Les protocoles Peer-to-Peer, leur utilisation et leur détection

Gabrielle Feltin
LORIA - Campus Scientifique BP 239 54509 VANDOEUVRE-les-NANCY Cedex
Gabrielle.Feltin@loria.fr

Guillaume Doyen
LORIA - Campus Scientifique BP 239 54509 VANDOEUVRE-les-NANCY Cedex
Guillaume.Doyen@loria.fr

Olivier Festor
LORIA - Campus Scientifique BP 239 54509 VANDOEUVRE-les-NANCY Cedex
Olivier.Festor@loria.fr

7 Octobre 2003

Résumé

Nous regarderons les caractéristiques des protocoles Peer-to-Peer (P2P) et les principales architectures (centralisées, décentralisées et hybrides). Nous détaillerons plus particulièrement deux protocoles, à savoir Gnutella et Napster. Nous dégagerons ensuite les principaux avantages du P2P (partage des coûts, équilibrage du trafic, passage à l'échelle, tolérance aux fautes ...) qui en font des protocoles aussi utilisés dans les grilles de calcul, le travail coopératif, ...et pas seulement dans l'échange de fichiers mp3, ... Nous aborderons aussi les problèmes à résoudre (sécurité, législation, supervision, routage). Nous engagerons une réflexion sur le filtrage et la détection de ces flux dans un réseau.

Mots clefs

P2P, peer, protocole, sécurité, passage à l'échelle, routage, supervision

1 Introduction

Au début, Internet avait été conçu comme un système P2P [1]:

- Libre échanges des données, n'importe quel ordinateur pouvait envoyer des paquets aux autres, sans firewall, sans translation d'adresse, sans connexion asymétrique ;
- Les applications telnet et ftp, bien que client/serveur étaient ouvertes à tous les utilisateurs ;
- La coopération était le but essentiel, sans spam et sans abus de consommation de bande passante.

Usenet News, DNS, les protocoles de routage (RIP, OSPF) sont des types de vieux modèles P2P.
L'émergence du P2P peut être vue comme une renaissance du modèle originel d'Internet, au niveau application.

Nous connaissons actuellement surtout le P2P par des logiciels comme napster, gnutella, edonkey, et plus récemment BitTorrent, pour les échanges possibles de fichiers de musique ou de film, de manière illicite. Ceci ne constitue qu'une petite face des protocoles et des utilisations de P2P, qui proposent une manière forte et équitable de collaborer en vue d'accroître le potentiel du réseau.

Une définition du peer-to-peer ou P2P ou pair à pair est : "**peer-to-peer** computing is the **sharing of computer resources and services** by **direct exchange** between systems" [2]. Une autre definition est : "**peer-to-peer** refers to a class of systems and applications that employ **distributed resources** to perform a critical function in a **decentralized manner**" [3].
Le P2P modélise donc **l'échange direct** des ressources et services entre ordinateurs, chaque élément étant client et serveur à la fois. Un des principaux avantages est une utilisation maximale de la puissance du réseau, en éliminant les coûts d'infrastructure et en faisant communiquer directement les clients entre eux. Du fait du déploiement massif des ordinateurs représentant un fort potentiel inactif en bordure de l'Internet, le P2P retient de plus l'attention de la recherche, des développeurs et des investisseurs.

Trois points importants sont à rajouter : tout d'abord les éléments constituants d'un réseau paritaire peuvent être de nature différente (PC, PDA, téléphone portable, …) , ensuite ce réseau présente une topologie virtuelle, enfin il est de nature « ad hoc », en ce sens que les éléments constituants peuvent aller et venir et que la topologie de ce réseau n'est pas stable.

2 Le modèle P2P

2.1 Objectifs

Le modèle P2P étant très général, des applications de nature très différentes peuvent l'utiliser, les objectifs sont variés [1] :

- Partage et réduction des coûts entre les différents peers ;
- Fiabilité et passage à l'échelle, l'absence d'élément centralisé pour l'échange des données permet d'accroître la fiabilité en supprimant tout point central de panne et d'améliorer le passage à l'échelle en évitant les goulots d'étranglement ;
- Agrégation des ressources et interopérabilité, en mettant en commun des ressources individuelles comme de la puissance de calcul ou de l'espace de stockage ;
- Accroissement de l'autonomie en l'absence d'une autorité centrale, il est de la responsabilité de chacun de partager ou non des fichiers ;
- Anonymat pouvant être assuré par certaines applications, en utilisant par exemple des algorithmes de routage qui rendent quasiment impossible le pistage d'une requête ;
- Communication ad-hoc et collaborative.

2.2 Taxonomie

2.2.1 Taxonomie des systèmes informatiques

Les systèmes informatiques peuvent être classés en deux grandes catégories, présentés dans la figure1 : les systèmes centralisés reposant sur des mainframes et les systèmes distribués. Ces derniers peuvent être construits selon deux modèles : le modèle client/serveur plat ou hiérarchique et le modèle pair à pair qui peut être pur ou hybride.

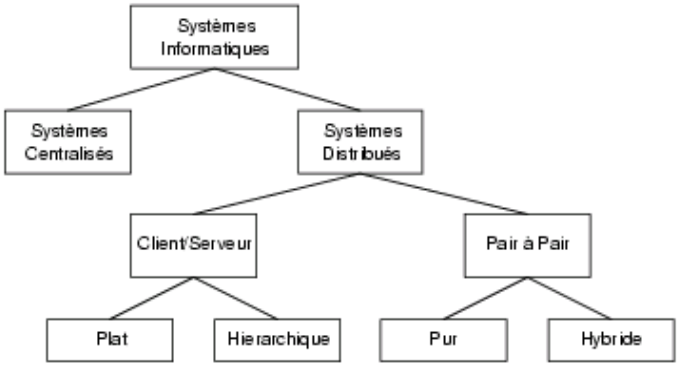

Figure 1 - *Classification des systèmes informatiques*

2.2.2 Taxonomie des systèmes P2P

Dans le modèle pur, il n'existe pas de serveur centralisé : Gnutella, Freenet.
Dans le modèle hybride, un serveur est contacté en premier pour obtenir des méta-informations, comme l'identité du peer sur lequel des informations sont stockées ou pour vérifier des crédentials de sécurité, puis la communication est réalisée en P2P : Napster, Groove, Magi.
Il existe aussi des modèles intermédiaires avec des SuperPeers, comme Kazaa. Les SuperPeers contiennent des informations que les autres peers n'ont pas. Ces autres peers consultent ces SuperPeers s'ils ne peuvent trouver l'information autrement.
L'émergence très récente du modèle P2P rend sa classification assez difficile.
Les autres approches sont la classification applicative (voir paragraphe 5), la classification technologique [4] (stockage des données, contrôle des données, usage de ressources, contrôle de l'état global, contraintes de qualité de service), la classification architecturale et particulièrement les opérations fondamentales [5] (identité, découverte, authentification et autorisation, fonction).

