

Sécurité du Peer-to-Peer

Introduction

En quelques années seulement, internet a connu un véritable boom, offrant aux utilisateurs du monde entier un moyen rapide et efficace pour se partager des informations.

L'augmentation considérable du nombre de transfert de fichiers est aujourd'hui de plus en plus important et la taille des fichiers transférés a également été multipliée ces dernières années. Il n'est plus rare aujourd'hui de voir des échanges de plusieurs Giga octets de données sur internet.

Cet accroissement des échanges montre aujourd'hui les limites du vieux modèle client/serveur. Malgré une augmentation considérable de leur puissance de calcul, les serveurs ont de plus en plus de difficultés à répondre aux demandes de tous les clients et on s'oriente de plus en plus vers des architectures N-tiers pour tenter de répartir la charge.

D'un autre côté, on peut remarquer, depuis quatre ou cinq ans, un véritable engouement pour les réseaux d'échange de fichiers. Ces "Peer-to-Peer" représentent aujourd'hui une partie considérable des échanges sur internet. Leur développement est lié en grande partie à l'augmentation des capacités matériels (capacité de stockage, bande passante) des internautes et intéressent de plus en plus les grandes entreprises. Des sociétés comme RedHat utilisent aujourd'hui les Peer-to-Peer pour distribuer leurs produits afin de soulager leurs serveurs (la taille des distributions linux atteignant plusieurs Giga de données).

Les Peer-to-Peer ne se limitent plus aujourd'hui au partage de fichiers et de nombreuses applications, reposant sur le même principe, ont vu le jour comme par exemple des logiciels de téléphonie, ou encore de travail collaboratif.

Le fait d'utiliser des architectures décentralisées ou semi décentralisées a conduit inévitablement à l'apparition de nombreux problèmes de sécurité pour les utilisateurs.

De plus, les protocoles d'échange de fichiers sont souvent considérés comme une violation des droits de propriété intellectuelle et subissent donc des attaques incessantes des sociétés représentant les ayants droits.

Ces difficultés sont à l'origine du développement permanent des Peer-to-Peer. On assiste aujourd'hui à l'apparition de protocoles crypté et "anonyme", etc.

Nous commencerons tout d'abord expliquer ce qu'on entend par "Peer-to-Peer" puis nous analyserons deux des protocoles les plus utilisés aujourd'hui, à savoir eDonkey et BitTorrent. Nous étudierons ensuite les différents problèmes relatifs à la sécurité que l'on peut rencontrer en utilisant des réseaux Peer-to-Peer.

Sommaire

1	Qu'est ce qu'un Peer-to-Peer	4
1.1	Définition	4
1.2	Avantages des Peer-to-Peer	5
1.3	Exemple de Peer-to-Peer	6
1.4	Catégories de Peer-to-Peer	6
1.4.1	Peer-to-Peer structurés	6
1.4.2	Peer-to-Peer non structurés	7
1.4.3	Autre classification	7
1.5	Statistique d'utilisation des réseaux	9
2	Étude du fonctionnement de deux P2P célèbres	10
2.1	eDonkey	10
2.1.1	Fonctionnement général	10
2.1.2	Les serveurs	10
2.1.3	Les clients	12
2.1.4	Fonctionnement détaillé d'un client	13
2.1.5	Les fichiers sur le réseau	16
2.1.6	Liste d'attente de téléchargement	17
2.1.7	Liste des serveurs eDonkey	18
2.1.8	Problèmes d'eDonkey et quelques solutions	18
2.1.9	Kad Network	22
2.2	BitTorrent	23
2.2.1	Principe de fonctionnement	23
2.2.2	Terminologie	24
2.2.3	Création d'un torrent	25
2.2.4	Indexation des torrents et recherche d'informations	27
2.2.5	Téléchargement des fichiers : connexion au tracker	28
2.2.6	Téléchargement des fichiers : connexion aux pairs	28
2.2.7	BitTorrent et les autres P2P	30
2.2.8	Développement de BitTorrent	32
3	Identification des risques	33
3.1	Virus et Spyware	33
3.2	Fichiers truqués	34
3.3	Faibles de sécurité	36
3.4	Spoofing ip : Restrictions des IP / Usurpation d'identité	36
3.5	Utilisation des ressources / DDoS	38
3.6	Actions en justice	40
3.1	Administration difficile	41
3.2	Absence d'anonymat	42
3.2.1	Identification par adresse IP	42
3.2.1	Identification des protocoles	42
3.2.2	Solutions au problème de l'anonymat ?	44
4	Attaques spécifiques aux réseaux P2P	45
4.1	Sybil Attack	45
4.2	Eclipse	45
4.1	Pollution de DHT	46

1 Qu'est ce qu'un Peer-to-Peer

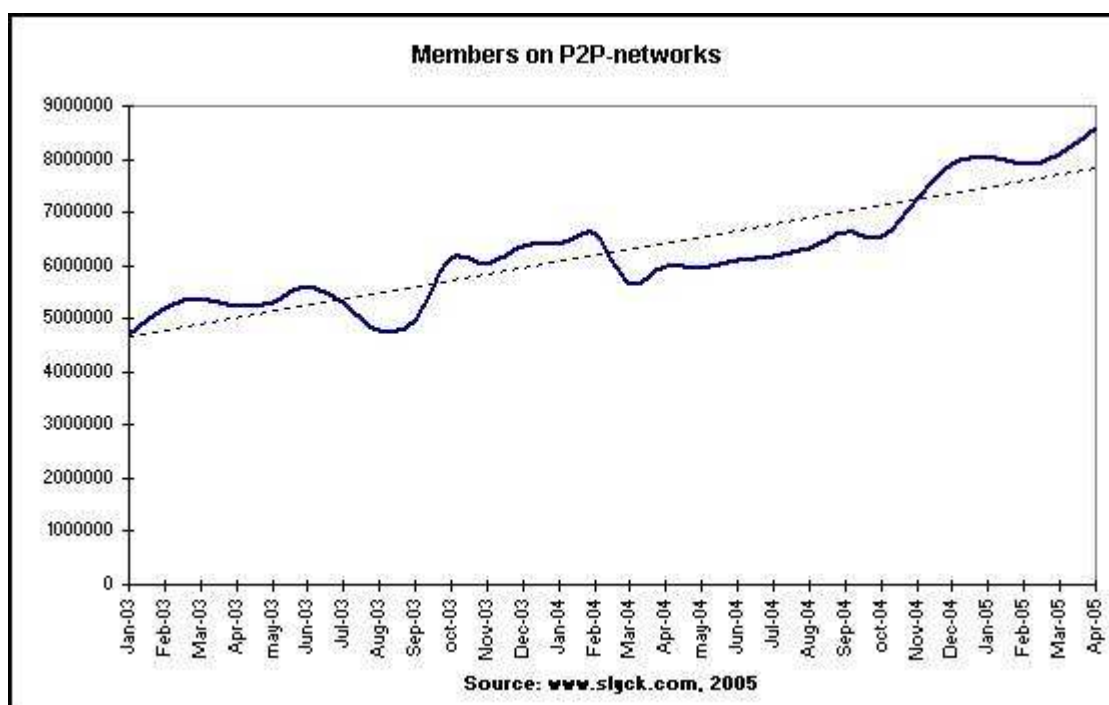
1.1 Définition

Le terme "Peer-to-Peer", "poste à poste" ou "pair à pair" en français (ou plus couramment P2P), désigne un modèle de réseau informatique dont les éléments (les *nœuds* ou "*peer*") sont à la fois clients et serveurs lors des échanges. Ce modèle a permis une décentralisation des réseaux, en offrant une alternative aux traditionnelles architectures client/serveur ou des architectures N-tiers.

Comme leur nom l'indique, les Peer-to-Peer permettent d'établir des communications directes, d'égal à égal, entre les différents nœuds du réseau, qui peuvent alors échanger différents types d'informations sans passer par un serveur central.

Napster, apparu en juin 1999, est considéré comme le premier P2P, même s'il possédait une architecture centralisée. C'est en effet le premier logiciel d'échange à avoir connu un succès mondial.

