outils dans des espaces de travail personnalisés et autonomes. Des outils de messagerie instantanée ou vocale sont également intégrés à Groove.
Il constitue une représentation « virtuelle » d'un groupe de travail à l'aide d'outils de groupware : tableau blanc, calendrier, gestionnaire de conversation hiérarchisé ou système de partage de fichiers.
Groove préserve l'autonomie et la liberté de chacun : c'est un gage d'appropriation

❖ Blubster, Piolet

Blubster offre uniquement le partage fichier MP3. C'est un pur Peer-to-Peer et il utilise un protocole basé UDP pour le transfert des fichiers. Son moteur de recherche décentralisé est extrêmement rapide et le nombre de fichiers présents est très important.
Théoriquement il peut se connecter de n'importe où…
Son créateur (Pablo Soto, 22 ans) annonce même : "Le plus gros avantage de Blubster finalement est sa faculté d'extension et l'intelligence de l'organisation de notre réseau. Il n'y a en fait aucune limite au nombre d'utilisateurs connectés en même temps".
Nous avons calculé que même 1.000.000 d'utilisateurs pouvaient facilement se connecter simultanément sans problèmes de performances.
A présent Blubster se nomme Piolet ". Le réseau Piolet attend les anciens utilisateurs d'Audiogalaxy ! "

❖ Tapestry, CAN, Chord

Jusqu'à présent, nous avons étudié des réseaux qui étaient soit centralisés mais permettant une recherche exhaustive (Napster) soit décentralisés mais ne permettant qu'une recherche non exhaustive (Gnutella et Freenet). Il existe cependant des réseaux décentralisés qui autorisent également une recherche exhaustive. Nous pouvons citer CAN (*Content Adressable Networks*) qui est un réseau en tore dont la taille est fixée initialement ou Chord qui est développé selon une architecture en hyper cube. Pour finir nous avons Tapestry dont le but est d'offrir un réseau de partage de données garantissant leur pérennité.
Tapestry est utilisé dans des systèmes tels qu'OceanStore qui permet la conservation persistante de données à travers une architecture globale, ou encore dans l'application Bayeux. Les mécanismes de localisation et de routage du système .Tapestry sont basés sur les travaux de Plaxton, Rajamaran et Richa. Tous les noeuds du réseau remplissent les rôles de client, en émettant des requêtes, de serveur, en conservant des objets, et de routeur, en redirigeant les requêtes.
Toutefois, Tapestry évite certains inconvénients du système Plaxton. Ainsi, si Plaxton se limite à l'utilisation d'algorithmes statiques, Tapestry supporte des opérations dynamiques et décentralisées.

❖ JXTA

Le projet JXTA est la plate-forme basée sur les technologies Java proposée par Sun Microsystems offrant un ensemble de mécanismes simples permettant le développement d'applications basées sur le peer-to-peer. La structure de JXTA peut se diviser en différentes couches.

- **Le noyau**. Il contient les principales fonctionnalités du système, à savoir la gestion des canaux de communication, de la sécurité ou encore des groupes de pairs.
- **La couche des services**. Certains services sont déjà implantés dans la plate-forme JXTA : ce sont les services de Sun, à savoir l'indexation, la recherche et le partage de données. D'autres services sont et seront développés et ajoutés par la communauté des développeurs JXTA.
- **La couche des applications**. A l'image de la couche précédente, il existe d'une part les applications proposées par Sun et d'autre part celles développées par la communauté JXTA.

La plupart de ces concepts, tels que les pairs, les groupes de pairs, sont des concepts hérités des architectures peer-to-peer. Cependant, certains de ces concepts comportent des particularités propres au système JXTA.

❖ Jini

La technologie de réseau de Jini fournit les mécanismes simples qui permettent à des dispositifs de se brancher ensemble pour former une communauté impromptue c'est-à-dire une communauté montée sans aucune forme de planification, d'installation ou d'intervention humaine. Chaque dispositif fournit des services que d'autres dispositifs dans la communauté peuvent également utiliser. Ces dispositifs fournissent leurs propres interfaces, ce qui assure fiabilité et compatibilité. Afin de clarifier les esprits, on peut représenter cette technologie sous forme de couches. La centrale de référence peut être comparée aux serveurs d'eDonkey à la différence près que les appareils ont chacun un identifiant unique (Il s'agit d'un annuaire).
L'apport de Jini au Peer-to-Peer est une mobilité quasi totale. En effet, n'importe quel dispositif comportant cette technologie peut changer de voisinage très aisément afin d'établir une connexion avec un ou plusieurs autres dispositifs, pour autant qu'il soit autorisé dans la centrale de référence. Dans les autres environnements Peer-to-Peer, un voisinage donné est très structuré. Java de son côté amène deux avantages. Tout d'abord la portabilité de la technologie, car il suffit d'avoir une JVM (machine virtuelle Java) dans le dispositif. Ensuite, le niveau sécuritaire du langage qui permet d'établir les droits de connexion entre dispositifs aisément.

❖ <u>SETI@home</u>

SETI@home est une expérience scientifique en radioastronomie exploitant la puissance inutilisée de millions d'ordinateurs connectés via Internet dans un projet de Recherche d'une Intelligence Extra-terrestre (Search for Extra-Terrestrial Intelligence, alias SETI). SETI est un cas à part dans les réseaux peer-to-peer grand public puisqu'il ne permet pas de partager des fichiers, mais du temps/machine.

Au début de 1960, le premier programme SETI a été lancé en recherchant des signaux radio émis par les galaxies. Le financement de ces programmes a toujours été aléatoire car il fallait en effet disposer d'une très grosse puissance de calcul. En 1995, David Gedye eut l'idée d'utiliser la puissance inutilisée des PC connectés à Internet. Avec 3 autres collègues de l'université de Berkeley, ils présentèrent l'idée sur un site Web au début de 1998. Ils recueillirent l'inscription de plus de 4.000.000 de volontaires. Le premier logiciel client fut disponible le 17 mai 1999. Le 14 octobre 2001, il y avait 3311948 participants dans 226 pays. Le nombre d'instructions exécutées a dépassé 1 ZettaFLOP (10 puissance 21) ce qui en fait le plus gros calcul jamais effectué. En 24 heures, la puissance de calcul a été de 88 TeraFlops/seconde. Le plus gros ordinateur du Monde est actuellement l'ASCI White d'IBM : 110 Millions de dollars, 106 tonnes et seulement 12.3 TeraFlops/s. Pour moins de 1% du coût, le programme Seti@Home est beaucoup plus puissant.

Même si on ne trouve pas d'extraterrestres, ce programme aura démontré comment les techniques logicielles pouvaient transformer un grand nombre d'ordinateurs individuels peu fiables et connectés de manière intermittente en un système très rapide et très fiable.

Ils regroupent 4 millions d’utilisateurs de 226 pays différents.

	Total
Utilisateurs :	4.230.713
Résultats reçus :	761394877
Temps total CPU :	2.553625e+21 année
Temps CPU par unité de travail :	15h09

Le volume de données est très important mais heureusement il est morcelable assez facilement pour être exploité par les membres du réseau.

Le site français SETI@home nous explique le principe des calculs de SETI : http://setiathome.free.fr/information/about_seti/about_seti_at_home_1.html

❖ Entropia

Entropia est l'exemple type, après SETI@Home, en ce qui concerne les applications centrées sur le calcul. Elle fournit une voie extrêmement rentable de passer à l'échelle d'un superordinateur afin d'exécuter des applications intensives en calculs.
La plate-forme informatique répartie d'une entreprise peut ainsi exploiter la capacité de traitement considérable, mais inutilisée, des PCs formant son réseau (ou des machines connectées à l'Internet) et la rend disponible pour exécuter de grandes applications commerciales et scientifiques.

L'idée de faire du calcul distribué est venu du fait que la majorité du temps, un PC le passe à ne rien faire. Entre les clicks de souris, les frappes de claviers, et les transitions de l'activité courante d'applications, le PC moyen veille jusqu' à 95% de son temps. En étendant ce constat à des centaines de millions de PCs dans le monde qui sont reliés en réseaux locaux, voire à toutes les machines sur Internet, il y a une énorme quantité de puissance de calcul inutilisées qui ne demande qu'à être exploitée.

Entropia propose quatre projets auquel quiconque possédant une connexion Internet peut y participer. Il s'agit de FightAIDS@Home, SaferMarkets, Benchmark App 0020 et Entropia Research.
Le but du premier est d'aider la recherche contre le SIDA en modélisant l'évolution de la résistance de médicaments pour le traitement HIV et de créer de nouveaux médicaments contre le SIDA.
Le deuxième est un projet voulant mieux comprendre les marchés financiers. L'idée est de faire tourner des modèles de simulations et de prévision afin de mieux prédire les comportements futurs sur la base de performances passées.
En ce qui concerne les deux derniers projets, aucune information n'a été trouvées, mais il pourrait bien s'agir d'applications commerciales qui aident Entropia à financer ces projets « pour la bonne cause ».