2.2.3 Caractéristiques des systèmes P2P

Les principales caractéristiques des différents protocoles [18]:

- la localisation des fichiers dans un environnement distribué ;
- des méta-données ou index du réseau P2P ;
- la libre circulation des fichiers entre systèmes ;
- un mode de communication standard (TCP et http) ;
- des capacités de connexion variables suivant les modèles ;
- des échanges d'informations non sécurisées ;
- peers non sûrs ;
- aucun peer n'a une vue globale du système.

2.2.4 P2P contre Client/Server

Conceptuellement, le P2P est une alternative au modèle client/serveur, qui est constitué d'un serveur ou d'un cluster de serveurs et de beaucoup de clients. Dans le modèle pur du P2P, il n'y a plus de serveur, tous les participants sont des peers. Cependant il n'y a pas de bordures nettes entre les deux modèles.

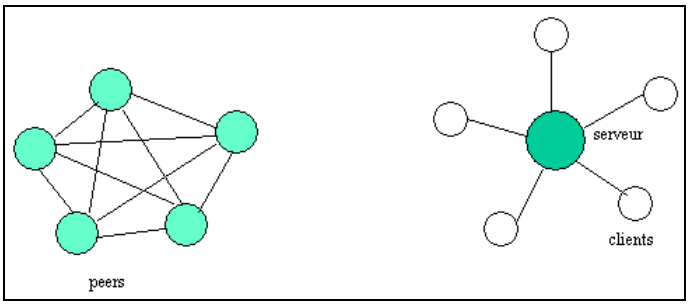

Figure 2 – *P2P versus Client/Serveur*

Peer-to-Peer

- Auto-organisé
- Evolution dynamique, Ad-hoc
- Découverte des peers
- Flux distribué (mesh)
- Symétrie du réseau
- Communication par Messages
- Adressage dynamique au niveau application
- Entités Autonomes
- Attaques difficiles (mobilité , anonymat)

Client/Serveur

- Management centralisé
- Configuré
- Consultation de tables
- Flux centralisé
- Asymétrie du réseau
- Orienté RPC
- Adressage statique @IP
- Entités dépendantes
- Attaques plus simples

2.3 NAPSTER ou le modèle hybride

Napster [12] est une application de partage de fichiers musicaux mp3. Le service est fondé en mai 1999 par Shawn Fanning et Sean Parker. Après différentes poursuites judiciaires pour droits d'auteur, il est arrêté en septembre 2001.
Aujourd'hui Napster est considéré comme une application phare du modèle P2P, par son architecture qui utilise le modèle centralisé et décentralisé et par le nombre de téléchargements effectué (40 millions de téléchargements de fichiers, en moyenne 160 000 utilisateurs).

2.3.1 P2P centralisé ou hybride

Napster utilise un modèle P2P à répertoire centralisé pour les données administratives et les index des fichiers mp3 téléchargeables par les peers.
Un peer se connecte directement sur le serveur. Le protocole basique est simple, avec des primitives de service : login request, login OK, login failed, user unknown, user not accepted, …
Le peer communique au serveur central la liste des fichiers qu'il partage, ainsi qu'un numéro de port TCP où il pourra être contacté pour un téléchargement.

Les peers sont donc les fournisseurs de fichiers, et les échanges vont se faire directement entre les peers suivant un modèle pur.
La recherche et le téléchargement vont se passer de la manière représentée sur la figure 3 [6]:
Un peer recherchant un fichier va envoyer sa requête au serveur central (1), lequel va lui répondre par une liste triée de peers qui hébergent le fichier recherché (2). Le peer va alors choisir parmi les réponses celle qui lui convient le mieux (pertinence de la réponse, bande passante, taille du fichier, …) et contacte directement le peer choisi. Le téléchargement s'effectue alors directement entre les deux peers (3).

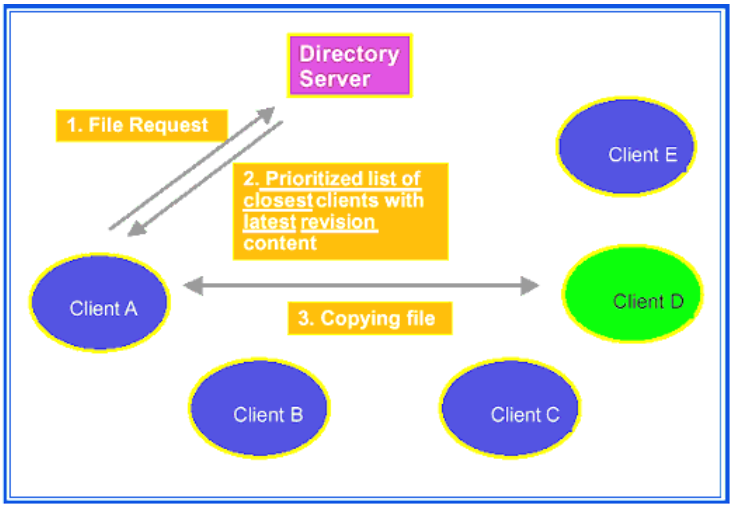

Figure3 - *Les échanges dans Napster*

2.3.2 Les limites

- Pas d'anonymat partout
  - Vous êtes connus du serveur …
  - … et des peers sur lesquels vous téléchargez
- Limites habituelles d'un serveur central
  - Disponibilité
  - Problème du passage à l'échelle :
    - Saturation de la bande passante
    - Saturation du nombreux processus
  - Légal : très facile de fermer le service car localisé
- Mauvaises informations du débit des peers pour ne pas être sollicités

2.3.3 Les avantages

- Avantages habituels d'un serveur central
  - Facile à administrer
  - Facile à contrôler, à fermer
- Evite les recherches coûteuses sur le réseau
  - Pas de routage
  - Planification de la gestion des utilisateurs
- Tolérance aux fautes
  - Par un sondage régulier des peers connectés, état cohérent
- Service novateur
  - Généralise le P2P
  - généralise les procès contre le non-respect des droits d'auteur

Un serveur Napster est toujours connu et son passage à l'échelle parait impossible, vu le goulot d'étranglement du serveur central.

2.4 GNUTELLA ou le modèle pur

Gnutella [7]est un protocole de fichiers partagés. La première version (version 0.4) date de mars 2000, il fut développé en une quinzaine de jours par Justin Frankel et Tom Pepper. Les applications qui implémentent le protocole Gnutella autorisent les utilisateurs à rechercher et charger des fichiers sur tous les autres utilisateurs connectés à la communauté Gnutella, avec plusieurs dizaines de milliers d'utilisateurs simultanés. Il existe beaucoup d'implémentations compatibles apportant des extensions : Limewire, ToadNode, BearShare.

2.4.1 Terminologie

Comme Gnutella repose sur une architecture réseau complètement décentralisée, chaque élément joue le rôle à la fois de client et de serveur, d'où la notion de **servent** qui est la contraction des mots **serv**eur et cl**ient.**
Les **messages** sont les informations émises par les servents à travers le réseau Gnutella.