L'utilisation des Peer-to-Peer est aujourd'hui en pleine croissance et de nombreux logiciels ont été développés. On comptait environ 5 millions personnes téléchargeant des fichiers via les P2P en 2003, 7 millions en 2004 et près de 9 millions en 2005.



Croissance du nombre d'utilisateurs des réseaux Peer-to-Peer

1.2 Avantages des Peer-to-Peer

Dès leur apparition, les P2P se sont distingués des architectures traditionnelles grâce à leurs différents avantages. En effet, ils sont souvent assimilés, par le grand public, aux logiciels de partage de fichiers, cependant leur application ne s'arrête pas là. Ils permettent de mutualiser les ressources de milliers d'ordinateurs dont les capacités sont sous-exploitées par leurs utilisateurs :

- ⇒ **La répartition de la charge** : les échanges sont gérés directement par les paires, ce qui élimine un des principaux problèmes des architectures clients/serveurs : la répartition de la charge. Les P2P ne posent pas le problème de congestion des réseaux qui peuvent se produire autour d'un serveur central devant répondre à de très nombreuses demandes. Chaque pair gère ses propres données et répond aux requêtes des autres pairs.
- ⇒ **La capacité de stockage** : chaque nœud ne possède qu'une infime partie des données du réseau, qu'il partage avec les autres ordinateurs. La capacité de stockage est ainsi infiniment supérieure à celle d'un serveur traditionnel, qui ne pourrait supporter une telle quantité d'information (que ce soit au niveau technique ou au niveau du coût de mise en place).
- ⇒ **La puissance de calcul** : pour les utilisateurs moyens, chaque ordinateur utilise moins de 20% de sa puissance de calcul: "travailler sous Word ou Excel (voir les 2 en même temps) n'occupe le processeur que de manière très faible, le reste étant simplement inutilisé.¹" Une application répartie, s'activant en général avec l'écran de veille des ordinateurs, basée sur la technologie Peer-to-Peer peut donc permettre de mutualiser cette puissance non utilisée pour des recherches demandant des capacités considérable, qu'un serveur isolé ne peut posséder.

Les Peer-to-Peer apparaissent également dans les entreprises qui souhaitent utiliser leurs nouvelles possibilités :

- ⇒ **Outils de travail collaboratif** (gestion d'agenda, etc.)

Exemple : le logiciel "groove²", utilisé dans les entreprises permet le partage des documents d'un projet, de conduire des réunions, assigner des tâches, etc.

- ⇒ **Résistance aux pannes** (sauvegardes croisées) : les données étant présentes sur de nombreux postes différents, les P2P peuvent donc être utilisés pour effectuer facilement des sauvegardes réparties à travers le réseau, évitant toutes pertes d'informations, ou la mise en place de lourds systèmes de sauvegardes.
- ⇒ **Extensibilité** : auto-configuration des pairs. Il est très facile de rajouter de nouveaux pairs. Ceux-ci sont gérés dynamiquement.

¹ <http://blackholesun.neuf.fr/Boinc.html>

² Voir le site de la société Groove Networks : <http://www.groove.net>

1.3 Exemple de Peer-to-Peer

Les Peer-to-Peer sont utilisés dans différents domaines. Le plus connu est le partage de fichier à travers internet. Cependant, diverses applications se sont développées à partir de ce modèle.

On retrouve donc :

- ⇒ Le **partage de fichiers** avec des logiciels comme KaZaA ou emule
- ⇒ Les logiciels de **téléphonie sur Internet** dont le plus populaire Skype.
- ⇒ Des programmes de **messageries instantanées** (ICQ, AIM)
- ⇒ La **Télévision** par P2P: Les fondateurs de Kazaa et de Skype préparent la télévision via le peer-to-peer ("Venice Project")
- ⇒ Des **moteurs de recherche** tels qu'Amoweba (moteur de recherche distribué en P2P basé sur l'utilisation intelligente des liens favoris des internautes).
- ⇒ Des projets permettant le **partage de la puissance de calcul** d'ordinateur du monde entier, tels que le "projet SETI" dont le but est de rechercher des traces d'Intelligence Extra-terrestre.
- ⇒ Des **Plate-forme de développement** telle que "JXTA"³, technologie développée par Sun Microsystems, qui a pour but de pouvoir interconnecter n'importe quel système sur n'importe quel réseau. Elle utilise des protocoles basés sur XML et doit permettre de mettre en relation des ordinateurs, des téléphones portables, des PDA, etc. pour les faire communiquer de manière décentralisée. "Java Binding" est actuellement la version la plus abouti.
- ⇒ Le **partage d'informations** : informations collaboratives pour la détection de spam (cf. Bibliographie : **Spam Attacks: P2P to the Rescue**)

Par la suite, nous nous intéresserons plus particulièrement aux architectures permettant le partage de fichier.

1.4 Catégories de Peer-to-Peer

Les réseaux Peer-to-Peer sont classifiés en deux catégories : les structurés et non structurés.

1.4.1 Peer-to-Peer structurés

Les P2P structurés sont basés sur l'établissement d'une DHT (Distributed Hash Table) permettant de "placer" les nouveaux nœuds aux seins du réseau. Chaque nœud reçoit une liste de voisins avec lesquels il pourra communiquer. Il s'agit ici de voisins "logiques", ceux-ci pouvant se trouver à l'autre bout du monde.

De plus, chaque pair est responsable d'une partie spécifique du contenu du réseau. Ils sont en général identifiés par un couple clé/valeur. Elle permet de réaliser des recherches avec un nombre de messages croissant de façon logarithmique, contrairement à certain algorithme proposant des techniques de "flood"⁴.

³ <http://en.wikipedia.org/wiki/JXTA> / <http://www.sun.com/software/jxta/>

⁴ Flood: (inondation en français) Cette technique permet de propager la requête de proche en proche, chaque peer la transmettant à ces voisins. Il y a donc des redondances, les peer pouvant avoir des voisins similaires.

Parmi les algorithmes structurés, on peut citer:

- CAN
- Chord
- Tapestry
- Pastry
- Kademlia utilisé par exemple par Overnet
- Viceroy
- Freenet

1.4.2 Peer-to-Peer non structurés

Un p2p est non structuré quand les liens entre les nœuds sont établis de façon arbitraire. La communication entre les nœuds est donc plus difficile à gérer.

Parmi ces P2P, on peut citer:

- Gnutella
- Fastrack dont le client le plus connu est KaZaA
- BitTorrent
- eDonkey utilisé notamment par le logiciel eMule

1.4.3 Autre classification

Il est également possible de regrouper ces architectures en 3 grands groupes souvent utilisées pour classifier les réseaux P2P :

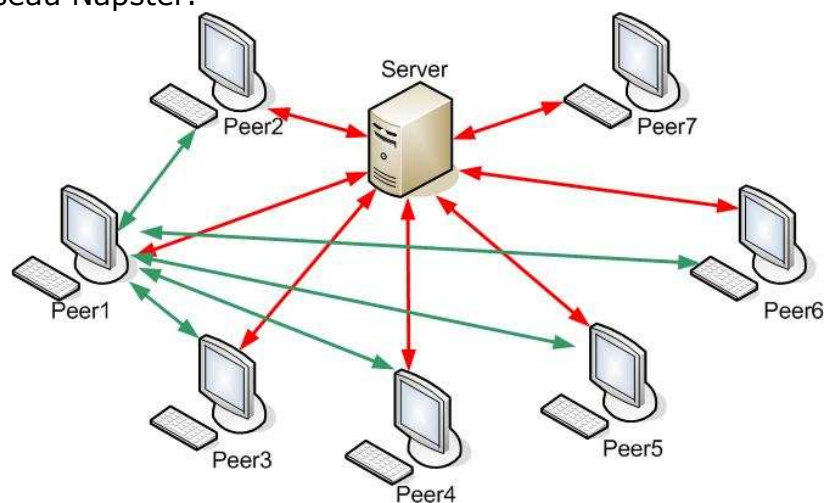
- P2P centralisé :

Il existe un serveur central, gérant l'identité des utilisateurs, l'indexation des partages, les recherches et la mise en relation des pairs : le serveur donne à chaque client l'adresse des pairs possédant le fichier recherché. Le client se connecte alors à ce pair pour échanger directement les données. Les fichiers ne passent donc pas par le serveur.