Il faut savoir que même s'il l'on ne souhaite aider que la recherche pour la lutte contre le SIDA, Entropia se réserve le droit d'utiliser votre ordinateur également pour des projets commerciaux. A vous de voir si votre conscience humanitaire vous le permet…

Il est a soulever que les problèmes qu'on dû rencontrer Entropia & Co ne sont pas nouveaux. En effet, ils se sont retrouvés face à des problèmes traditionnels d'architectures distribuées parallèles et massivement parallèles. A une différence près tout de même : ces entreprises n'ont pas le contrôle de leur périmètre de travail, et c'est justement cette caractéristique qui permet d'inclure ce genre de projet dans la dénomination Peer-to-Peer

❖ Autres réseaux

Nous avons essayé de répertorier tous les réseaux peer-to-peer qui présentaient le plus d'intérêt au niveau de notre étude sur les réseaux peer-to-peer. Voici à présent une liste de réseau qui mérite d'être cité même titre que les réseaux précédents :

- **Firstpeer, Biz2peer**

Ces réseaux liés au commerce électronique construisent des places de marché distribuées…

- **Akamai**

Le réseau Akamai est une architecture distribuée de contenu Internet. L'internaute va chercher les données issues de son site favori sur le serveur le plus disponible.

- **iMaestro**

Ce réseau propose des enchères en P2P

- **Infrasearch**

Grâce a Infrasearch, moteur de recherche en P2P,il es possible d'aller chercher l'info là où elle a le plus de chance de se trouver…

- **Neurogrid**

Neurogrid propose une nouvelle approche de la recherche en réseau : je demande à … qui va dire de demander à …

- **Human links (Amoweba)**

Grâce à Human links nous pouvons analyser les pages Web issues des bookmarks des internautes « abonnés »

- **Océanstore, eMikolo, Opencola**

Ces projets proposent une alternative au stockage d'information centralisé en le répartissant sur un réseau de serveurs ou même de Px.

- **IRC**
- **URLBlaze**
- **Kademlia**
- **MojoNation**
- **Carracho**
- **…**

Commentaire :

La technologie du P2P est en constante évolution et il est difficile pour nous d'établir une liste exacte de tous ces réseaux apparaissant sur le net. Certains ne font pas « long feu », d'autres sont ciblés à une catégorie d'utilisateur a but professionnel et d'autres permettent à tous les utilisateur d'obtenir de tous et de rien à la fois. A cela vient s'ajouter la centaine de projets reposant sur la technologie du P2P. Il y a aussi certains réseaux que nous n'arrivions pas définir comme telles (manque d'information ou trop d'information contradictoire).

Grâce a une étude plus approfondi sur les réseaux Gnutella et Freenet, nous tenterons de comprendre plus en détaille comment fonctionne les réseaux peer-to-peer. Mais aussi de comprendre l'intérêt d'une telle technologie et la possibilité qu'elle nous offre à l'avenir.

II. Etudes des réseaux : Gnutella et Freenet

Il ne s'agit pas ici de vous présenter une liste exhaustive des réseaux Peer-to-Peer accessibles via l'Internet mais plutôt de vous montrer les principaux acteurs présents et leur motivation respective.

2.1 Technologie Gnutella

2.1.1 L'architecture du réseau Gnutella

Gnutella est un protocole plutôt qu'un logiciel. Il a été inventé par Nullsoft, l'éditeur de WinAmp aujourd'hui racheté par AOL, mais les sources sont disponibles. Ce protocole permet de relier les ordinateurs les uns aux autres, il permet le **transfert de fichiers** d'une machine à une autre de façon plus ou moins anonyme. Chaque machine est à la fois un serveur de fichiers et un client. Contrairement **Napster** qui était relié à un serveur central, il n'existe pas de tel serveur pour Gnutella, chacun fait partie intégrante du réseau et ce réseau se modifie au cours des connexions : **Peer to Peer**.
Le principe est le suivant : un peer A, équipé d'un programme spécifique (Baptisé « servent » par Gnutella, car il remplit les fonctions de client et de serveur à la fois.) se connecte à un peer B, lui aussi équipé de ce programme. A lui annonce ainsi sa présence sur le réseau. B relaie cette information à tous les peers auquel il est connecté. Ceux-ci signifient alors leur présence à A et relaient l'information à leur tour aux peers auxquels ils sont connectés, et ainsi de suite jusqu'à « l'horizon visible (La distance de cet horizon est déterminée comme sous IP par une durée de vie (TTL) des paquets.) » du peer A.

Une fois que A a annoncé sa présence aux autres membres du réseau de peers, il peut s'appuyer sur les peers qu'il connaît pour effectuer ses recherches.
Pour obtenir une ressource, A lance une requête vers certains peers du réseau qui la relaient vers les peers auxquels ils sont connectés, qui eux même la transmettent.
Si l'un des peers dispose d'une ressource qui pourrait convenir à A, il transmet l'information vers A. Ce dernier pourra ainsi ouvrir une connexion directe vers cet ordinateur et obtenir la ressource.

Ce modèle, en étant décentralisé, est beaucoup plus robuste qu'un modèle centralisé puisqu'il n'est pas dépendant d'un serveur, qui est le point de défaillance potentiel d'un réseau centralisé. Il tire partie de l'intermittence des connexions des nœuds car si l'un des noeuds se déconnecte du réseau, la requête pourra être poursuivie vers les autres ordinateurs connectés.

Un grand avantage de ce nouveau type de réseaux, est en théorie le total anonymat qu'il procure.
En effet en évitant de communiquer avec une machine centralisant les demandes et les annuaires, on évite les problèmes de récupération des données utilisateurs.

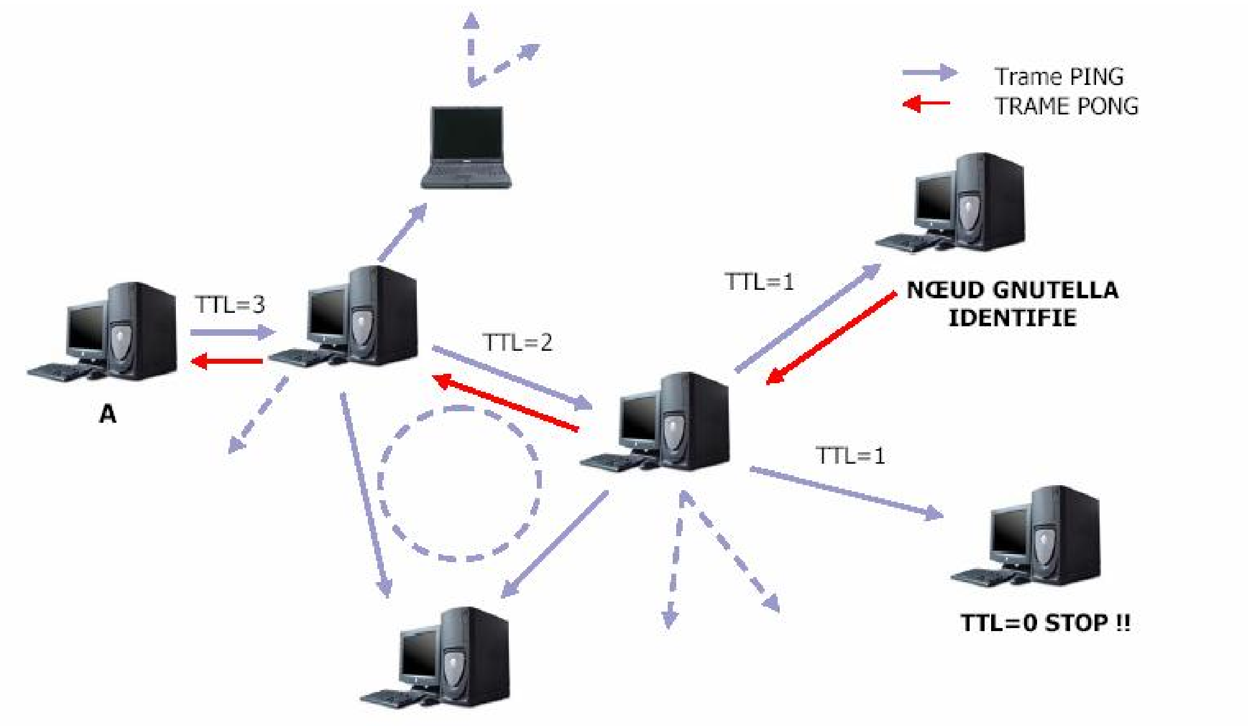
En revanche, en raison de la façon dont sont transmises les requêtes (**broadcast**), la bande passante nécessaire pour chaque requête croît exponentiellement quand le nombre de peers croît linéairement.

De plus, ce type de mécanisme est très facilement victime d'activités malicieuses.
Des membres malintentionnés peuvent envoyer en grande quantité des requêtes erronées qui produisent une lourde charge sur le réseau, réduisant ainsi son efficacité.

2.1.2 L’identification des nœuds du réseau

Un client A se connecte sur le réseau. Il commence par rechercher tous les nœuds Gnutella présents. Pour cela il transmet une trame d’identification (**PING**) à tous ces voisins qui eux-mêmes la transmettront à leurs voisins. Ces envois sont encapsulés dans une trame **TCP**. Pour borner la recherche le mécanisme de recherche joue sur le **TTL** de la trame. A chaque nœud du réseau le **TTL** est décrémenté et lorsqu’il devient égal à 0 la retransmission est stoppée.

Un mécanisme permet d’éviter les boucles dans la transmission. Lorsqu’une trame est reçue elle est stockée pendant un court laps de temps. Si le nœud reçoit pendant ce laps de temps une trame identique il la rejette car elle est déjà traitée. Lorsqu’un nœud est identifié, il envoie à l’émetteur une trame de réponse (**PONG**).