2.4.2 Le protocole

Gnutella repose sur un modèle pur. C'est un protocole qui fonctionne au-dessus de TCP/IP.
Un servent dispose de 6 types de messages principaux :

- Ping : Utilisé pour découvrir les autres servents sur le réseau. Un servent qui reçoit un Ping doit répondre avec un ou plusieurs Pong ;
- Pong : C'est la réponse à un Ping; ce message contient l'adresse IP et le port du servent qui répond, ainsi qu'une information sur les fichiers et leur tailles partagés par celui-ci ;
- Query : Message pour la recherché de fichier sur le réseau. Un servent qui reçoit un Query répondra par un Query Hit s'il trouve une correspondance avec un ou plusieurs de ses fichiers partagés et les critères de recherche ;
- Query Hit : C'est la réponse à un Query (Figure 8) ;
- Get : téléchargement des données (Figure 9) ;
- Push : Permet de télécharger des données depuis un servent qui se trouve derrière un Firewall .
  Remarque : Si les deux servents sont derrière un firewall, le téléchargement est impossible.

Les messages ont un en-tête commun (figure 4) :

- Payload Type : type du message (Ping, Pong, …) ;
- Hops : nombre de servents traversés ;
- Payload lenght : longueur du message.

	Message ID		Payload Type	TTL	Hops	Payload length	
Octet	0	15	16	17	18	19	22

Figure 4 – En-tête message Gnutella

2.4.3 Routage des messages

La première opération réalisée par un servent est la connexion à un autre servent connu du réseau Gnutella (figure 5), chaque servent possédant une liste de servents à utiliser au démarrage.

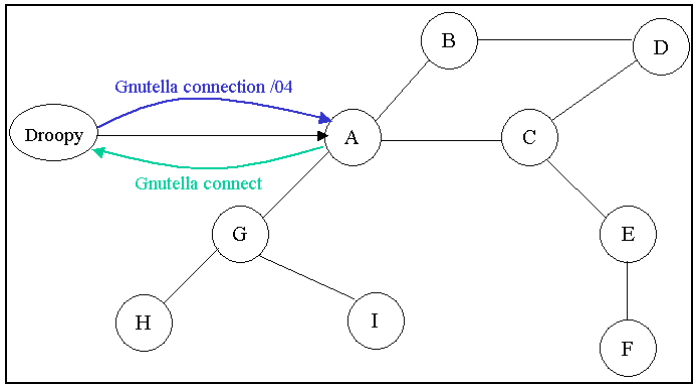

Figure 5 – *Connexion d'un Peer Gnutella*

Un nouveau peer s'annonce lui-même sur le réseau en envoyant un ping Gnutella (figure 6).
Afin de diffuser les messages de requête, Gnutella utilise un mécanisme de flooding.
Le ping est propagé à ces voisins, qui envoient eux-mêmes des pings.
Tous les voisins répondent par un pong.

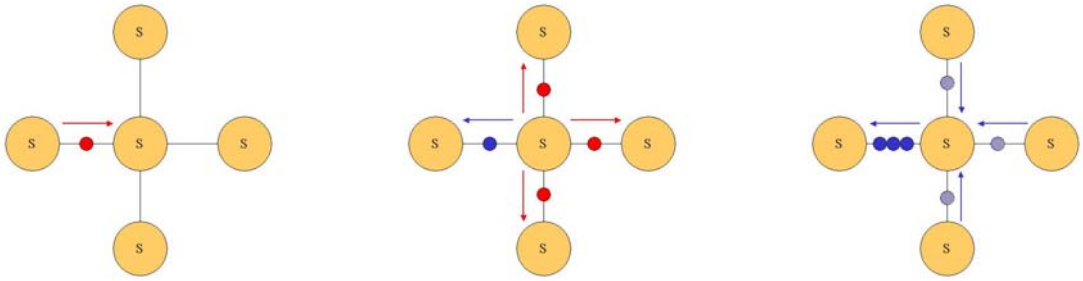

Figure 6 - *Routage des messages*

Des champs de TTL [1] et d'identification des messages évitent la diffusion de messages zombies ainsi que l'apparition de boucles. Chaque requête ping contient un TTL, qui est décrémenté par chaque voisin. Il est souvent positionné à 7 au départ. Quand le TTL passe à zéro, la propagation du ping s'arrête, le message est droppé (figure 7).

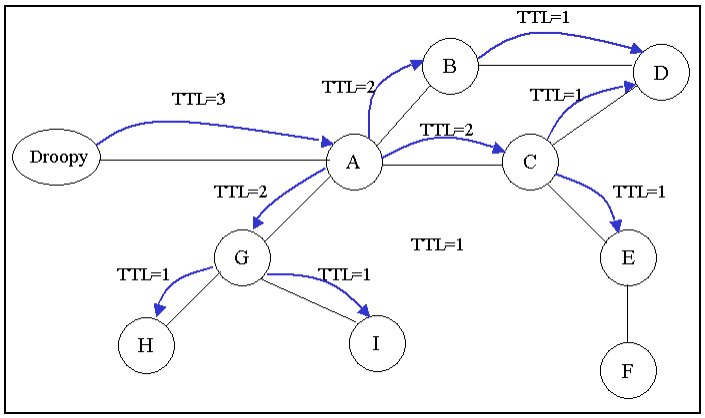

Figure 7 – *Mécanisme de TTL*

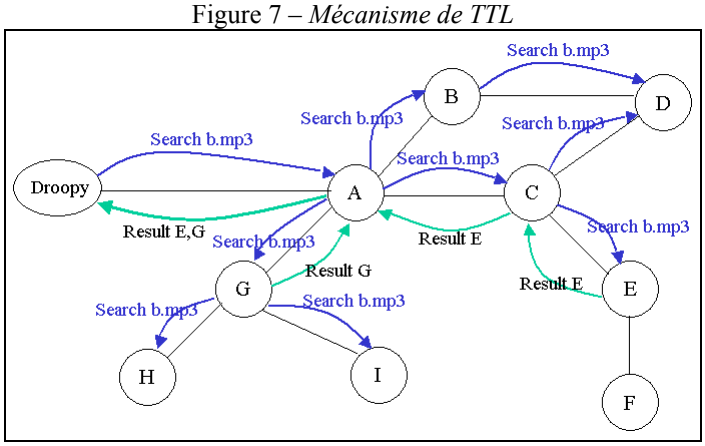

Figure 8 – *Recherche de données (Query-Query Hit)*

Bien que le réseau supporte 8000 à 10000 usagers, son mécanisme de flooding amène des limites, et ne semble pas supporter le passage à l'échelle :

[1] Time To Live

- Réseau rapidement inondé par des Ping et des Pong ;
- Supporte mal une forte montée en charge du nombre d'utilisateurs ;
- Message Push continuellement envoyé si pas de réponse.

Son mécanisme de recherche présente une limite, fixée par la valeur initiale du TTL. Ainsi une requête peut être stoppée par une expiration de TTL sans avoir parcouru l'intégralité du réseau et retourner une réponse négative. Seulement 25% des requêtes aboutiraient.