L'index est donc centralisé, mais les fichiers sont décentralisés.

Ce type de P2P est considéré comme l'architecture la plus faible car il suffit de s'attaquer au serveur central pour faire tomber le réseau.

Exemple: réseau Napster.

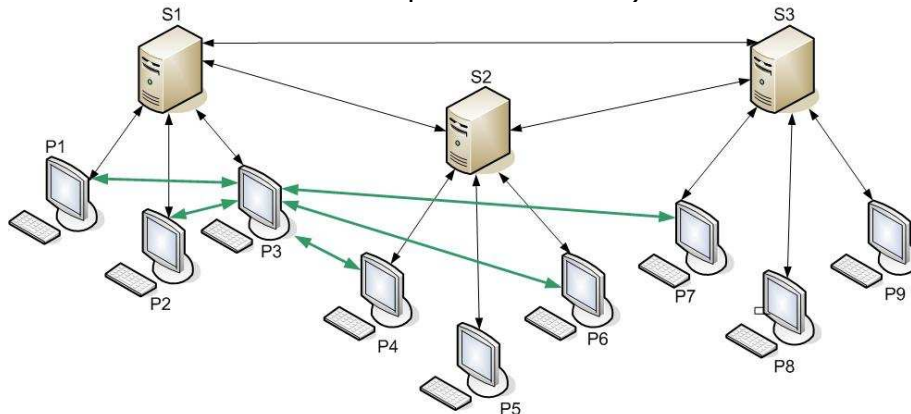


Principe de fonctionnement d'un P2P centralisé

- **P2P Hybride :**

Ces réseaux utilisent des nœuds spéciaux appelés "**super-peer**" réalisant les indexations des fichiers partagés et servant d'intermédiaire pour les nœuds qui leur sont rattachés. Ils sont choisis en fonction de leur puissance de calcul, leur bande passante...

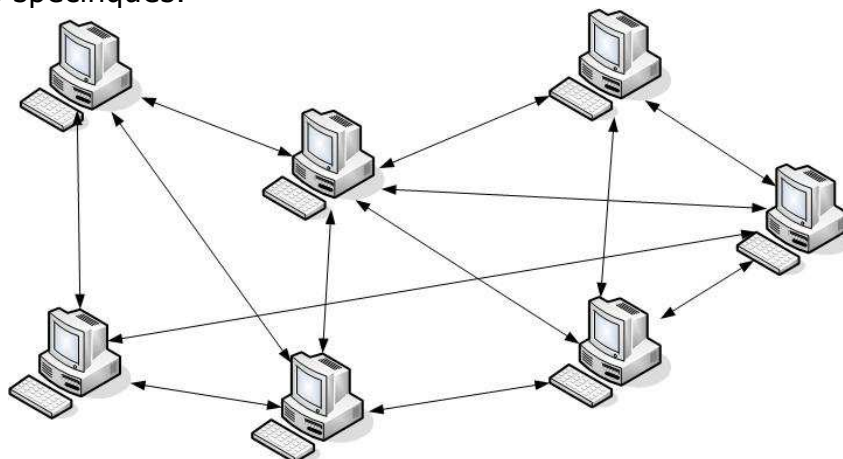
Ce type d'architecture est notamment utilisé dans les réseaux FastTrack tel que KaZaA. Les serveurs utilisés dans eDonkey2000 peuvent eux aussi être vu comme des super-peers (Très nombreux serveurs spécialisés communiquant entre eux et servant d'intermédiaire pour les clients).



Principe de fonctionnement d'un P2P Hybride

- **P2P "purs" :**

Les réseaux totalement décentralisés sont dits "purs". Dans ce type d'architecture, l'ensemble des nœuds sont égaux et jouent le même rôle. Chaque pair gère donc les recherches et le partage, sans passer par un serveur central ou des super-peer, en utilisant en général des techniques de "flood" ou des algorithmes spécifiques.

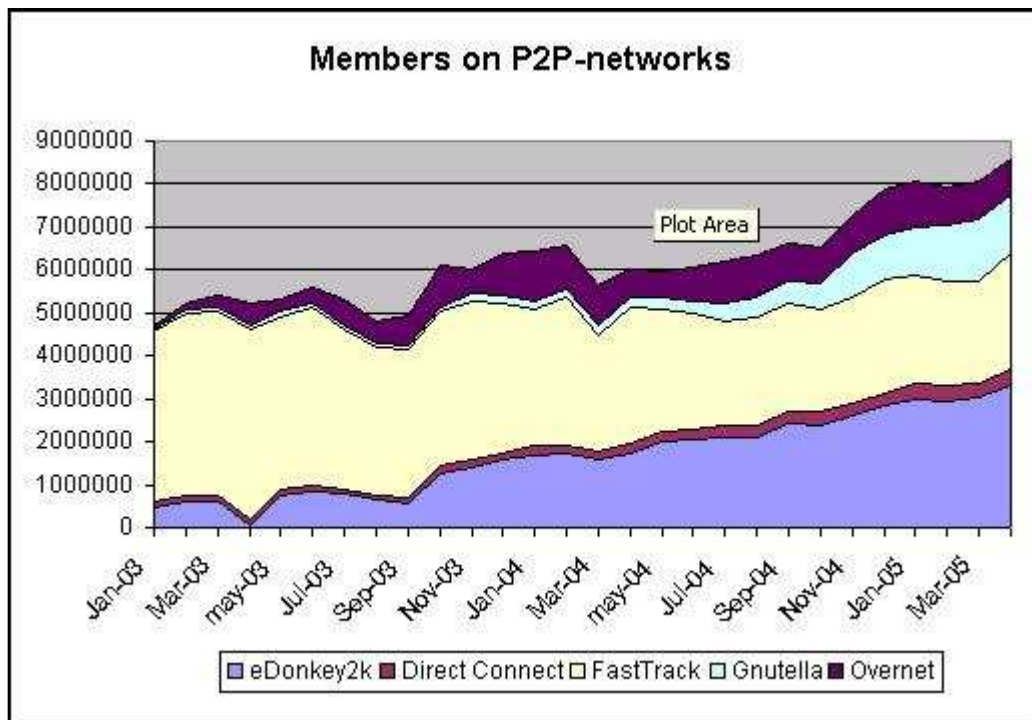


Principe de fonctionnement d'un P2P pur

Cette classification permet de mieux rendre compte du niveau de "dépendance" du réseau. Les réseaux basés sur l'utilisation de serveurs / de super-peer étant beaucoup plus sensible que les réseaux "purs". Dans un réseau totalement centralisé du type Napster, il suffit de bloquer le serveur central pour bloquer le réseau dans son intégralité.

1.5 Statistique d'utilisation des réseaux

Les P2P permettant l'échange de fichiers se sont considérablement développés depuis la disparition de Napster en 2002. Parmi les réseaux les plus populaires aujourd'hui, on retrouve les réseaux FastTrack avec le logiciel KaZaA ou encore le réseau eDonkey (avec le logiciel eMule) qui est actuellement le plus utilisé.



Importance de l'utilisation des principaux Peer-to-Peer dans le monde

Ainsi, avec près de 10 millions d'utilisateurs connectés simultanément sur ces différents réseaux, à n'importe quel moment de la journée, les Peer-to-Peer constituent un phénomène incontournable en informatique aujourd'hui.

On remarquera que le réseau BitTorrent pourtant très populaire puisqu'il s'agit d'un des deux premiers P2P les plus utilisés, n'apparaît pas dans ce classement. Il est en effet difficile d'établir des statistiques d'utilisation précises.