Ce protocole fonctionne au moyen de 5 descripteurs principaux, qui permettent la transmission des informations entre les servents (ou noeuds) du réseau, et d'un ensemble de règles qui régissent l'échange de ces descripteurs. Les voici :

Descripteur	Description
Ping	Utilisé pour trouver les autres noeuds sur le réseau. Un servent recevant un Ping doit répondre avec un (ou plusieurs) Pong.
Pong	Réponse à un Ping. Le servent répondant par un Pong livre son adresse IP ainsi que des informations sur les données qu'il partage.
Query	Requête visant à trouver un ou plusieurs fichiers vérifiant certains critères.
QueryHit	Réponse à un Query. Donne une liste de fichiers correspondant à la requête, ainsi que l'adresse IP des servent où ces fichiers on été trouvés.
Push	Mécanisme permettant aux contributeurs situés derrière un Firewall de se raccorder au réseau.

2.1.3 En tête de Gnutella

L'en-tête commune Gnutella est constituée de 5 champs et mesure 23 octets :

Gnode ID 16 octets	Payload Descriptor 1 octet	TTL 1 octet	Hops 1 octet	Payload length 4 octets

Gnode ID : identification du nœud dans le réseau Gnutella (Cet ID est initialisé lors de l'installation du client).

Payload Descriptor : identifiant de fonction de la trame (PING, PONG, …). Le corps de la fonctionnalité est transmis dans les trames TCP qui suivent.

0x00 Ping
0x01 Pong
0x40 Query
0x80 QueryHit
0x81 Push

TTL (Time To Live) : permet de borner l'horizon du réseau. Le champ TTL est décrémenté à chaque passage dans un nœud. A 0 la trame est détruite.

Hops Indique le nombre de sauts que peut encore effectuer descripteur entre servents Gnutella avant sa destruction.

Payload length : longueur du corps de la fonctionnalité.

2.1.4 Les différentes fonctionnalités

PING : Trame de demande d'identification des nœuds. La trame principale se suffit à elle même car le champ Payload est vide et sa longueur est nulle. Il permet de sonder le réseau à la recherche d'autres servents. Chacun des servents qui reçoit un Ping répond avec un Pong qui contient l'adresse et la quantité de données partagée par lui-même ou par un autre servent actif. La responsabilité de déterminer la fréquence d'émission de Ping revient aux programmes utilisant Gnutella. En effet, des Pings fréquents permettent de "resserrer" les liens entre un servent et le reste du réseau, mais ces Pings (et surtout les Pongs ainsi générés) risquent à court terme de saturer le réseau.

PONG : Trame de réponse au PING. Et sa structure est la suivante :

Port 2 octets	IP Address 4 octets	Number of Files shared 4 octets	Number of Kbytes shared 4 octets

Port : Port de communication sur lequel les échanges peuvent s'effectuer.
IP Address : Adresse IP du nœud identifiée.
Number of Files : Nombre de fichier partagés.
Numbers of Kbytes : Taille des données partagées.

Il est à noter qu'à un Ping reçu, plusieurs Pongs de réponse peuvent être renvoyés. Cela permet à un servent de renvoyer des informations le concernant personnellement, mais aussi des informations qu'il gardait en mémoire cache sur d'autres servents.

QUERY : Trame de recherche d'informations.Sa structure est la suivante :

Minimum Speed 2 octets	Search criteria N octets

Minimum Speed : vitesse de transmission minimum requise (ex : >10ko). Les servents plus lents que ce seuil ne sont pas censés répondre.

Search criteria : Cette chaîne se termine par 0x00. Sa longueur n'est limité que par le champ Payload length de l'en-tête. Cette chaîne contient des critères de recherche, tels que des noms de fichier, ou des types de fichiers (MP3, jpeg,...)

QUERY HITS : Trame de réponse au QUERY, de structure :

Number of Hits	Port	IP address	Speed	Reult Set			Gnode ID
				File Index	File size	File name	
1 octet	2 octets	4 octets	4 octets	4 octets	4 octets	N octets	16 octets

Number of hits : Nombre de "coups au but" dans le Result Set.

Port : Port de communication sur lequel les échanges peuvent s'effectuer.

IP Address : Adresse IP du nœud atteint.

Speed : Vitesse de transmission du noeud.

Gnode ID : Identification du nœud répondant à la requête dans le réseau Gnutella.

Result Set : Ce champ est composé d'un index des fichiers partagés correspondant aux critères de recherche. Et deux autres champs qui représentent la taille et le nom du fichier correspondant.

File Index : Il s'agit d'un nombre attribué par le servent répondant, destiné à identifier de façon unique le fichier correspondant à la requête.

File Size : Taille en ko du fichier référencé par File Index.

File Name : Nom du fichier référencé par File Index, se termine nécessaire ment par 0x0000. Sa taille est limitée par le champ Payload Lenght de l'en-tête.

De même que le Pong ne peut exister qu'en tant que réponse à un Ping, un QueryHit n'est envoyé que si le servent possède des fichiers correspondants aux critères de recherche d'un Query.
De plus, il est à noter que le champ Gnode ID dans l'en-tête du QueryHit doit contenir la même valeur que le champ Gnode ID de l'en-tête du Query. Ainsi, lorsqu'un servent reçoit un QueryHit, il sait à quelle requête ce QueryHit fait réponse.
PUSH
Structure :

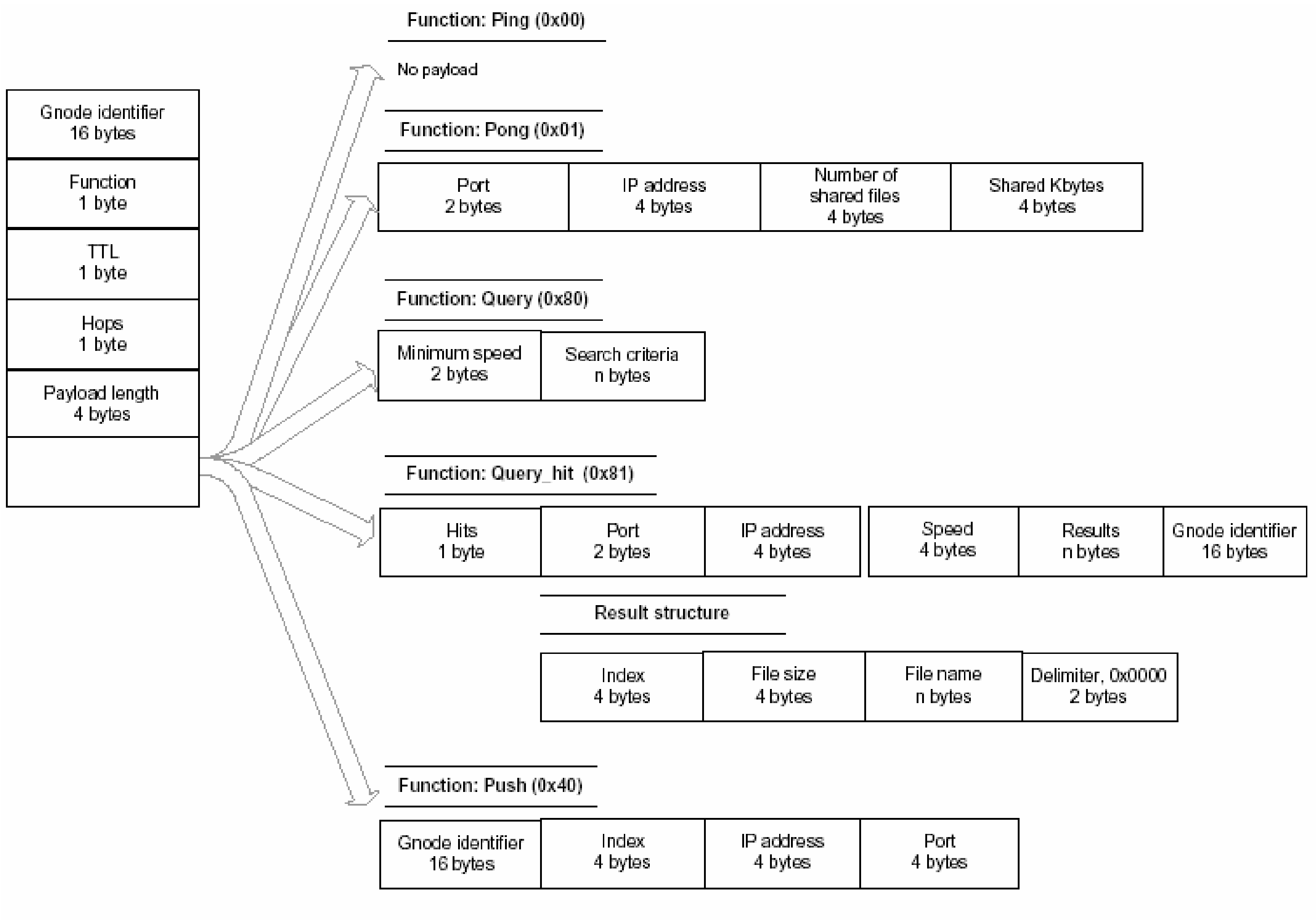
Gnode ID 16 octets	File Index 4 octets	IP adress 4 octets	Port 1 octet

Gnode ID : Chaîne de 16 octets qui identifie le servent à qui l'on demande de "pousser" le fichier référencé par File Index. Le servent qui initie la requête Push doit remplir ce champ avec la valeur Servent Identifier du QueryHit correspondant. De cette manière, un servent recevant la requête Push peut déterminer s'il en est ou non le destinataire.
File Index : Cet index détermine le fichier qui doit être "poussé" à partir du servent cible. Il doit correspondre au File Index du Result Set du QueryHit correspondant.
IP Adress : Adresse IP du servent vers lequel le fichier référencé par File Index doit être "poussé".