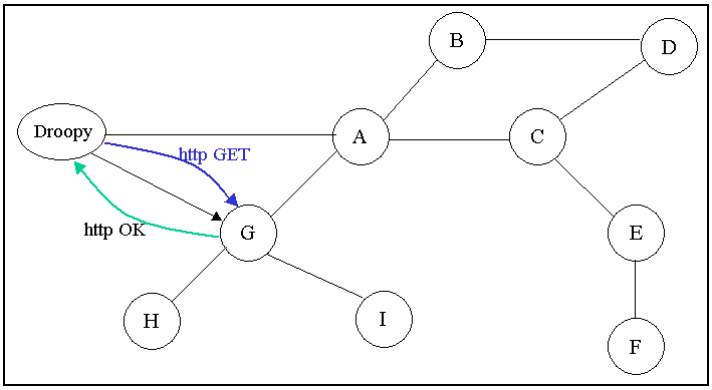

Figure 9 – *Echange de fichiers*

2.4.4 Les avantages

- Administration simple mutualisée
- Topologie évolutive
- Disponibilité du réseau, on ne peut l'arrêter.

Gnutella semble ne pas supporter un passage à l'échelle, et manque de fiabilité dans ses requêtes.
Une orientation possible est les super-peers ou arbre de peers comme dans le protocole en cours de définition Gnutella Version 0.6 ou comme pour Kazaa.

3 Quelques axes de recherche

3.1 Localisation, routage et partage de fichiers

Si, contrairement à Napster, Gnutella offre une solution de partage de fichiers inaccessible a toute autorité qui tenterait de la contrôler, elle présente quelques inconvénients majeurs : tout d'abord les requêtes propagées au sein de la communauté peuvent échouer par simple expiration du TTL. Ainsi, un utilisateur désirant télécharger un fichier particulier effectivement disponible au sein du réseau virtuel peut se voir retourner une réponse négative de la part de Gnutella et doit alors réitérer sa requête ultérieurement en espérant que la topologie virtuelle évolue en sa faveur ou alors qu'un nouveau servent situé dans sa zone de couverture partage ce fichier. La conséquence directe de l'utilisation du TTL est que Gnutella supporte assez mal le passage à l'échelle; avec un TTL statique fixé à 7 Gnutella a du mal à satisfaire des requêtes dans un réseau pouvant comporter plus de dix milles usagers simultanément.
En vue de palier ces problèmes, de nombreux travaux de recherche ont été menés afin de fournir des solutions P2P de stockage et recouvrement de données fiables. En outre, les contributions actuelles scindent le problème de localisation et routage dans un environnement distribué P2P dynamique où les nœuds peuvent aller et venir d'une manière imprévisible de celui de l'application de partage de fichiers a proprement parler.
Cette section décrit les principales architectures P2P qui permettent de construire des applications de partage de fichiers fiables. Nous présentons tout d'abord les deux méthodes principales et routage et localisation de données P2P. Dans un second temps, nous détaillons les principales applications P2P de partage de fichiers fondées sur les précédentes méthodes.

3.2 L'algorithme de Plaxton

Plaxton [13] propose un algorithme de localisation d'objets et de routage dans un environnement de type overlay complètement distribué. Une de ses particularités repose sur le fait que la localisation et le routage sont fait en même temps, réduisant ainsi la charge de ce procédé. Dans sa description, Plaxton distingue plusieurs entités qui peuvent être des serveurs hébergeant des objets, des clients effectuant des requêtes sur ceux-ci et des routeurs qui relaient les requêtes. Dans le cadre des réseaux P2P, nous considérons qu'un peer peut prendre simultanément ces trois rôles de manière indifférente. Le

nommage des objets et peers est construit de manière aléatoire ou pseudo aléatoire, selon une fonction de hachage unique et connue, et chaque identifiant est écrit dans une base identique à chaque objet et peer et possède une longueur fixe.

3.2.1 Routage

Concernant le routage, chaque peer P maintient une table de routage scindée en différents niveaux. Chaque niveau n de la table contient une liste de voisins de P dont les n premiers chiffres sont communs à ceux de P. Un exemple de routage est présenté sur la figure 10. Ici, chaque noeud possède un identifiant écrit sous forme hexadécimale d'une longueur de quatre chiffres. Un noeud d'identifiant 0325 fait parvenir un message à un noeud d'identifiant 4598. Il va alors consulter le premier niveau de sa table de routage et y chercher un voisin dont le dernier chiffre est 8 (ici le noeud B4F8) puis lui faire suivre ce message. A son tour, ce nœud va consulter le second niveau de sa table de routage et chercher un voisin dont l'identifiant se termine par 98 (ici 9098) et lui faire suivre le message. Ce processus se répète jusqu'à l'atteinte du nœud considéré. Cette méthode de routage garantit qu'un noeud présent dans un système de N peers peut être joint en $\log_b(N)$ itérations, avec b, la base des identifiants.

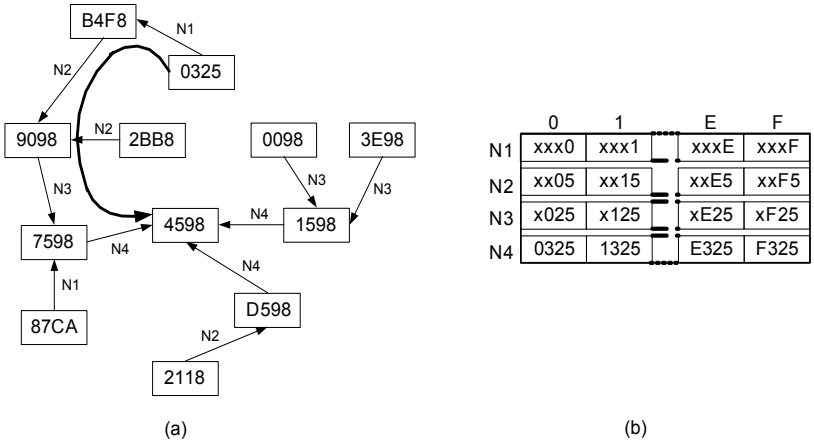

Figure 10 – *Routage dans l'algorithme de Plaxton*

3.2.2 Localisation

Le mécanisme de localisation de Plaxton permet à un peer, de localiser et d'envoyer des messages à un objet particulier situé sur un peer, serveur, distant. Pour cela, un serveur S publie le fait qu'il possède un objet O au noeud racine de O. Tout au long du chemin emprunté pour acheminer ce message, chaque peer conserve un pointeur vers le serveur de l'objet sous la forme d'un couple (ObjetId, ServerId). Un noeud racine d'un objet est un simple pointeur vers le serveur de l'objet.
En considérant ce mécanisme de publication, le traitement d'une requête de localisation est effectué de la manière suivante : Lorsqu'un noeud désire localiser un objet O, il émet une requête qui est propagée vers le noeud racine. Si un noeud intermédiaire possède un pointeur vers le serveur de l'objet, la requête est redirigée vers celui-ci, sinon, c'est le noeud racine qui se charge de cette opération.