L'utilisation des P2P est donc sûrement plus importante que ce que montrent ces statistiques et pourrait atteindre près de 20 millions d'utilisateurs simultanés, ce qui représente une partie non négligeable du trafic sur internet, soit près de 60% du trafic mondial fin 2004⁵.

⁵ http://www.cachelogic.com/home/pages/studies/2005_07.php

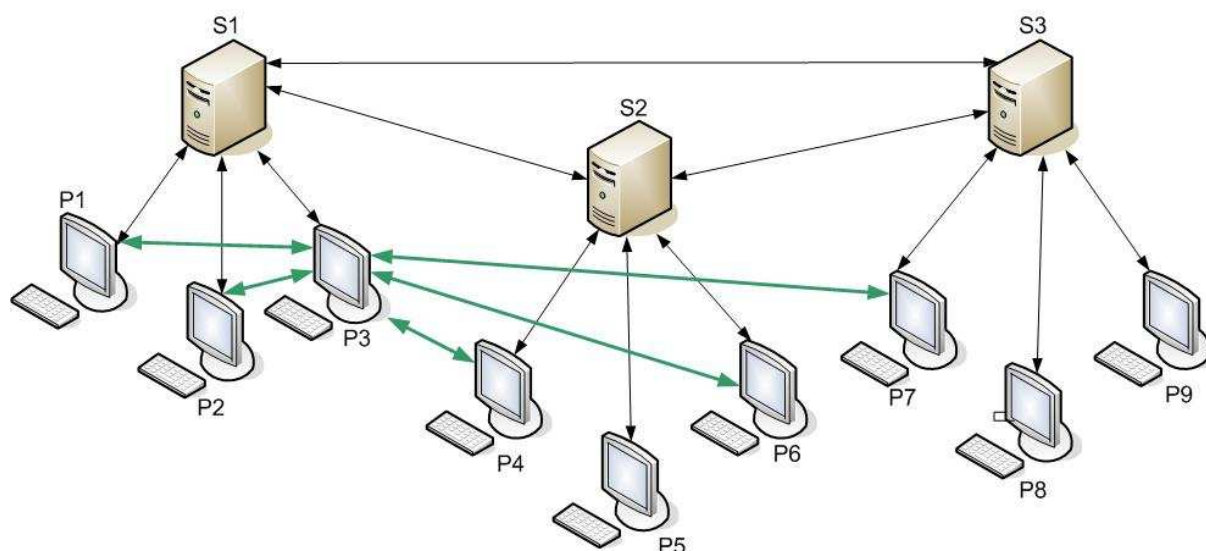
2 Étude du fonctionnement de deux P2P célèbres

2.1 eDonkey

A partir de 2003, le réseau eDonkey devient très populaire auprès du grand public souhaitant partager des fichiers. Il est aujourd'hui le Peer-to-Peer le plus utilisé.

2.1.1 Fonctionnement général

Ce réseau fonctionne dans un type de P2P hybride. Il dispose de nombreux serveurs pouvant être considérés comme des super-peers (S1, S2 et S3 sur la figure ci-dessous). Ils permettent de gérer l'indexation des données, les recherches et la mise en relation des utilisateurs pour commencer les échanges. eDonkey permet uniquement l'échange de fichiers. Son principal atout est de pouvoir partager les parties de fichiers dont on dispose tout en continuant de le télécharger.



Structure générale du réseau eDonkey

2.1.2 Les serveurs

Les serveurs ED2K sont les points d'entrée du réseau. Ils permettent aux pairs de se faire connaître sur le réseau et d'en « rencontrer » d'autres. Ils permettent d'indexer l'ensemble des fichiers des clients connectés et ainsi de leur proposer un moteur de recherches.

La petite histoire des serveurs eDonkey (Jed McCaleb – lugdunum)

Au lancement du réseau, un seul logiciel serveur était disponible, celui créé par Jed McCaleb. Cependant, le code serveur était peu maintenu et souffrait de problèmes d'optimisation, de bugs. Un internaute passionné par ce réseau, nommé "lugdunum" a proposé une version du serveur modifiée par reverse-engineering en ajoutant du code assembleur. Ces patches permettaient alors de découpler les capacités des serveurs passant alors d'environ 10 000 utilisateurs à plus de 100 000 utilisateurs par serveur.

McCaleb décida de lui donner les sources du serveur mais lugdunum, à la vu des sources, préféra réécrire entièrement le code. La première version de cette réécriture a apporté plus de 30% de performance supplémentaire. Aujourd'hui, il n'existe plus que la version lugdunum du serveur, qui est maintenu régulièrement.

Malgré le fait que lugdunum ait réécrit entièrement le code source du serveur, il ne le diffuse pas et ne mets à disposition que des binaires pour les différentes plates-formes.

L'installation des serveurs est très simple, il suffit de télécharger le binaire correspondant et l'exécuter. Un fichier de configuration permet de modifier légèrement le comportement du serveur.

Exemple de paramètres de configuration :

Le port d'écoute du serveur, par défaut 4661

```
port=4661
```

Le nom du serveur: il permet d'identifier plus facilement le serveur dans la liste des serveurs proposés aux clients. Il n'y a aucune restriction sur les noms, ce qui peut entrainer certains abus: on suppose que la RIAA⁶ utilise des noms de serveurs connus pour tromper les utilisateurs. (cf. Razorback)

```
name=eserver 16.44
```

Le nombre maximum de clients permet d'éviter une saturation du serveur et un plantage par manque de ressources (DoS).

```
maxClients=1024
```

La taille maximum des fichiers: si le client envoi une chaîne supérieure, elle est tronquée.

```
max_string_size=140
```

Le nombre maximum de résultats retourné par la fonction de recherche. Elle permet d'éviter de surcharger la connexion et la mémoire du serveur.

```
maxSearchCount=200
```

Le nombre maximum de serveurs que le serveur peut ajouter dans sa liste. Le but est d'éviter les attaques par dénis de service.

```
maxservers=4096
```

Le serveur envoie périodiquement des messages « ping » pour savoir si la connexion TCP est maintenue. Ce paramètre permet de définir l'intervalle entre chaque ping.

```
ping_delay=300
```

Le paramètre thisIP est très important car "eserver" ne détermine pas l'ip publique du serveur.

```
thisIP=
```

⁶ Association d'enregistrement américaine des disques (Société de droit d'auteur)

Les fichiers peuvent être des "faux fichiers", c'est-à-dire que le nom du fichier ne correspond pas au contenu. Pour éviter cela, les clients peuvent donner une évaluation, un commentaire sur ce fichier. Il va être transmis au serveur qui va envoyer ce message aux autres possesseurs du fichier. Ce paramètre va permettre de limiter le nombre maximum de ces messages.

```
warnfakes=0 (A partir de la version 16.50)
```

Les communications entre les serveurs sont « limitées » car il n'y a pas de réel dialogue entre eux mais ils maintiennent juste leur liste de serveurs à jour. Si un nouveau serveur arrive, il se déclare auprès de plusieurs autres qui vont avertir à leur tour l'arrivée d'un nouveau serveur.

2.1.3 Les clients

Le client est sûrement l'élément le plus important dans le réseau eDonkey car c'est lui qui assure réellement la fonction de « Poste à Poste ». Son rôle est de :

- ⇒ Se connecter à un serveur pour indiquer ce qu'il partage
- ⇒ Emettre des requêtes de recherche de fichiers
- ⇒ Demande les sources possibles pour un fichier demandé
- ⇒ Télécharger les fichiers demandés sur les sources
- ⇒ Vérifier l'intégrité et la corruption des fichiers
- ⇒ Envoyer les fichiers demandés par les autres clients
- ⇒ Gérer une file d'attente des clients
- ⇒ Proposer une interface graphique efficace

Le client eDonkey2000

eDonkey2000 (**MetaMachine**) est le premier client utilisant le réseau eDonkey. C'est ce logiciel qui a fait connaître ce réseau, mais ses performances moyennes et sa programmation peu efficace ont engendrées la création d'un client concurrent: Emule. La société MetaMachine a fermé ses portes définitivement le 28 septembre 2006 suite à des poursuites judiciaires engagées la RIAA.