Port : Port vers lequel le fichier référencé par File Index doit être "poussé".
Un servent peut envoyer un Push lorsqu'il reçoit un QueryHit venant d'un servent n'acceptant pas les connexions incidentes (à cause d'un Firewall par exemple). Un servent recevant un Push peut s'y conformer si et seulement si le champ Gnode ID du Push contient la valeur de son propre identifiant.

Concrètement, l'utilité du Push est la suivante :
Si un servent se trouve situé derrière un Firewall, il sera dans l'incapacité de construire une liaison TCP avec le servent sur lequel se trouvent les fichiers désirés, et ne pourra donc pas initier le téléchargement. Cependant, ce dernier servent (jouant ici le rôle de serveur) peut, lui construire cette liaison TCP, et " uploader " les fichiers plutôt que laisser l'autre servent (ici dans le rôle de client) les " downloader ". L'objet du Push est de dire au servent serveur : "A toi d'initialiser le transfert".

On peut résumer tout ce qu’on vient de voir dans un schéma expliquant l’encapsulation des trames du protocole Gnutella .



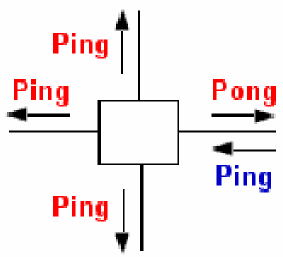
Gnutella protocol, messages data structures, version 0.4

2.1.5 Routage des descripteurs

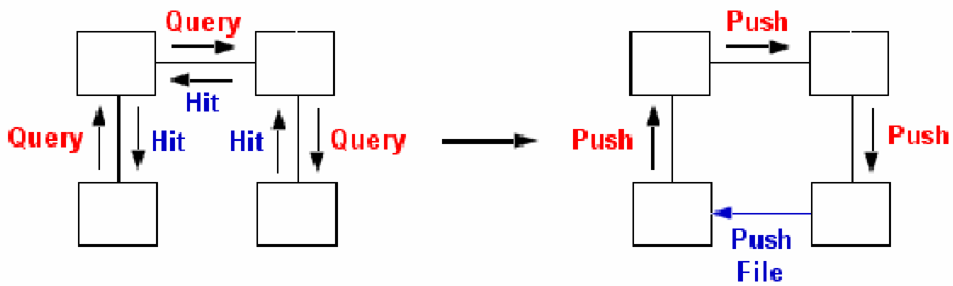
Les servents du réseau Gnutella doivent router les descripteurs de façon appropriés. Ce routage s'effectue en suivant les règles suivantes :
Les Pongs empruntent le même chemin que les Pings auxquels ils répondent. Ainsi, seul les servent ayant routé le Ping initial verront passer le Pong de réponse. Un servent recevant un Pong avec un Gnode ID donné, et n'ayant jamais vu passer un Ping avec le Gnode ID correspondant, détruira le Pong reçu.
Même mécanisme pour les QueryHit : ils suivent le chemin parcouru par le Query correspondant, et s'ils passent par un servent n'ayant jamais vu le Query correspondant, ils sont détruits.

Même mécanisme pour les Push : ils suivent le chemin parcouru par le QueryHit correspondant, et s'ils passent par un servent n'ayant jamais vu le QueryHit correspondant, ils sont détruits.
Un servent recevant un Ping ou un Query le transmettra à tous ses voisins immédiats, sauf celui d'où provient le descripteur.
Chaque servent, à la réception d'un descripteur, décrémente son TTL et incrémente son Hops. Si le TTL est alors à 0, le descripteur n'est plus transmis.
Un servent recevant un descripteur avec le même Payload Descriptor et le même Gnode ID qu'un autre descripteur précédemment reçu évitera de le transmettre à d'autres servents. Ces derniers devraient l'avoir déjà reçu auparavant, inutile donc d'augmenter le trafic.

Exemples de routage :



Example 1. Ping/Pong Routing



Example 2. Query/QueryHit/Push Routing

Une fois que les requêtes sont reçues par le nœud d'origine. L'utilisateur peut décider de récupérer un fichier. Pour cela le téléchargement s'effectue directement entre les deux nœuds via le protocole http. Ce transfert ne s'effectue jamais sur le réseau Gnutella lui-même (pour des raisons évidentes de surcharge!).

2.1.6 Les clients Gnutella

Le partage de fichiers partiellement téléchargés n'est possible qu'avec certains clients du réseau Gnutella (Ares, BearShare, Morpheus…). Certains d'entre eux, à la manière de Kazaa, imposent l'affichage de publicités.
Gnutella dispose de clients Linux et Macintosh :

Windows	Linux / Unix	Macintosh
Gnutella Clients BearShare Gnucleus Morpheus Shareaza Swapper XoloX LimeWire Phex	Gnutella Clients Gnewtellium Gtk-Gnutella Mutella Qtella LimeWire Phex	Gnutella Clients LimeWire Phex

2.2 Technologie Freenet

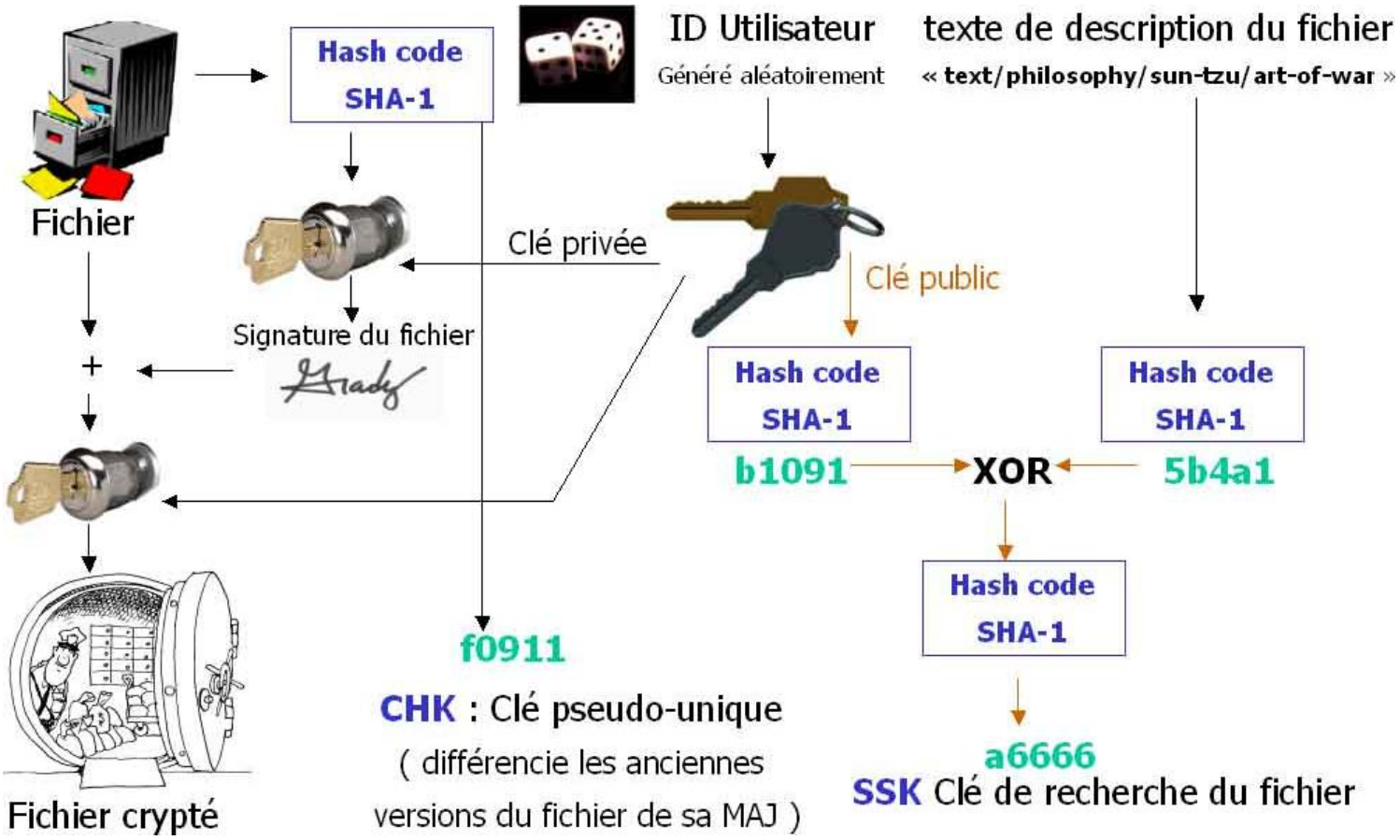
2.2.1 La publication de données

Dans un premier temps, un document est stocké dans un **noeud FreeNet**. Puis à chaque fois (ou presque) qu'un utilisateur demandera à consulter ce document, il sera dupliqué et stocké sur un autre nœud plus proche de cet utilisateur.
Ainsi, il devient absurde de référencer un document d'après son emplacement physique comme sur le web (http://machine/répertoire/nom_du_document) puisque sur FreeNet l'emplacement non seulement n'est pas unique, mais qui peut changer à tout moment.