3.3 Utilisation d'une topologie connue

La seconde méthode générale de routage et localisation déployée dans les architectures P2P consiste à organiser la topologie virtuelle en vue de la faire correspondre à une topologie connue (Hypercube, anneau, tore, …). Dans ce contexte, chaque topologie présente des méthodes de nommage, routage, localisation qui lui sont propre et ne peuvent être généralisées. Ainsi, nos présentons ici Chord, une infrastructure P2P de localisation et routage reposant sur une topologie en anneau.
Chord est un protocole de recherche P2P pour les applications Internet : pour une clé donnée, Chord y associe un noeud. Chord est déployé dans plusieurs applications : tout d'abord, dans CFS (Collaborative File System) qui est un système de fichiers distribué à l'échelle de l'Internet, ensuite dans ConCHord qui utilise CFS afin de fournir une infrastructure distribuée pour la délivrance de certificats de sécurité SDSI (Simple Distributed Security Infrastructure) et enfin dans DDNS (Distributed Dnomain Name System) qui propose une implémentation P2P du DNS (Domain Name Service).

3.3.1 Le protocole Chord

Chord [14] repose sur une topologie en anneau ; un noeud Chord a la connaissance de son prédécesseur et du noeud suivant. Une fonction de hachage régulière génère une clé pour chaque nœud à partir de son adresse IP. Ensuite, chaque noeud est placé dans l'anneau de manière à ordonner les clés par ordre croissant. Ainsi, chaque noeud Chord est responsable de l'intervalle de clés `[clé(noeud actuel), clé(suivant)[`.

Soit un anneau contenant N noeuds. La table de routage maintenue par chaque noeud N s'exprime en O(log(N)). En effet, la simple connaissance de son prédécesseur et de son suivant permet certes de construire une topologie en anneau mais présente des performances médiocres en terme de nombre de noeuds à parcourir pour acheminer une requête ; cette valeur pouvant atteindre N-1 pour un noeud de clé n envoyant une requête de clé `précèdent(n) - 1`. La figure 11.a illustre ce type de problème avec un anneau contenant 10 noeuds avec une plage d'adressage comprise dans l'intervalle [0, 64[. On peut constater qu'un noeud de clé N8 désirant effectuant une requête de clé K54 s'effectuera en 8 sauts.

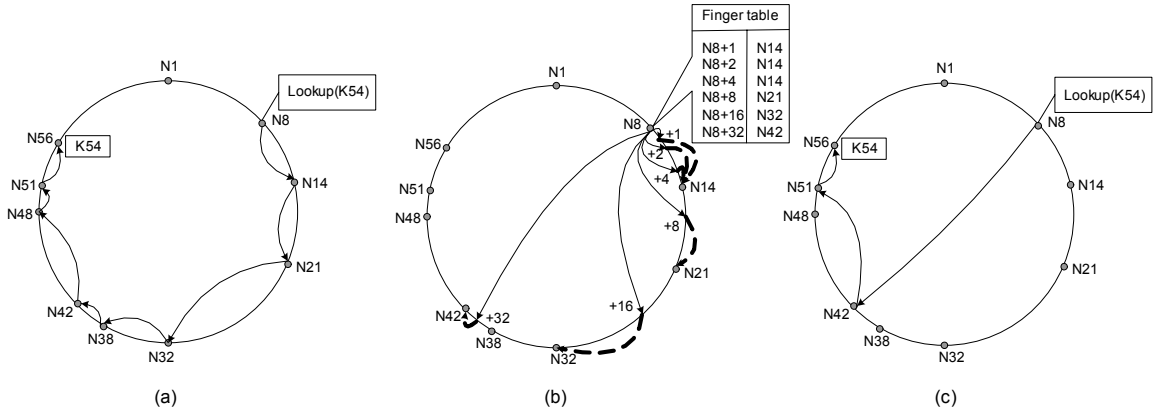

Figure 11 – (a) Acheminement d'une requête par parcours de l'anneau. (b) Définition des fingers d'un nœud. (c) Acheminement d'une requête en utilisant les fingers.

Afin de palier ce problème, pour un espace de clés compris dans l'intervalle [0, 2^m[, chaque noeud n se voit doté d'une entrée vers les noeuds (appelés fingers) de clé suivant($n + 2^{i-1}$) avec $1 < i < m$. Ainsi, nombre maximum de nœuds parcourus pour acheminer une requête est alors exprimé en terme de O(log(N)). La figure 2.b présente la table de routage du noeud N8 muni de fingers et la figure 2.c montre qu'une requête de clé K54 est cette fois acheminée en 3 sauts.

3.3.2 1.2.2. Passage à l'échelle

Chord supporte bien le facteur d'échelle. Différentes simulations ont été menées et montrent que la longueur moyenne d'un chemin utilisé pour acheminer une requête évolue en O(log(N)). Par exemple pour un anneau Chord contenant $N = 2^{12}$ noeuds, la longueur moyenne avoisine 6. Ensuite, un noeud Chord maintient $O(\log_2(N))$ informations dans sa table de routage. Ainsi, pour un anneau contenant $N = 10^6$ noeuds, un noeud maintiendra 400 entrées.

3.3.3 Performances

La garantie de performances de Chord repose sur trois aspects : (1) l'utilisation d'une fonction de hachage qui équilibre correctement les clés au sein de l'anneau, qui évite ainsi tout goulot d'étranglement au niveau d'un ou plusieurs noeuds, (2) l'ajout de noeuds virtuels dans chacun des noeuds réels qui permet d'améliorer l'équilibre du trafic, et (3) le choix, pour un noeud donné, de ses fingers en fonction du temps de latence mesuré.

3.3.4 Tolérance aux fautes

Pour gérer la présence intermittente des noeuds et maintenir la cohérence de l'anneau, Chord utilise un processus de stabilisation qui s'exécute régulièrement sur chaque noeud. Pour un noeud donné, ce processus consiste à vérifier la présence et l'identité des nœuds précédents et suivants et, si nécessaire, mettre à jour la table de routage. Afin de tester la robustesse de Chord, l'expérience suivante a été menée : Pour un anneau de N = 1000 noeuds, une fraction de [0, 0,5[noeuds a été retirée aléatoirement. On montre alors que la longueur moyenne d'un chemin passe de 3,84 à 5,09 et que, pour 10 000 requêtes générées aléatoirement, le nombre de requêtes qui échouent passe de 0 à 5,1. Cette expérience montre que Chord présente une bonne tolérance aux fautes.

3.3.5 CFS

CFS [15] est une application P2P de stockage de données à grande échelle. Son architecture repose sur Chord pour le routage des messages et sur DHash (Distributed Hash) pour assurer la fragmentation et le recouvrement de données. En effet, contrairement à d'autres applications comme PAST ou OceanStore, CFS sépare les données à stocker en blocs de petite taille. Ceci permet en outre d'optimiser l'utilisation de l'espace de stockage en permettant à des peers présentant un faible espace de stockage de participer au service en n'hébergeant que quelques blocs, d'améliorer la sécurité des éléments

stockés face aux attaques qui tenterait de rendre indisponible un document et d'équilibrer le trafic circulant sur le réseau. Un système de quota permet de limiter le nombre de données publiées par un peer en fonction de l'espace total.

4 Peer-to-peer et supervision

A ce jour, il n'est pas d'application P2P qui présente une interface de supervision explicite. Bien souvent, le minimum de gestion nécessaire au bon fonctionnement d'une application est effectué de manière autonome. Citons par exemple, Kazaa qui auto configure les peers participant au service de manière à organiser sa topologie autour de super peers, CAN qui effectue des mesures de RTT entre peers adjacents virtuellement pour garantir une performance optimale ou PAST qui propose l'utilisation de cartes à puce pour contrôler l'accès des usagers à des données distribuées.