La fermeture de l'entreprise n'empêche cependant pas le réseau de continuer à fonctionner car il n'est pas centralisé et donc ne peut pas être contrôlé.

Le client eMule

Emule est le plus réputé des clients du réseau eDonkey aujourd'hui. Ce logiciel a été développé initialement par un certain Merkur et ses codes sources sont publics (disponible sur sourceforge.net). Le fait qu'il soit open source a fait apparaître un certain nombre de versions enrichies ou « mods », qui sont censées apporter de meilleures performances et d'autres options. Il fonctionne uniquement sur les systèmes d'exploitation Microsoft Windows d'où la création d'autres clients pour fonctionner sur linux par exemple.

Emule propose en plus une extension du protocole d'eDonkey conçue pour la sécurité et l'utilisation du protocole UDP.

Les autres clients

On peut citer d'autres clients comme mldonkey, amule (linux), shareaza, ...

Une tendance est de créer des logiciels capables d'utiliser de nombreux réseaux à la fois. Par exemple, MLDonkey (créé initialement par Fabrice Le Fessant de l'INRIA) utilise les réseaux eDonkey, Overnet, BitTorrent, Gnutella, FastTrack, Kad Network, OpenNap, SoulSeek, DirectConnect et HTTP/FTP.

2.1.4 Fonctionnement détaillé d'un client

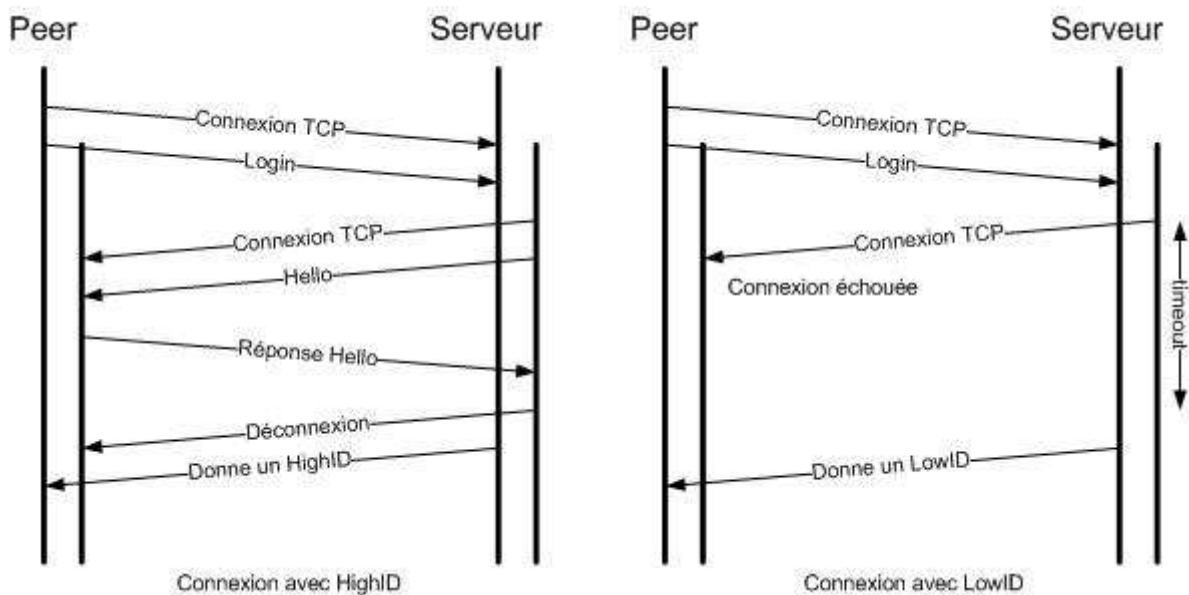
La connexion à un serveur

La connexion à un serveur peut être de deux types. La première est la connexion **highID**: c'est celle qui est recommandée car elle permet une connexion facile sur le pair. Ce highID correspond à l'adresse IP du client sous forme d'entier.

La seconde s'appelle le **lowID**. Cette ID est assigné au client quand le serveur n'a pas réussi à initier une connexion avec lui. Il est en fait injoignable directement, comme par exemple dans le cas où le pair se trouve derrière un NAT (Network Address Translator) sans PAT (Port Address Translator) ou si un pare-feu filtre les connexions entrantes.

Un lowID restreint l'utilisation du réseau par le client, et le serveur peut également rejeter sa connexion. Le lowID est un numéro attribué par le serveur pour identifier le client n'ayant pas d'IP publique (ou directement accessible).

Ainsi, deux lowID ne peuvent jamais communiquer ensemble s'ils ne sont pas sur le même serveur car les serveurs ne s'échangent pas entre eux les lowID.



Le User ID (ou User Hash):

Le client supporte un système de crédit dans le but d'encourager les utilisateurs à partager leurs fichiers. Plus le client envoie des données à des autres clients, plus il gagne des crédits chez ces clients et avance dans leur file d'attente. Le User ID est un nombre de 128 bits (16 octets) créé par la concaténation de nombres aléatoires. Seules les octets 6 et 15 ne sont pas des nombres aléatoires dans le client eMule, la valeur est fixée respectivement à 14 et 111.

```
void CPreferences::CreateUserHash()
{
    for (int i = 0; i < 8; i++)
    {
        uint16 random = GetRandomUInt16();
        memcpy(&userhash[i*2], &random, 2);
    }
}
```

```
// mark as emule client. that will be need in later version
userhash[5] = 14;
userhash[14] = 111;
}
```

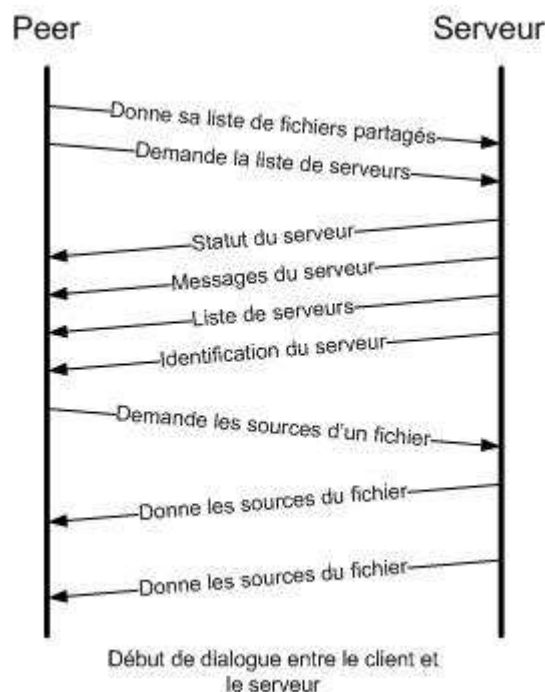
Le user ID joue un rôle important dans le système de crédit et cela motive les « hackers » à trouver un user ID disposant de nombreux crédits. Le client eMule implémente un cryptage qui est conçu pour éviter la fraude. Cette implémentation est un simple échange de type challenge/réponse qui repose sur RSA (cryptage à clé privée/publique).

L'échange de message lors de l'établissement d'une connexion avec le serveur

Après l'établissement de la connexion, le client et le serveur échangent plusieurs messages. Le but de ces messages est de mettre à jour leurs informations.