Chaque utilisateur peut installer son propre noeud FreeNet, pourvu qu'il soit doté d'une connexion "permanente", ou du moins, pas trop volatile. L'ADSL étant ainsi considéré comme une connexion "permanente".
Il suffit d'installer un serveur FreeNet sur sa machine. Les implémentations standard des serveurs FreeNet sont des programmes JAVA, mais ce n'est nullement une obligation technique.
Lors de l'installation, vous indiquez au serveur la **bande passante** qu'il est autorisé à consommer, ainsi que **l'espace disque** que vous lui allouez. Une fois le serveur démarré, l'espace ainsi octroyé à votre noeud viendra enrichir l'espace de stockage global de FreeNet.

2.2.2 L'identification des données

Afin de protéger les hébergeurs tous les fichiers sont cryptés. Ainsi, le possesseur d'un nœud ne peut pas connaître le contenu des fichiers stockés sur son nœud. Notons cependant que en France, la possession de matériel illégal (même si on l'ignore) est considérée comme du recel.
Le fichier est crypté grâce à une paire de clés asymétriques (générée aléatoirement) qui caractérisent le propriétaire du fichier. La partie privée de cette clé sert pour signer le fichier (contrôle d'intégrité). La partie publique, concaténée et hashée avec un texte descriptif du fichier donne la clé de recherche. Seul le propriétaire d'un fichier peut le mettre à jour, en (ré)insérant sur le réseau un fichier avec la même clé : les différentes versions sont alors identifiées par un hashage direct de leur contenu (clé CHK). Les anciennes versions des fichiers restent accessible grâce à leur clé CHK, mais étant de moins en moins accédés, elles finiront par disparaître naturellement du réseau.



Actuellement, la fonction SHA-1 (*Secure Hash Algorithm)* sur 160 bits est utilisée pour effectuer cette opération.

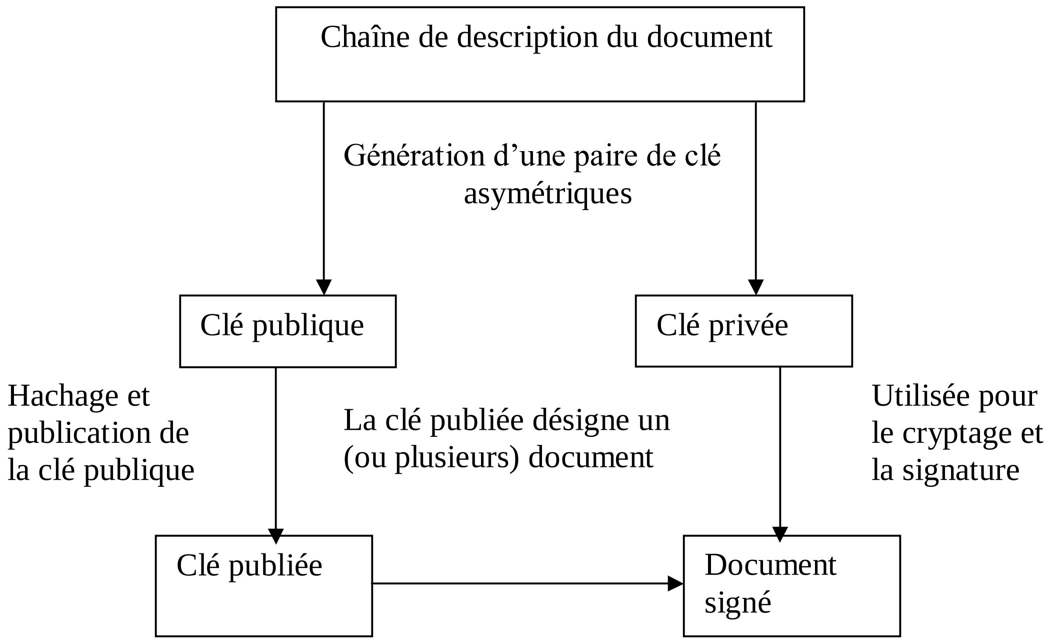
Freenet permet l'utilisation de trois types de clés différentes.

La clé à signature de mot clé (*Keyword Signature Key* ou KSK)

Elle est basée sur une courte chaîne de description, généralement ce sont un ensemble de mots-clés qui peuvent décrire le document (exemple : University/UFR/codage/p2p.doc). Cette chaîne est utilisée comme entrée pour générer de manière déterministe une paire de clés publique/privée.

La moitié publique est alors hachée pour produire la clé du fichier.

Génération d'une clé KSK



La moitié privée de la paire de clés asymétriques est utilisée pour signer le fichier, fournissant un contrôle minimum d'intégrité pour qu'un fichier retrouvé corresponde à sa clé de fichier.

Pour permettre aux autres de retrouver le fichier, l'utilisateur a seulement besoin de publier la chaîne descriptive. Cela rend les clés à signature par mots clés faciles à se rappeler et à communiquer aux autres.
Néanmoins, ils forment un identifiant très banal ce qui est problématique. Rien n'empêche, par exemple, deux utilisateurs de choisir la même chaîne descriptive pour des fichiers différents.

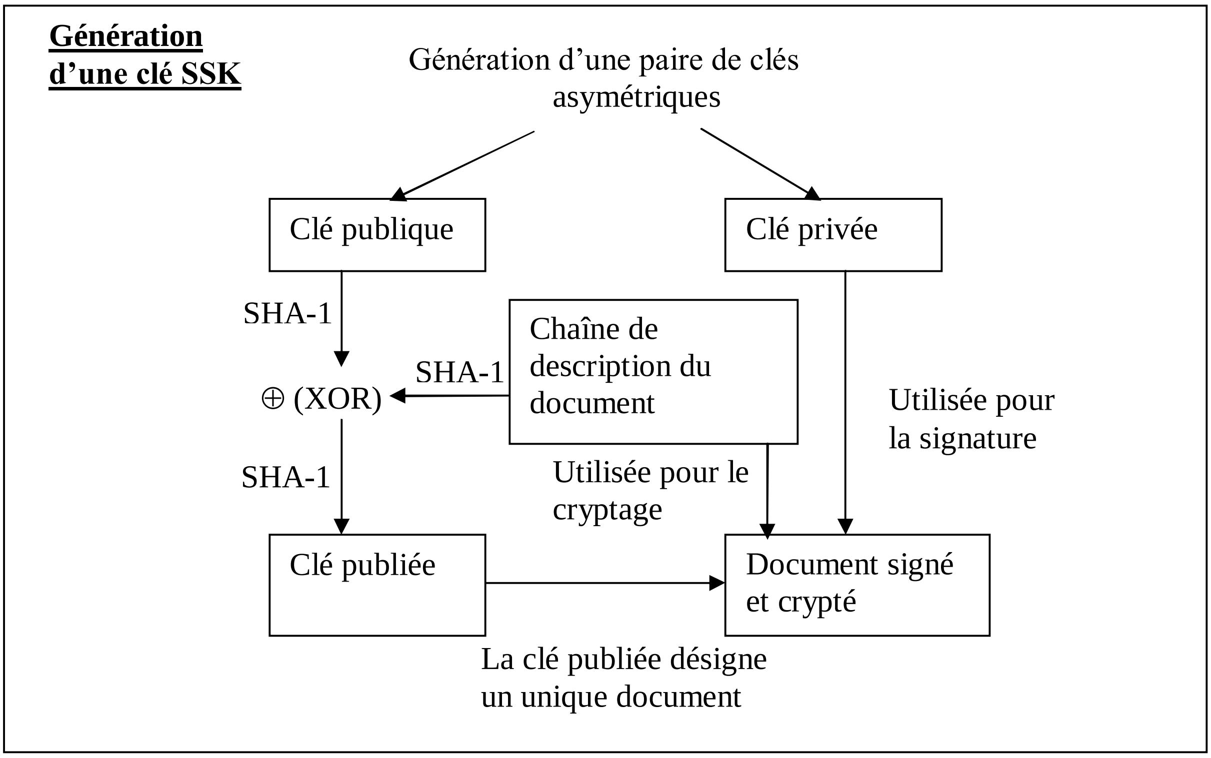
La clé KSK ne suffit donc pas à garantir son unicité au travers de l'Internet, c'est pourquoi on introduit la notion d'espace des noms privée.

La clé à signature d'espace des noms (*Signed Subspace Key* ou SSK)

Un utilisateur crée un espace de noms en générant aléatoirement une paire de clés publique/privée qui servira à reconnaître son identifiant. Pour insérer un fichier, il choisit une chaîne de texte descriptive comme précédemment.
La clé publique de l'identifiant et la chaîne descriptive sont hachées indépendamment, associées avec un XOR, et re-hachées pour produire la clé de fichier.

Comme avec les clés à signature par mot clé, la moitié privée de la paire asymétrique est utilisée pour signer le fichier. Cette signature, générée à partir d'une paire de clés aléatoire, est plus sécurisée que les signatures utilisées par les clés à signatures par mots clés.
Le fichier est aussi crypté par la chaîne descriptive comme avant.