4.1 Gestion des performances

De plus, les quelques travaux de recherche qui sont menés dans le but d'ajouter un aspect gestion à des applications P2P ne consistent pas à offrir une interface de supervision à un éventuel administrateur de domaine, afin que cette catégorie d'application souvent déployée de manière sauvage puisse être contrôlée, mais plutôt à accroître le niveau intrinsèque de qualité d'une application P2P, sans se soucier du contexte dans lequel elle est exécutée. C'est par exemple le cas de *The active virtual peer technology* [16], une infrastructure P2P qui permet d'effectuer de la gestion de performance sur une application de type Gnutella. Celle-ci propose d'utiliser un cœur de réseau P2P actif qui permette de restreindre les messages de signalisation à des domaines clos, de paramétrer le routage en fonction de l'état des liens, de rediriger des requêtes de téléchargement, et enfin de contrôler et adapter la topologie virtuelle.
Dans le même ordre d'idée, on peut citer le projet MMAPPS (Market Management of Peer-to-Peer Services) [17] qui a pour objectif de gérer les communautés P2P en utilisant des techniques issues de la gestion de marché. Ainsi, les applications P2P pourraient être plus couramment déployées dans un environnement où la coopération entre les peers serait constructive et ne mettrait pas en danger le potentiel de la communauté. Pour inciter les peers à contribuer plutôt que de simplement consommer, MMAPPS utilise un mécanisme d'évaluation de la contribution et n'autorise l'accès aux ressources fournies qu'en fonction de celle-ci.

4.2 Contrôle du trafic P2P

Considérons maintenant le coté administration système. De ce point de vue, la supervision d'un service de partage de fichiers P2P s'apparente plus à un contrôle du trafic plutôt qu'à la garantie d'une qualité de service aux usagers. Néanmoins, contrôler ce type de trafic s'avère assez compliqué. En effet, le contrôle du trafic P2P passe tout d'abord par son identification ; et les applications P2P n'utilisant pas un port bien défini, il est ainsi nécessaire d'inspecter les paquets jusqu'au niveau des couches 3 à 7 pour identifier un paquet appartenant à un flux P2P. Ensuite, une fois l'identification d'un paquet P2P effectuée, le contrôle du trafic doit être en mesure de traiter l'ensemble du flux auquel appartient le paquet, quel que soit le sens du trafic. Enfin, d'une manière plus générale, les applications P2P changeant constamment, il est nécessaire de pouvoir répondre à ces changements rapidement et efficacement en s'adaptant aux nouveaux protocoles et en fournissant de nouveaux mécanismes d'identification.
Identifier le trafic P2P permet de lui appliquer un traitement particulier en accord avec une politique de supervision propre. Celle-ci peut consister à contrôler la bande passante P2P parmi l'ensemble d'un trafic, à différentier le traitement effectué en fonction des sources, destinations, la date ou le profil d'un usager ou mettre en place un système de quotas par personne. Néanmoins, ce type de comportement ne peut être déployé que sur des éléments de réseaux capables d'assurer une bonne identification du trafic et un niveau de ressources suffisamment conséquent.

5 Les applications

Une classification [1] reconnue scinde les applications du modèle paritaire en 3 grandes catégories (Figure 12), à laquelle on peut rajouter les plates-formes de développement qui peuvent supporter ces applications.

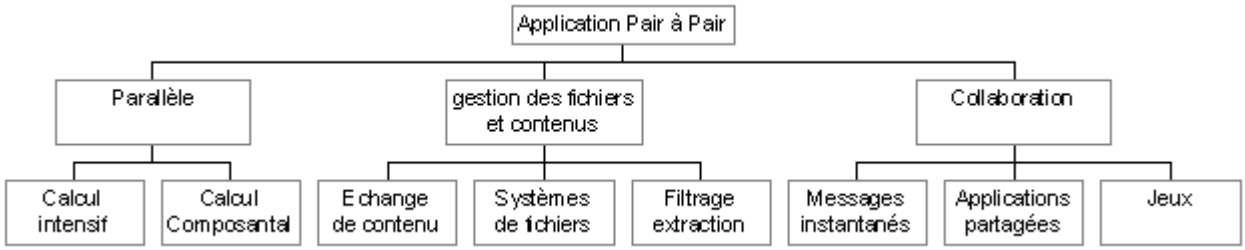

Figure 12- *Taxonomie des applications P2P*

5.1 Les applications parallèles

Ce type d'applications P2P découpe un gros calcul en petites unités indépendantes sur un grand nombre de peers. Une des idées est aussi d'utiliser les machines oisives d'un réseau pour effectuer les calculs. Deux sortes de calcul parallèle peuvent être effectuées :

- tout d'abord le calcul intensif qui consiste à effectuer le calcul d'une même opération munie de paramètres différents, comme par exemple SETI@Home, genome@Home;
- ensuite le calcul composantal qui consiste à découper un même calcul en petites unités indépendantes réassemblables pour effectuer le calcul complet.

Tout le domaine des grilles de calcul se rapproche du P2P pour en utiliser l'architecture, d'ailleurs le Global Grid Forum (GGF) a rejoint en avril 2002 le P2P Working Group (Avaki, Lattice, …). Des travaux portent actuellement par exemple sur le partage de mémoire et d'adressage global à très grande échelle, sur une infrastructure P2P [9].

5.2 La gestion des fichiers et contenus

Ce type d'applications, comme nous l'avons vu au chapitre 2, consiste à stocker et retrouver des informations sur différents éléments du réseau.
L'application principale concerne l'échange de contenu ; elle est fortement connue par Napster, Gnutella, Morpheus, Freenet (anonymat des sources et intégrité des documents), Kazaa et BitTorent, (transfert parallèle de plusieurs sources, possibilité d'arrêt et de reprise d'un transfert).
D'autres projets consistent à établir un système de fichiers distribué dans une communauté, comme PAST, OceanStore (plus de 6 millions d'utilisateurs), CAN ou Chord.
Les bases de données distribuées et les tables de hachage distribuées commencent aussi à utiliser les protocoles P2P (Mariposa).

5.3 La collaboration

Ce type d'applications permet à des usagers de collaborer en temps réel sans utiliser de serveur central.
Une application populaire est l'utilisation des messages instantanés comme Yahoo, AOL, Jabber.
Le principe d'applications partagées ou de travail coopératif est très populaire. Les applications permettant de travailler de manière commune sur un projet distribué sont de plus en plus nombreuses : Groove, Magi, PowerPoint distribué, projet de système de CAO entièrement P2P.
Le commerce électronique commence aussi à utiliser des modèles P2P.
Un dernier aspect de la collaboration : les jeux en réseau (DOOM par exemple), dont l'architecture est exempte de toute autorité centrale.

5.4 Les plates-formes

Elles proposent une infrastructure générique pour développer des applications P2P.
Elles assurent les composants primaires de P2P : gestion des peers, nommage, découverte de ressources, communication entre peers, sécurité, agrégation des ressources.
Les plates-formes les plus connues : JXTA, .net, Anthill.