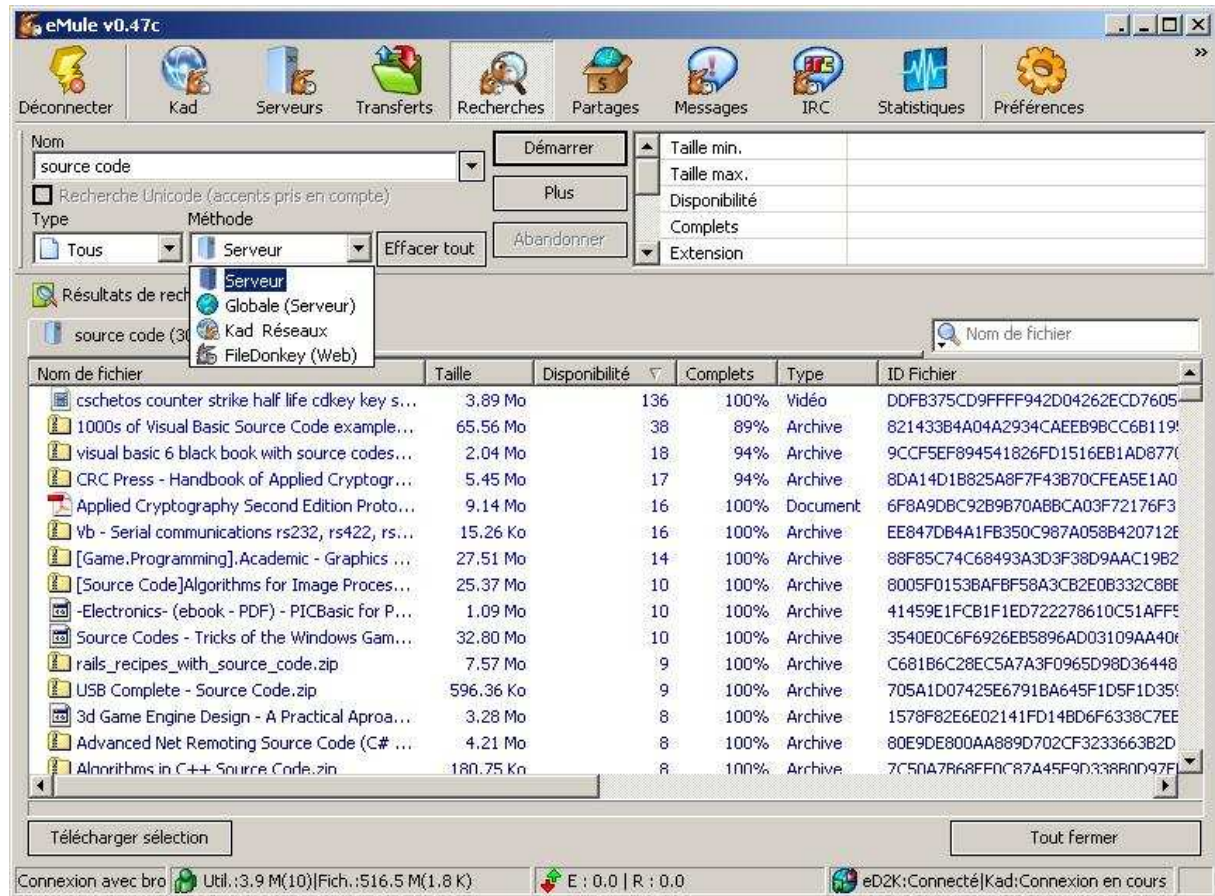
Le client commence par offrir au serveur la liste de ses fichiers partagés et demande ensuite une mise à jour de sa liste de serveurs. Le serveur envoie son statut, sa version de serveur puis sa liste de serveurs connus.

Enfin, le client demande les sources pour les fichiers qu'il est en train de télécharger. Le serveur répond par une suite de messages avec les sources pour chaque fichier.



La recherche de fichiers

Le client peut effectuer une recherche de fichiers en donnant **une liste de mots clés**. Ces mots peuvent être liés par des opérations binaires telles qu'OR, AND ou NOT. La figure ci-dessous montre un exemple de recherche :



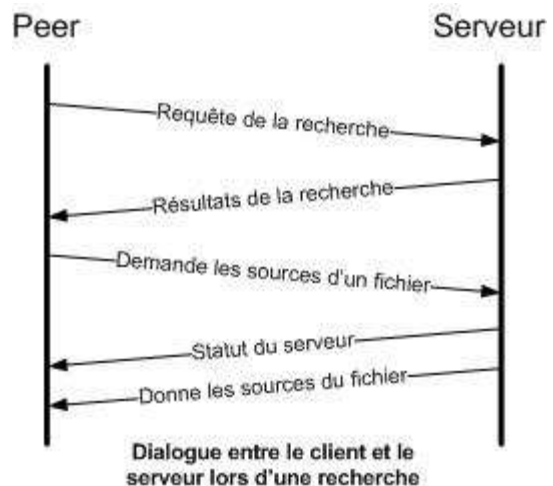
Exemple de recherche sur un serveur à partir d'un client eMule

Le serveur va ensuite comparer les mots clés aux noms des fichiers disponibles et va répondre en envoyant la liste de fichiers correspondants.

La recherche peut avoir aussi plusieurs autres paramètres comme le type de fichier, la taille minimum, la taille maximum et l'extension du fichier.

Les recherches peuvent être effectuées sur le serveur sur lequel le client est connecté (TCP), mais il est également possible d'interroger les autres serveurs de manière non connectée (UDP). Cette recherche permet de trouver plus facilement des fichiers peu répandus (cette recherche est nommé « recherche serveur globale » dans eMule).

L'utilisateur peut alors choisir le fichier qui lui convient dans la liste. Le client eDonkey peut alors demander au serveur les sources ayant ce fichier. Le serveur va lui retourner la liste des pairs correspondant pour pouvoir télécharger comme décrit dans la figure ci-contre.



La connexion et le téléchargement sur les pairs

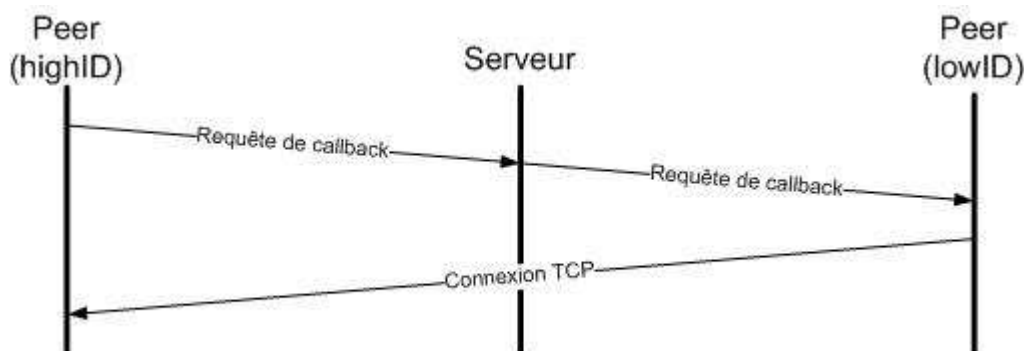
Après avoir récupéré la liste des pairs ayant le fichier ID du serveur, le client va essayer de se connecter à ceux-ci. Pour éviter une accumulation d'attentes de connexion, le client va effectuer une régulation en essayant de se connecter aux pairs les uns après les autres (« sliding window »).

S'il y a de très nombreuses sources, le client peut se limiter à un nombre maximum de connexions à maintenir.

La valeur par défaut du **nombre de connexions maximum** sur eMule est de 500 et le nombre de nouvelle connexion est de 20 par seconde.

Des messages sont émis en UDP périodiquement entre les pairs pour connaître leur position dans la file d'attente. Il y a 3 types de réponses à ce message : soit le client est dans la file, soit la file est pleine, soit le fichier n'existe pas.

Le mécanisme de callback (ou de rappel)



Mécanisme de callback pour les lowID

Dans le cas d'un client lowID ne disposant pas d'IP directement joignable, le pair highID va demander au lowID d'établir une connexion vers lui par l'intermédiaire du serveur. Le serveur va ainsi servir de **relai** (mécanisme de callback) entre le pair lowID et le pair highID. Le client lowID va donc établir une connexion vers le highID et ainsi une communication pourra être réalisée entre les deux.

Une connexion entre un pair lowID et un autre pair lowID est possible en passant tout le trafic par le serveur. Cependant, cette fonctionnalité n'est jamais permise car elle demande des ressources réseaux et systèmes trop importantes.