Pour permettre aux autres utilisateurs de retrouver le fichier, l'utilisateur publie la chaîne descriptive avec sa clé publique. Le stockage de nouveaux fichiers dans un sous espace de nom requiert la clé privée.



La clé à hachage de contenu (*Content Hash Key* ou CHK)

Elle voit son utilité essentiellement dans la mise à jour et le découpage des fichiers, permettant ainsi la récupération de différentes parties d'un même fichier auprès de multiples sources. Cette clé est obtenue par hachage du contenu du fichier. Ceci donne à chaque fichier une clé pseudo-unique. Les fichiers sont aussi cryptés par une clé générée aléatoirement.

Pour permettre aux autres utilisateurs de retrouver le fichier, l'utilisateur publie la clé CHK avec la clé de décryptage. Notez que la clé de décryptage n'est jamais stockée avec les fichiers mais est seulement publiée avec la clé du fichier.

Pour mettre à jour un fichier, le propriétaire insère d'abord une nouvelle version sous sa clé CHK, qui doit être différente de celle de la version précédente. Il insère ensuite un nouveau fichier d'indirection sous la clé originale SSK pointant vers la version mise à jour.
Lorsqu'une insertion atteint un noeud qui possède l'ancienne version, une collision de clé apparaîtra. Le noeud contrôlera la signature de la nouvelle version, vérifiant qu'elle est valide et plus récente et remplacera l'ancienne version. De cette façon, les clés SSK mèneront à la version la plus récente du fichier alors que les anciennes versions pourront toujours être trouvées directement par leur clé CHK.

Une **clé FreeNet** est de la forme :
« Type de clé »@« la clé binaire du fichier désiré »

Exemple :
SSK@npfV5XQijFkF6sXZvuO0o kG4wEPAgM/homepage//

Dans cet exemple, **homepage** est l'espace de noms, **SSK** le type de clef, et **npf…AgM** est la clé publique.

2.2.3 La consultation des données et le mécanisme de routage

Chaque nœud d'un réseau FreeNet gère localement ses propres données qui sont disponibles à la fois en lecture et en écriture.
Il dispose également d'une table de routage dynamique contenant les adresses d'autres nœuds ainsi que les clés de fichiers qu'ils sont susceptibles de détenir.

Lorsqu'un utilisateur désire retrouver une donnée, la première chose qu'il doit faire est d'obtenir (sur un site WEB par exemple) ou de calculer la clé binaire correspondant à cette donnée.
Ensuite, le nœud de l'utilisateur va créer un message *DataRequest* qui contiendra cette clé, ainsi qu'une valeur *hops to live* qui correspond à un compteur qui est décrémenté à chaque passage dans un nœud (c'est l'équivalent du *Time To Live* de Gnutella).
Ce message est alors envoyé à l'un des voisins du nœud.

Lorsqu'un nœud reçoit un message *DataRequest*, il consulte ses données locales afin de contrôler s'il est bien le possesseur de la donnée demandée.
Si c'est effectivement le cas, il va retourner au voisin qui lui a transmis le *DataRequest* la donnée en question à l'aide d'un message *DataReply* (message 10 de la figure b).
Si ce nœud ne possède pas la donnée, il va chercher dans sa table de routage une similitude (déterminée par la distance lexicographique, la plus courte, de deux clés) avec la clé binaire du fichier contenue dans la requête et il fait suivre le message *DataRequest* au nœud correspondant.

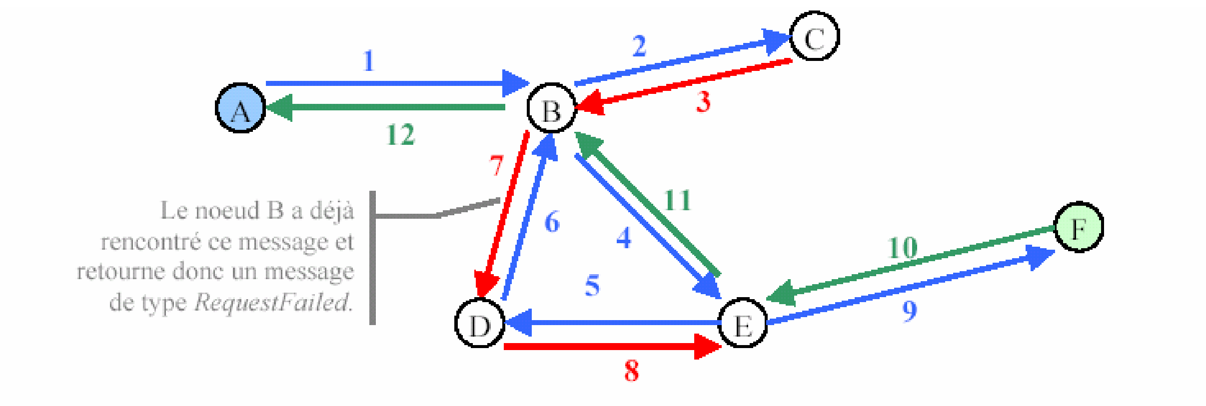


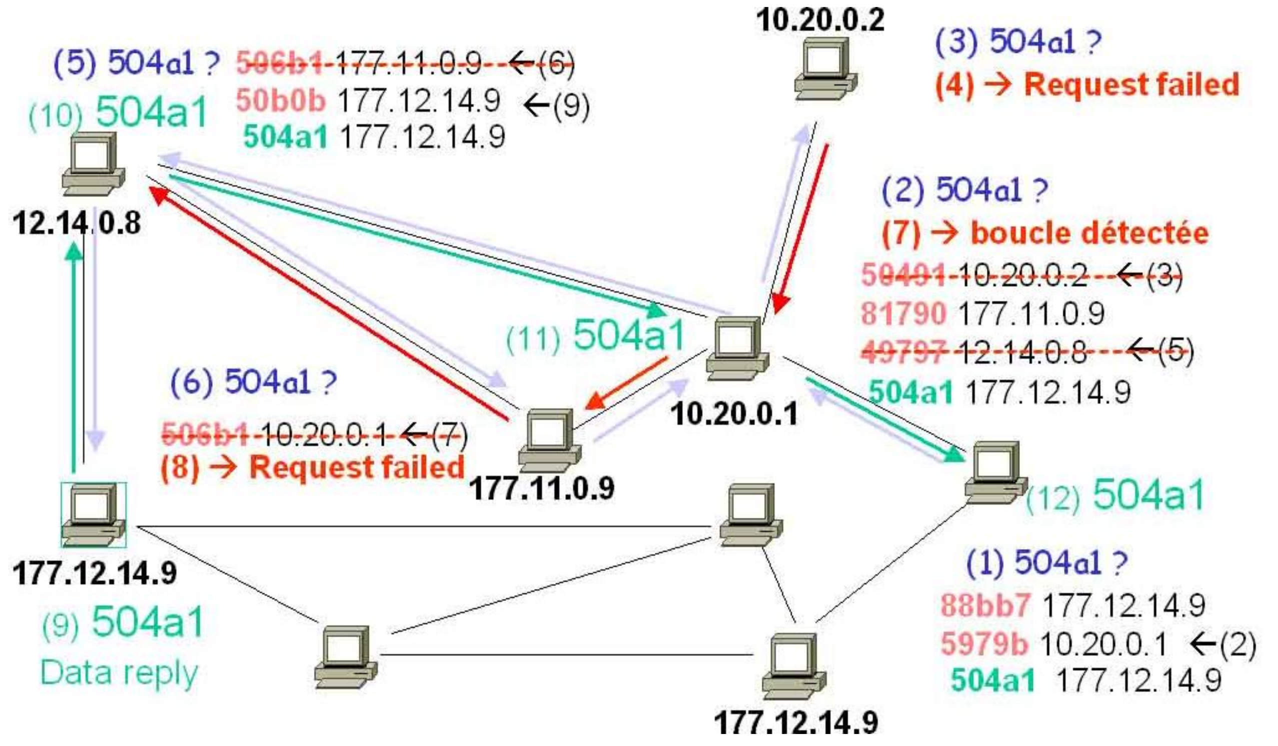
Figure b. Recherche par le nœud A d'une donnée que possède le nœud F. Les liens bleus correspondent aux messages *DataRequest*, les rouges aux *RequestFailed* et les verts aux *DataReply*.

Lorsqu'un nœud reçoit un message *DataReply*, il stocke la donnée contenue dans ce message dans son propre cache local avant de faire suivre le message à son prédécesseur, tout comme il ajoutera dans sa table de routage une nouvelle entrée associant la source de la donnée et la clé de la requête.

Par exemple, sur la figure b, la donnée possédée initialement par le nœud F sera stockée finalement par les nœuds F, E, B et A. Ceci implique que ces nœuds seront plus prompts à répondre à nouveau à une recherche de ce fichier.
Ainsi, dans le cas où le nœud C vient à demander ce fichier au nœud B, ce dernier possédant désormais cette donnée en mémoire cache pourra lui répondre immédiatement sans avoir à transmettre la requête au nœud F.

Un nœud ne peut satisfaire une requête soit parce qu'il ne possède plus de voisins à qui faire suivre le message (comme le nœud C dans la figure b), soit parce que la valeur du *hops to live* du message a atteint 0, soit encore parce que le nœud a déjà reçu la requête. Lorsqu'un nœud se retrouve dans l'un de ces cas, il répond à l'aide d'un message *RequestFailed*. Son prédécesseur enverra donc la requête vers un autre nœud de sa table de routage (comme par exemple les messages 3,7 et 8 de la figure b).