6 La sécurité et la détection

6.1 Les droits légaux des oeuvres

La mise en ligne d'œuvres protégées sans autorisation vous expose à des poursuites au titre de l'article L 335-4 du code de la propriété intellectuelle.
Le téléchargement de copies illicites vous rend coupable de copies illicites et donc de contrefaçon, voire même de recel pour peu que vous ayez eu connaissance de l'origine frauduleuse des fichiers vu l'article 321-1 du code pénal.
Une première action est une véritable sensibilisation des utilisateurs et des directions à ses problèmes de droits d'auteur et aussi de consommation de la bande passante qui n'est pas gratuite, nous payons tous nos agréments à RENATER.
Les utilisateurs doivent aussi faire très attention aux partages de leur données, un logiciel mal configuré peut exporter tout le disque de leur station de travail sur Internet, voire les dossiers partagés.

6.2 Traitement du P2P

A un moment donné, nous avons la connaissance d'un certain nombre de logiciels P2P, bien souvent servant aux échanges musicaux et cinématographiques. Cependant les nouveaux logiciels apparaissent très rapidement, certains sites peuvent

proposer des œuvres avec le consentement des auteurs, et surtout les logiciels P2P peuvent être utilisés à d'autres fins : chargement de la redhat dernièrement, partage de fichiers bien plus rapide que ftp, travail collaboratif, recherche sur le P2P. Le filtrage n'est pas toujours possible à mettre en œuvre.

6.2.1 Passage des firewalls et filtrage

Rappelons que les logiciels P2P sont conçus pour passer un firewall (Push de gnutella).
Lorsqu'un peer P2 a reçu une réponse positive à une recherche de la part d'un peer P1, situé derrière un firewall, la connexion directe ne pourra être établie. Dans ce cas (figure), P2 initiera le téléchargement par un push ou par une requête au serveur avec son adresse IP et son port (1), ces informations sont transmises au peer P1(2) qui initiera alors la connexion tcp avec P2 (3).On aura donc ouvert une connexion tcp etablish, utilisée alors en retour par P2.
Par contre si les deux peers sont derrière un firewall, le téléchargement est impossible.

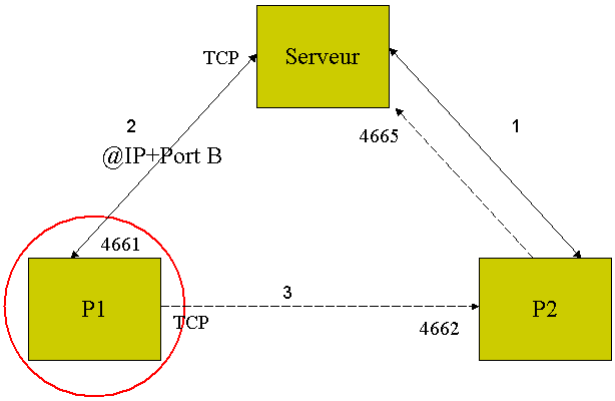

Figure 13 – *Passage d'un Firewall*

On peut filtrer les ports pour ne pas avoir de trafic P2P : en entrée c'est pratiquement toujours fait, en sortie c'est plus difficile car beaucoup d'applications utilisent des ports dynamiques, et il faudrait tout fermer sauf ce dont on a besoin…Citons les principaux ports, un petit digest peut-être trouvé[2] sur le web :

412	Direct Connect	6346 à 6347	Gnutella
1214	Kazaa/Morpheus	6881	Napster/WinMX
6881	Groove	6881 à 6889	BitTorent
4662 à 4665	eDonkey		

Mais fermer tous les ports connus du P2P ne fera t'il pas trouver d'autres solutions comme le tunneling dans icmp ou http ou utiliser des solutions cryptées comme SSH ? Et là, on n'identifie plus aucun trafic.

6.2.2 Détection

L'utilisation d'un IDS (Intrusion Detection System), comme snort [10] par exemple, permet de repérer rapidement certains trafics P2P. Il possède actuellement une base de signatures pour Napster, Gnutella, BitTorent, Fastrack/Kazaa/Morpheus. Par exemple, pour Gnutella il suffit de détecter les chaînes « GNUTELLA CONNECT/0.4<lf><lf> » ou « GNUTELLA OK » ou la détection peut aussi se faire sur un GET, comme le montre la fiche extraite de la documentation snort.

[2] http ://outpostfirewall.com/guide/rules/prest_rules/p2p.htm

SID	1432	message	P2P GNUTella GET
Signature	alert tcp $HOME_NET any -> $EXTERNAL_NET !80 (msg:"P2P GNUTella GET"; flow:to_server,established; content:"GET "; offset:0; depth:4; classtype:policy-violation; sid:1432; rev:4;)		
Summary	This event is generated when activity by Peer-to-Peer (p2p) clients is detected.		
Impact	Informational event. Unauthorized use of a p2p client may be in progress.		
Detailed Information	This event indicates that use of a p2p client has been detected. This may be against corporate policy. p2p clients connect to other p2p clients to share files, commonly music and video files but can be configured to share any file on the local machine. This activity may not only use bandwidth but may also be used to transfer company confidential information to unauthorized hosts external to the protected network bypassing other security measures in place.		
Affected Systems	Any host using a p2p client.		
Attack Scenarios	This is indicative of the use of a p2p client.		

Figure 14 – *Extrait des signatures de snort pour le GET de Gnutella*

Une autre possibilité que nous utilisons fortement est NetMET-Network's Metrology disponible sur le site www.netmet-solutions.org, outil de métrologie basé sur l'analyse des flux avec Netflow.
Une station faisant du P2P a rapidement un fort trafic réseau sortant, bien supérieur aux autres serveurs du site (figure 15).

Host	Cumul (octets)	Cumul (%)	
guest-224.loria.fr	4.7 Go	59.49	Détails
linping.loria.fr	1.2 Go	15.05	Détails
auxerois.loria.fr	691.7 Mo	8.79	Détails
mirabelle.loria.fr	366.7 Mo	4.66	Détails
loria1.loria.fr	346.6 Mo	4.40	Détails
macker.loria.fr	87.6 Mo	1.11	Détails
preny.loria.fr	56.1 Mo	0.71	Détails
noers.loria.fr	38.3 Mo	0.49	Détails
loria2.loria.fr	36.8 Mo	0.47	Détails
152.81.7.6	34.6 Mo	0.44	Détails