2.1.5 Les fichiers sur le réseau

Le File ID

Les File ID sont utilisés pour identifier de façon unique les fichiers sur le réseau et ils sont aussi utilisés pour détecter la corruption. Le File ID est calculé par hachage du contenu du fichier, c'est pour cela qu'il est aussi appelé le File Hash. Ainsi, le File ID a deux rôles: le premier est l'identification d'une manière unique un fichier et le second est utile pour la détection de la corruption.

Le File hash

Les fichiers sont identifiés par un hash sur 128 bits calculé par le client et basé sur le contenu du fichier. Ce hash est calculé par l'algorithme MD4. Le fichier est divisé en parties de 9,28Mo, dont on calcul, pour chacune d'elles, la somme MD4. Ensuite, l'ensemble des hashes sont combinés, ce qu'on appelle le "hashset". On applique le MD4 sur le hashset pour obtenir le File hash.

Gestion de la Corruption (ICH - Intelligent Corruption Handling)

Quand un pair a fini de télécharger une partie du fichier, son hash MD4 est généré et il le compare à la partie du hashset fournie par le possesseur du fichier. S'ils sont différents, c'est que la partie téléchargée est corrompue. Un processus de restauration (ICH) se met alors en action.

Pour éviter de re-télécharger la partie entière de 9,28Mo, ICH reprend uniquement des blocs de 180Ko à partir du début de la partie corrompue, et à chaque bloc téléchargé, le hash est recalculé jusqu'à ce que la partie ait le même parhash. Dans le pire des cas, on aura re-téléchargé la partie entière. En moyenne, on gagne 50% du re-téléchargement de la partie du fichier.

Bien sûr, le client attend que la partie soit vérifiée avant de la diffuser à d'autres clients.

Gestion avancée de la corruption (AICH - Advanced Intelligent Corruption Handling)

Ayant le même principe que l'ICH, l'AICH⁷ utilise l'algorithme SHA1 sur des blocs de taille 180 Ko. Il fournit un niveau de **fiabilité et détection de corruption** meilleur que l'ICH. Il permet un re-téléchargement plus fin des parties corrompues. L'AICHset (l'ensemble des hashes de chaque partie) peut être très important en mémoire. Il est ainsi stocké dans un fichier "*known2.met*" qui est lu à la demande.

Ce découpage du fichier en plusieurs parties permet aussi d'envoyer une partie du fichier même si l'on ne possède pas entièrement celui-ci.

2.1.6 Liste d'attente de téléchargement

Lorsqu'un client souhaite télécharger une partie du fichier sur un autre client, il est placé dans une file d'attente de téléchargement. Si la file est vide, alors le pair commence immédiatement le téléchargement. Si au contraire, des clients sont déjà en attente, alors il est placé à la suite.

La file d'attente est de taille finie et il est possible que le client soit rejeté parce qu'elle est pleine. Un système de crédit est mis en place pour aller plus vite dans la file.

Le système de crédits

Le système de crédit permet de télécharger plus rapidement si on partage beaucoup de données, mais aussi d'éviter que des pairs téléchargent sans participer au réseau. Le pair contribuant au réseau **gagne des places dans la file d'attente** et diminue ainsi son temps d'attente pour le téléchargement.

Ce système est basé sur le calcul de rapports⁸ entre **les données émises et reçues**. Les valeurs extrêmes des rapports, qui interdisent le téléchargement ou qui place un pair toujours premier dans la file sont évitées grâce à des maxima et minima.

Pour éviter la fraude, vos propres crédits sont sauvegardés par le pair qui accorde ce crédit.

La possibilité de définir des plafonds d'utilisation de la bande passante apporte un équilibre entre l'envoi et la réception de données.

⁷ http://www.emule-project.net/home/perl/help.cgi?l=1&topic_id=589&rm=show_topic

⁸ www.emule-inside.net

2.1.7 Liste des serveurs eDonkey

Pour accéder au réseau, la première étape est de se connecter à un serveur. Le système étant semi centralisé, il faut que le client possède une liste de serveurs (adresse ip et port) sur lesquels il est possible de se connecter.

Cette liste est enregistré dans un fichier « server.met » pour le client eMule et eDonkey2000. Par défaut, elle contient des serveurs « stables », c'est-à-dire qu'ils sont en permanence disponible pour se connecter. Mais le dynamisme du réseau implique un mécanisme de mise à jour de cette liste.

Lorsqu'un client se connecte, le serveur lui envoi sa liste de serveurs. Le client peut alors mettre à jour sa liste. Il effectue régulièrement des **tests de présence** des serveurs (requête en UDP). Si un serveur ne répond pas dans un délai imparti, il est marqué inactif dans sa liste et un compteur de tentative est incrémenté. Si le compteur dépasse un seuil (seuil configurable), le serveur est supprimé de la liste.

Lorsqu'un nouveau serveur arrive sur le réseau, il **annonce sa présence** aux autres serveurs et sera ainsi ajouté dans leur liste. Les serveurs testent régulièrement l'activité des autres serveurs. Si l'un d'eux ne répond plus, il sera supprimé de la liste.

2.1.8 Problèmes d'eDonkey et quelques solutions

Les problèmes réseaux

Le problème du NAT (Network Address Translator)

Beaucoup de personnes utilisent un routeur pour se connecter à internet. Ils ne disposent alors que d'une seule adresse publique, celle du routeur. Il faut donc rediriger les ports TCP et UDP de eMule vers le bon client (PortForwarding ou PAT). Dans le cas où cette redirection n'est pas effective, le client pourra uniquement se connecter en lowID et ainsi ne va pas être privilégié pour le partage.

Les extensions non respectueuses du protocole

Le protocole eDonkey n'est pas pensé pour la sécurité ainsi que son respect. En effet, lorsque seul le client eDonkey2000 existait, un logiciel nommé **eDonkeyBot**⁹ a mis en péril le réseau.

Ce logiciel, créé par un hacker utilisant le pseudonyme "pb33", utilise des méthodes ne respectant pas le protocole. Il permettait de limiter l'envoi de données (upload) ou même de le couper complètement, de rechercher des sources de façon agressive (multiple demande au même serveur et demande en UDP à tous les serveurs connus en même temps), de mettre à jour automatiquement la liste des serveurs eDonkey, de changer de serveur périodiquement toutes les x heures (dans le but d'essayer de trouver encore plus de sources).

⁹ http://tdn.no-ip.org/pl/tdn.pl/tools_bot.html

L'ensemble de ces fonctionnalités a fait chuter les performances du réseau à cause d'un surnombre d'informations de contrôle. Heureusement, l'arrivée d'eMule avec des fonctionnalités plus intelligentes a permis de limiter le déploiement de ce logiciel.

Redirection de trafic

Le rôle du serveur permettant d'indiquer qui est le possesseur d'un fichier est un risque pour le réseau. En effet, un serveur avec un binaire spécialement forgé peut indiquer toujours une même source (client ou serveur), quelque soit le fichier demandé. Ainsi, on peut créer un **DDoS (Distributed Deny of Service)** en particulier si le serveur possède de nombreux clients. L'ajout d'un serveur ne pose pas de problème (grâce au paramètre seedIP ou grâce à un fichier serverList.met) hormis le fait de la nécessité de nombreuses ressources (surtout en bande passante).

Faibles induites par intégration de bibliothèque

Aussi bien le serveur lugdunum que le client eMule intègre des bibliothèques développées par des tiers. Des failles peuvent être découvertes dans ces bibliothèques pouvant ainsi atteindre le logiciel.

Le développeur de lugdunum n'intègre pas forcément toutes les nouvelles bibliothèques. Par exemple, le serveur lugdunum et le client eMule intègrent "*Zlib*", or des failles ont été découvertes dans cette librairie. Ainsi en utilisant le protocole compressé, on peut déclencher à distance la faille et corrompre la mémoire du serveur/client. eMule a aussi des failles induites par des libraires (inclus statiquement) comme par exemple *ID3Lib* ou encore *CxImage*. La plupart de ces failles restent au niveau local et ne permettent pas de prendre le contrôle du client distant.