Voici un schéma et quelque phrase qui résume le principe de routage de l'architecture freenet :



Un nœud Freenet possède une table de routage dynamique. Les entrées de cette table associent un identifiant avec un nœud, ainsi lorsque l'on soumet une requête à un nœud, si celui ci ne possède pas le document, il transmet la requête vers le nœud dont la clé dans la table de routage est la plus proche (lexico graphiquement) de l'identifiant du fichier désiré.
Les nœuds ajoutent des entrées dans leur table de deux façons. Lorsqu'un nouveau nœud se joint au réseau, d'autres nœuds contactés aléatoirement déterminent conjointement (et semi aléatoirement) la clé qui sera associée au nouveau dans leur table de routage (ainsi le nouveau nœud recevra les requêtes se rapprochant de cette clé). Ensuite, lorsqu'une requête est soumise, elle est répercutée de nœuds en nœuds en fonction de l'identifiant cherché. La recherche est donc réalisée par un parcours en profondeur du graphe des connexions. Un identifiant unique pour chaque requête (généré aléatoirement) permet de détecter et d'éviter les boucles.(exemple : voir figure : 5.6.7)
Une fois la requête aboutie, la réponse, qui contient le fichier demandé ainsi que l'adresse du nœud qui a fournit la réponse, parcours le chemin inverse. Les nœuds transmettant cette réponse ajoutent alors le nœud d'origine avec la clé du fichier dans leur table (exemple : figure 9 ,10,11,12). Au passage, chaque nœud va copier le fichier en transit dans son propre espace de stockage. Cette copie systématique va permettre aux nœuds de se « spécialiser » et donc de diminuer le temps de réponse aux requêtes. En effet, puisque les recherches sont dirigées en fonction de l'identifiant du nœud, les nœuds, petit à petit, vont être amenés à stocker des fichiers dont les identifiants seront proches de celui que les autres lui ont affecté.

Ce **mécanisme de routage** implique deux choses importantes :

- La première est que les nœuds vont se spécialiser dans la localisation de clés similaires. En effet, lorsqu'un nœud est listé dans une table de routage avec une clé particulière, il recevra un grand nombre de requêtes pour des clés similaires. Ainsi, ce nœud ajoutera dans sa table de routage les nœuds possédant des clés semblables.
- Le second point important est le regroupement des nœuds en "*clusters*" de fichiers à clés similaires.

2.2.4 L'insertion d'un nouveau fichier

L'**insertion d'un fichier** sur le réseau commence par la création de la clé binaire correspondante, puis par l'envoi aux voisins du propriétaire du fichier d'un message *DataInsert* qui contient la clé précédemment calculée. Lorsqu'un nœud reçoit un message *DataInsert*, il compare la clé contenue dans ce message avec celles dont il dispose pour décrire les fichiers stockés localement.

Si cette clé existe déjà, alors il retourne la donnée correspondant à la clé au nœud source du *DataInsert*, exactement comme s'il avait reçu un message *DataRequest* pour cette donnée. Le nœud qui désirait insérer la donnée avec cette clé déduira de cette réponse que la clé existe déjà et devra donc en calculer une nouvelle.
Si un nœud reçoit un message *DataInsert* et qu'il ne dispose pas localement d'un fichier identifié par la clé contenue dans ce message, il va chercher dans sa table de routage la clé la plus proche et fera suivre le message *DataInsert* au nœud correspondant.
Si aucune collision n'a été rencontrée lorsque le message voit son *hops to live* (nombre de sauts) atteindre la valeur nulle, un résultat confirmant l'insertion de la donnée sera alors envoyée en direction du nœud source du *DataReply*.
La donnée nouvellement insérée sera enfin propagée le long du chemin établit par la requête d'insertion. Tout comme le système de routage, le processus d'insertion a différents effets sur le comportement du réseau.

Les fichiers nouvellement insérés sont stockés sur des nœuds possédant des clés similaires, ce qui renforce le mécanisme de regroupement de clés.

Lorsqu'un nœud se connecte au réseau Freenet, il ne dispose alors que de quelques voisins donnés par le nœud introducteur. Les nœuds introducteurs sont des nœuds possédant une adresse fixe connue de tous et qui permet l'insertion de nouveaux nœuds sur le réseau.
L'insertion de données lui permettra alors de se faire connaître par d'autres nœuds et par conséquent d'élargir sa propre connaissance du réseau.

Une tentative d'attaque visant à supplanter un fichier déjà existant en créant délibérément des collisions par l'insertion de mauvais fichiers sous une clé existante, aura de grandes chances d'échouer et de disséminer le vrai fichier. En effet, lorsqu'une collision apparaît (c'est-à-dire que deux clés identiques sont rencontrées), le fichier déjà présent sur le réseau et qui est définit par la clé en question sera envoyé aux émetteurs des messages d'insertion, propageant ainsi le fichier déjà présent sur le réseau.

2.2.5 L'auto-archivage ou l'auto-destruction des données

Le réseau Freenet peut voir une donnée complètement effacée du réseau si tel est le choix fait par les nœuds. Les données étant effectivement conservées dans des caches (de type LRU, *Least Recently Used*), les données très peu demandées se verront être petit à petit supprimées de ces caches et donc du réseau.

Il est ainsi possible de profiter de ce processus pour mettre à jour une donnée, puisque les fichiers obsolètes ne seront plus demandés et donc retirés des nœuds. Toutefois, si une version plus ancienne d'une donnée continue à être demandée, celle-ci sera conservée sur le réseau.

2.2.6 Les faiblesses et limites

2.2.6.1 le temps de récupération des fichiers

Le temps nécessaire pour trouver un fichier dans Freenet est très honorable. La méthode de routage par clé semble, dans le temps, porter ces fruits en « spécialisant » les nœuds. Mais une fois un fichier trouvé sur un nœud, il doit repasser par tous les nœuds sur le chemin. Même si ces transferts devraient être (dans le futur) tunellés[7], (les nœuds n'attendent pas d'avoir complètement reçu le fichier pour le retransmettre), il n'en demeure pas moins que toutes ces copies ont un coût en temps non négligeables dont il n'est pas fait mention dans l'article.
On peut cependant noter que ce temps de recopie ira en diminuant en fonction de l'intérêt que le fichier suscite.

2.2.6.2 la diffusion des clés

Il est (pratiquement) impossible de retrouver une information dans Freenet sans connaître sa clé. (bien que les auteurs évoque l'improbable possibilité d'une forme d'indexation automatique) : il faut donc que celui qui l'a insérée dans le système dispose d'un moyen pour diffuser cette clé. Les développeurs envisagent la mise en place d'un site Web permettant de retrouver les clés des documents. Mais un tel concept va à l'encontre même de l'idée de totale décentralisation dont est née Freenet. Cependant, il est possible avec la version actuelle de créé un "site web" sur Freenet (donc sans base fixe) pouvant permettre l'indexation du reste.Cela étant dit, il est, pour l'instant, relativement difficile de récupérer des informations depuis Freenet

2.2.6.3 QoS : Attaque possible

La persistance des données n'étant pas assurée, un spam général du réseau avec des données corrompues pourrait possiblement provoquer son effondrement. Si la place totale disponible sur un réseau P2P peut être considérée comme virtuellement infinie, les disques durs locaux qui la composent sont eux, loin de l'être Aussi une attaque ciblé sur une clé précise aurait plus de chance d'aboutir :
En générant aléatoirement un grand nombre de clés, l'attaquant peut obtenir des clés proches de celle du fichier qu'il désire faire effacer du réseau. L'attaquant insère ensuite d'énormes fichiers avec ces clés, qui devraient saturer les nœuds ayant des chances de posséder le fichier ciblé. En réalisant massivement des requêtes sur des clés aléatoires, l'attaquant peut récupérer les adresses de nombreux nœuds à qui adresser des requêtes pour les fichiers corrompus qu'il a précédemment insérés,d'utilisant le système de duplication pour encore diffuser le spam dans tout le réseau.
Pour palier à cette attaque, plusieurs systèmes peuvent être mis en place, par exemple :

- Allouer aux nœuds nouvellement insérés une clé proche de celle d'un des nœuds le plus plein pour répartir la charge.
- Utiliser des agents chargés de surveiller et de répartir de façon plus ou moins homogène les données, comme dans le projet Anthill[16]
- Mettre en place une heuristique de gestion locale de la place moins agressive que le FILO, par exemple en prenant en compte la présence ou non d'une copie d'un fichier dans le réseau avant de le supprimer.
- Fragmenter aléatoirement les copies des fichiers et les disperser dans le réseau pour augmenter les chances de pouvoir les reconstruire ensuite, comme IBP [17]

2.2.6.4 Conclusion

Freenet n'est peut être pas exactement le système de stockage et de diffusion de données idéales. Il y a cependant des idées intéressantes sur la méthode pour propager des données au travers du système en suivant les chemins des requêtes. C'est là finalement le réel point qui le différencie des autres systèmes de partage en vogue actuellement : un nœud n'héberge pas que des données qui intéresse sont propriétaire, il permet aux autres d'y stocker leurs données. D'autre projet, comme Free Haven, propose ce même genre de système avec des approches différentes pour la réplication des données, nécessitant la mise en place de techniques complexes pour leur réalisation. Freenet a l'avantage, par rapport aux autres, d'avoir un système de réplication simple, systématique et relativement efficace mais qui souffre d'un total manque de garantie sur la durée de vie d'un fichier. Ce projet open source bénéficie du soutient de nombreux programmateurs. Leur enthousiaste provoque une dispersion dans les domaines de recherche, ce qui finit par nuire à la cohérence des développements, voir à l'avancement du projet, car il reste encore beaucoup à faire...