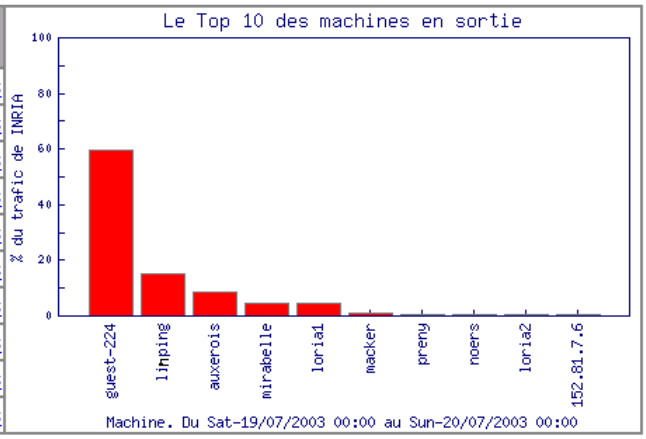

Figure 15 – Le top des machines de l'organisme en sortie par NetMET

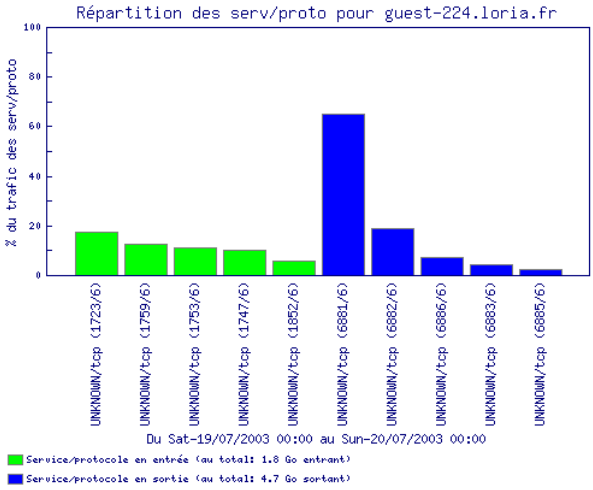

Figure 16 - *Répartition des protocoles pour une machine par NetMET*

Une analyse plus fine permet d'identifier ensuite les ports utilisés (figure 16), à savoir les ports 6881, 6882, … de BitTorent.

6.3 Sécurité d'un réseau P2P

Un réseau P2P étant constitué de peers inconnus et donc potentiellement dangereux, il convient de mettre en œuvre différents mécanismes de sécurité, parmi lesquels on trouve le cryptage des données avec des clés multiples, le sandboxing qui consiste à exécuter une application dans un environnement clos, la gestion des droits d'utilisation qui permet la protection intellectuelle des données, la facturation qui permet d'éviter le phénomène de pure consommation, et enfin la protection des hôtes par des pare-feux.
Au niveau sécurité, IPV6 [11] par ses mécanismes d'authentification, d'autorisation et d'intégrité pourrait pallier le regroupement de machines sous la protection de pare-feu et assurer de la sécurité dans les échanges.

7 Conclusion

Dans cet article, nous avons présenté les principaux protocoles mis en œuvre dans les applications P2P de partage de fichiers. D'un modèle initialement centralisé, les applications ont évolué vers des modèles complètement décentralisés. De plus, de nombreux travaux ont été menés pour fournir des services de routage et localisation fiable et performants dans un environnement distribué et dynamique. En effet, d'un mécanisme de flooding déployé par Gnutella, qui présente de fortes limites en terme de fiabilité, passage à l'échelle et performances, les applications utilisent maintenant des infrastructures reposant sur l'algorithme de Plaxon ou des topologies connues. C'est le cas de PAST, OceanStore, CAN, …
Néanmoins, si de nombreux efforts sont fournis pour améliorer la qualité des applications P2P de partage de fichiers, il n'en est pas de même pour la partie supervision ; les quelques initiatives actuelles de gestion de ce type d'application portent d'avantage sur l'amélioration intrinsèque du service que sur leur intégration propre et transparente dans un environnement de réseau administré ou l'usage des ressources est coûteux et s'intègre dans le cadre d'une politique de gestion.
De plus, l'identification et le traitement particulier de ce type de trafic s'avèrent difficiles car ils nécessitent la mise en place de procédés lourds et sans cesse actualisés. En effet, identifier un flux P2P nécessite souvent de scruter les paquets jusqu'au niveau applicatif et l'apparition continue de nouveaux protocoles P2P oblige les systèmes d'analyse à mettre à jour leur procédé d'identification très régulièrement. C'est pourquoi, dans ce contexte, nous pensons que la meilleure méthode pour gérer ou plus modestement contenir ce type de trafic passe principalement par la sensibilisation des usagers.
En outre, le modèle P2P ne doit pas être systématiquement associé aux applications sauvages de partage de fichiers ; ce modèle est actuellement utilisé dans de nombreux autres domaines d'applications comme le travail collaboratif ou le calcul distribué, avec par exemple, la découverte de la carte du génome humain. Enfin, si l'on considère l'évolution actuelle de l'informatique, avec les réseaux sans fils, la mobilité, l'accroissement continuel de la puissance des machines, mais aussi de la bande passante, et surtout l'aspect ubiquitaire du déploiement de services, il nous semble que, dans ce contexte, le modèle P2P trouve sa place et occupe une part de plus en importante du trafic des réseaux, imposant à ce type applications leur ouverture vers une interface de supervision.

Références

[1] Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, Zhichen Xu, Peer-to-Peer Computing HP, http://www.hpl.hp.com/techreports/2002/HPL-2002-57.pdf

[2] http://www.p2pwg.org

[3] Karl Aberer, Manfred Hauswirth, ICDE 2002, Peer-to-peer information systems: concepts and models, state-of-the-art, and future systems http://lsirwww.epfl.ch

[4] Kant K., Iyer R., Tewari V. , A framework for classifying peer-to-peer technologies, May 2002 2nd IEEE/ACM Intl. Symposium on Cluster Computing and the Grid, http://kkant.ccwebhost.com/papers/taxonomy.pdf

[5] To-peer working group, Taxonomy of peer-to-peer architectures, November 2001, Draft for TAC review, http://www.peer-to-peerwg.org/tech/taxonomy

[6] Corbet I, Pruski C., Geoffroy P. , Les protocoles P2P, Rapport de DESS 2003 - UHP Nancy

[7] http://sourceforge.net/projects/rfc-gnutella

[8] http//groups.yahoo.com/group/the_gdf

[9] GRID'2002 – Ecole thématique sur la globalisation des ressources informatiques et des données - Décembre 202

[10] www.snort.org

[11] www.ipv6.org

[12] Napster messages, open-nap.sourceforge.net/napster.txt

[13] Plaxton C. Greg, Rajaraman Rajmohan, Richa Andrea W., Accessing nearby copies of replicated objects in distributed environment, ACM Symposium on Parallel algorithms and Architecture

[14] Stoica I., Morris R. Karger D., Kaashoek M. F., Balakrishnan H., Chord: A scalable peer-to-peer lookup service for internet applications, proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications, ACM Press, 2001

[15] Dabek F., Kaashoek M.F., Karger D., Morris R., Stoica I., Wide-area cooperative storage with CFS, Proceedings of the 18th ACM Symposium on Operating Systems principles, Canada, 2001

[16] Tutschku K., henjes R., De Meer H., Koulouris T., The active virtual peer technology, www.ee.ucl.ac.uk/netdist/AVP_presentation_web.ppt

[17] The MMAPPS Consortium, Market Management of peer-to-peer services, www.mmapps.org

[18] Festor O., Peer-to-Peer : Principles&Applications, Protocols&Architectures, Cours de DEA UHP-Nancy1