2.2.7 Comparaison entre les deux réseaux Gnutella et Freenet

Voici trois tableaux qui résument les principaux différence entre les deux réseaux étudiés :

Gnutella	**Freenet**
►le mot clé de la recherche est le nom de fichier voulu	►le mot clé de la recherche ici est très complexe, en effet c'est un mot crypté par des outils spécifique.
►on peut facilement détruire un fichier	►impossible d'effacer un fichier si on n'est pas propriétaire.
►pas de structure pour le réseau,pour intégrer le réseau il suffit d'avoir assez de nœuds actives, pas besoin de maintenir l'ensemble réseau	►une structure dispersée du réseau, pour connaître le réseau il faut apprendre du voisin, un main → éseau constamment
►les nœuds stock seulement leurs fichiers	►un fichier peut être stocker par plusieurs noeuds
►les fichiers ne sont pas cryptés → pas de sécurité	►les fichiers sont cryptés → sécurité absolu
►la source est connue par le destinataire → absence de l'anonymat	►la source est inconnu par le destinataire → anonymat complète
►les broadcast 'ping' sature le réseau	►pas de diffusion, mais de bouche à oreille

Nom	***Répertoire***	***Cache***	***Stockage***
Gnutella	Non	Non	Non
Freenet	Oui	Oui	Non

caractéristique	Type de réseau	Gnutella	Freenet
Type de recherche		Nom du fichier	Chaîne descriptive
Mécanisme de recherche		Propagation aux nœuds voisins	Propagation aux nœuds voisins guidée par le critère de la clé
Efficacité de la recherche		Faible (limité par le voisinage)	Dépend de la proximité sémantique de la clé
Anonymat		Non	Oui
Cache automatique		Non	Oui
Passage à l'échelle		Recherche efficace	Recherche efficace+ cache du contenu

Remarque :

Comme cette technologie est en cours de construction, alors il n'y ai pas de publication, donc on a pas pu trouver les documents nécessaire pour faire une recherche approfondie. Mais on comme même pu faire un résumé sur cette dernière.

2.3 Technologie Fasttrack

FastTrack est une arrivée récente à la scène de pair-à-pair et avec son venir il apporte un nouveau, scalable, l'architecture qui suit toujours une conception décentralisée. Le protocole de FastTrack est actuellement employé les deux applications, KaZaA et Morpheus. L'application de KaZaA a eu vers le haut de 20 millions de téléchargements et KaZaA peut avoir n'importe où jusqu'à 800.000 utilisateurs reliés en même temps.

L'architecture de FastTrack suit un système 2-tier dans lequel la première rangée se compose des raccordements rapides au réseau (Cable/DSL et se lève) et la deuxième rangée se compose des raccordements plus lents au réseau (modem et plus lent). Des clients sur la première rangée sont connus pendant que SuperNodes et clients sur la deuxième rangée sont connus comme noeuds. Lors du raccordement au réseau ce qui se produit est que le client décide si vous convenez pour devenir un SuperNode ou pas. Si vous pouvez devenir un SuperNode vous vous reliez à l'autre SuperNodes et commencez à prendre des raccordements des noeuds ordinaires. Si vous devenez un noeud vous trouvez un SuperNode qui vous permettra de se relier à eux et de vous relier. Ceci produit une topologie à deux niveaux dans laquelle les noeuds au centre du réseau sont plus rapides et produisent donc une épine dorsale plus fiable et plus stable. Ceci permet à plus de messages d'être conduits que si l'épine dorsale étaient plus lente et permet donc un plus grand scalability.

Le cheminement sur FastTrack est accompli par l'annonce entre le SuperNodes. Par exemple, quand un noeud publie une demande de recherche au SuperNode il est relié à la demande de recherche est pris par ce SuperNode et a puis annoncé à tout le SuperNodes qu'il alternativement est actuellement relié à. La recherche continue de cette façon jusqu'à ce que sa TTL ait atteint zéro. Chaque SuperNode qu'elle atteint des recherches un index qui contient tous les dossiers de ses noeuds reliés. Ce les moyens, celui avec TTL de 7 et avec une quantité moyenne de noeuds par SuperNode de 10 une demande de recherche rechercheront 11 fois plus de noeuds sur un réseau de FastTrack que sur GNUtella. Malheureusement puisque les recherches sont émission le réseau produit immobile

d'énormes quantités de données qui doivent être passées de SuperNode à SuperNode. Cependant puisque le SuperNodes sont garantis pour être raisonnablement rapide il ne produit pas aussi grand un problème que sur GNUtella.

Le cheminement des réponses suit les mêmes lignes que GNUtella. Des réponses sont conduites en arrière le long du chemin qu'elles sont venues de jusqu'à ce qu'elles atteignent les clients qui les ont à l'origine publiés. Un grand problème avec ce type de conduite dans GNUtella était que les clients faisant vers le haut de son épine dorsale étaient très passagers et reliés et débranchés au réseau très sporadiquement qui a signifié que les paquets étant conduits en arrière le long du chemin qu'ils sont venu pourraient trouver le chemin allé parce qu'un lien dans la chaîne avait débranché. Ce problème se produit moins sur FastTrack comme des clients faisant vers le haut de l'épine dorsale sont garantis pour être plus rapides et plus stables et donc les chemins pour les paquets de retour de cheminement devraient être plus fiables.

Le téléchargement sur FastTrack est le même que sur GNUtella. Une fois l'endroit du dossier a été trouvé le client qui veut le dossier se relie au client qui accueille le dossier et envoie une demande de HTTP du dossier. Le client accueillant le dossier interprète la demande et envoie une réponse de HTTP. Les en-têtes de HTTP utilisés dans FastTrack ont été modifiés pour adapter à l'information supplémentaire telle que des méta-données mais les en-têtes HTTP/1.1 standard sont soutenus qui signifie que des dossiers peuvent être téléchargés des clients de KaZaA par un enchaînement-web-browser tel que l'Internet Explorer ou le Mozilla.

Malheureusement bien que la topologie de FastTrack soit une exécution d'un KaZaA décentralisé exige que tous les clients s'inscrivent à un serveur central avant d'être laissé à relier au réseau qui infirme tous les avantages d'avoir une topologie décentralisée. KaZaA subissent actuellement l'instance judiciaire avec le RIAA (association d'industrie d'enregistrement de l'Amérique) et si elles perdent et sont arrêtées le réseau de KaZaA cessera de fonctionner. L'enlèvement de l'aspect décentralisé de la conception de FastTrack a présenté un point d'échec au réseau.

Bibliographie

Historique :

• Historique peer-to-peer : http://www.figer.com/publications/p2p.htm
• Historique Napster : http://www.ensmp.fr/~00toure/napster/historique.htm

Présentation générale :

• Open-Files, quelques chiffres : http://www.open-files.com/site/news/287.htm
• Openp2p : http://openp2p.com/
• abou.org : http://www.abou.org/p2p/
• Comment ça marche : http://www.commentcamarche.com/
• Indexel : http://www.indexel.net
• 1formatik : http://www.1formatik.com/news_p2p.htm
• 01net : http://www.01net.com/article/189227.html
• Actu Peer to Peer : http://actup2p.free.fr
• Ratiatum : http://www.ratiatum.com/
• Journal du Net : http://solutions.journaldunet.com/
• InfoAnarchy : http://www.infoanarchy.org/
• Les fiches de l'AWT : http://www.awt.be/cgi/fic/fic.asp?fic=fic-fr-T11-1
• Journal du Net , définition du peer-to-peer :
http://solutions.journaldunet.com/0104/010419_p2p.shtml

Architecture réseau :

• Cours de Didier Donsez de l'université de Grenoble : http://www-adele.imag.fr/~donsez/
• Edonkey 2000 France : http://www.edonkey2000-france.com
• Forum de discussion de Kazaa : http://www.kazaa.com
• Peer-to-Peer Technologies and Protocols : http://ntrg.cs.tcd.ie/undergrad/4ba2.02/p2p/
Logiciels et réseaux :
• e-Donkey France : http://www.edonkey2000-france.com/
• e-Mule project : http://www.emule-project.net
• e-Mule France : http://www.emule-france.com
• Lugdunum, un serveur d'e-Donkey : http://lugdunum2k.free.fr/
• Project 100k, le plus gros serveur e-Donkey : http://81.91.66.90/ed2kch/projet100k/
• SETI : http://setiathome.free.fr/
• Kazaa : http://www.kazaa.com
• Gnutella : http://www.gnutella.com
• DirectConnect : http://www.neo-modus.com
• Blubster : http://www.blubster.com
• Kanari : http://www.kanari.com
• Groove Networks : http://www.groove.net
• Freenet : http://freenetproject.org

Livre et revues :

• PC Team du mois de février 2003 : dossier sur le peer to peer
• Articles de Sciences et Vie Micro année 2002
• Peer to Peer: Harnessing the Power of Disruptive Technologies Andy Oram (editor) First Edition March 2001 ISBN: 0-596-00110-X