L'ouverture et le non anonymat du pair sur le réseau

Lorsque l'on se connecte à un serveur et partage des fichiers, notre adresse IP est fournie à tous les clients désirant télécharger un de nos fichiers. Cette diffusion de l'adresse, unique au monde, apporte un manque d'anonymat total au pair. Ainsi, on peut savoir qui dispose d'un fichier avec un copyright.

Cette mise à disposition permet aussi aux pirates de connaître des ordinateurs qu'il serait possible d'exploiter. Par exemple, ils peuvent tenter une recherche de ports ouverts pour essayer d'infiltrer l'ordinateur.

Le filtrage d'eDonkey et l'offuscation du protocole

Le filtrage des paquets eDonkey est facile à mettre en place car les ports utilisés sont souvent ceux par défaut (4661-4665) et le premier octet de l'en-tête est 0xE3. Avec l'augmentation du filtrage des FAI, les dernières versions eMule offusque le protocole pour le contourner et donne des ports aléatoire à l'installation.

L'établissement de la connexion entre clients ou entre client et serveur débute par l'échange de clés et ainsi permettre le chiffrement des données. Ces données ne pourront donc pas être décryptées par les routeurs pour effectuer le filtrage. Pour plus détail, consulter le code source d'eMule (EncryptedStreamSocket.cpp & EncryptedDatagramSocket.cpp)

Les problèmes fichiers

On trouve sur le réseau de nombreux fichiers dont le nom ne correspond pas à leur contenu (**fake file**). Pour limiter ce phénomène, plusieurs solutions existent. La première consiste à donner un commentaire et une évaluation aux fichiers que l'on possède. Le problème est que lorsque l'on a téléchargé un fichier et que ce n'est pas le bon, on ne le garde pas et le commentaire n'existera plus. En effet, les commentaires sont stockés dans un fichier qui est transmis au serveur lors de la connexion et qui sera effacé si le fichier n'est plus partagé par l'utilisateur.

La deuxième solution est la prévisualisation. eMule intègre cette fonctionnalité (dans le cas d'un fichier vidéo, d'une musique ou d'un fichier d'archive .rar) en téléchargeant d'abord le début du fichier (les en-têtes sont nécessaires pour identifier le codec et les paramètres) et ensuite d'autres parties qui seront assemblées pour fournir un « extrait ». On pourra ensuite décider de continuer le téléchargement ou de l'arrêter.

Un autre problème est **le manque de complétude**. Il n'est pas rare qu'une personne mette à disposition un fichier durant quelques heures et qu'il ne se connecte plus jamais au réseau. Ainsi, aucun autre pair n'a ce fichier et ceux qui ont commencé le téléchargement ne pourront jamais le finir. Pour éviter ce phénomène, un taux de complétude est calculé et permet de savoir si le fichier est plutôt complet ou incomplet chez tous les pairs lors de la recherche.

Enfin, un dernier problème concerne l'envoi de fausses données. Cet envoi est possible grâce à la **collision de hash MD4**. MD4 crée un condensat qui identifie des données. Mais il est possible que deux données différentes donnent la même empreinte MD4, c'est ce que l'on appelle une collision. La création d'une collision est d'autant plus facilitée qu'il existe des problèmes dans MD4. Ce problème est corrigé grâce à AICH qui repose sur SHA1 (nombre de collisions plus faible car condensat plus long et pas de problèmes identifiés).

Les problèmes logiciels

Besoin en ressources matérielles

La limitation du nombre de fichiers que l'on partage est de 120 (configuration serveur) et Lugdunum (le créateur des serveurs) explique cette limitation par ce raisonnement :

"Un client qui partage par exemple 10000 fichiers bloque tous les autres clients du serveur pendant 20 à 120 secondes lors de son départ. En gros, le programme avait inscrit en mémoire des couples. A la déconnexion d'un client possédant beaucoup de fichiers, le temps nécessaire pour parcourir toutes ces structures en mémoire pour les retirer devient astronomique (parcours linéaire de listes simplement chaînées)".¹⁰

Les serveurs peuvent demander beaucoup de ressources (quad opteron 32go de RAM et liaison 100Mbits) car ils peuvent gérer un nombre de connexions allant de plusieurs centaines de milliers, à plus d'un million d'utilisateurs.

¹⁰ www.ratiatum.com

Un problème était apparu au début de la pleine expansion d'eDonkey, c'est le manque de serveurs ayant les capacités nécessaires. L'ensemble des serveurs était saturé mais cette saturation a été levée en particulier grâce à l'arrivée du serveur lugdunum, plus performant et moins gourmand en ressources.

Un autre problème de connexion au serveur existait aussi car de nombreux serveurs avaient des IP non statique. La liste de serveur posait donc de nombreux problèmes de mise à jour et la connexion à un serveur était difficile. La solution a été de mettre en place la liste de serveurs eDonkey dont la plus célèbre est "*ocbMaurice*".

La fermeture des codes sources des serveurs (lugdunum) peut également poser problèmes car peu de personnes peuvent travailler dessus pour limiter l'apparition de nouveaux bugs. eMule est supposé plus fiable car son code source a été analysé par de nombreuses personnes.

Problème des serveurs

Comme tout Peer-to-Peer centralisé, la faiblesse du réseau eDonkey est principalement due à l'existence de serveurs gérant les connexions. Si ces serveurs sont supprimés, cela peut entraîner des ralentissements sur le réseau, en particulier s'il supportait de nombreuses connexions. Ça a été le cas par exemple pour le serveur "*Razorback*".

Razorback¹¹ est une association suisse qui possédait un serveur eDonkey/eMule de P2P extrêmement populaire, qui brassait à son apogée, plus de 33% des utilisateurs du réseau ed2k. Il a évolué le 16 février 2004 vers sa version 2.0 et a été suivi en 2005 par son petit frère nommé Razorback2.1.

Les serveurs, situés à Zaventem près de Bruxelles, ont été saisis le mardi 21 février 2006 par la brigade anti-piratage de la police fédérale belge et le porte-parole de l'association a été entendu par la police suisse (l'association était basée en Suisse). La MPAA¹² s'est félicitée de la fermeture à son initiative de ce « serveur infâme ».

Malgré l'arrêt des deux serveurs, le réseau n'a pas montré de réels signes de faiblesse, et ce en grande partie grâce à la prise en charge des utilisateurs par d'autres serveurs et à l'utilisation du protocole décentralisé reposant sur l'algorithme Kademlia (qui fonctionne sans serveur dédié).

Les serveurs nommés Razorback actuellement en ligne n'appartiennent pas à l'association Razorback. Celle-ci est toujours en activité mais n'a pas encore ré ouvert de serveur eDonkey.

Le fait qu'eDonkey utilise de nombreux serveurs dans différents pays, lui permet donc de surmonter les problèmes de pertes de serveurs, ce qui lui a permis de continuer à exister.

¹¹ http://fr.wikipedia.org/wiki/Razorback_%28serveur_ed2k%29

¹² MPAA : Motion Picture Association of America

2.1.9 Kad Network

Ce système à la différence d'eDonkey est complètement décentralisé, même en ce qui concerne la recherche. Il n'y a qu'un seul programme qui sert au téléchargement et à l'indexation des fichiers (système réparti homogène).

Le réseau Kad utilise **Kademlia** qui est un système à base de DHT (Distributed Hash Table).

Le réseau Overnet de MetaMachine utilise aussi Kademlia mais n'est pas compatible avec le réseau Kad de eMule.

Le problème de Kad est son initialisation. En effet, par défaut il n'y a pas d'adresse statique (pas de serveur) et pas de liste de pairs Kad. Donc s'il n'y a pas d'adresses, on ne peut pas se connecter au réseau...

Son intégration à eMule permet d'éviter ce problème. Il utilise ainsi le réseau eDonkey pour trouver des pairs utilisant aussi Kad. Cela permet de mémoriser les adresses IP des clients utilisant Kad, pour une future connexion (en espérant qu'il y ait au moins un client joignable).

Une force du client eMule est la possibilité de télécharger un même fichier à la fois sur les réseaux Kad et sur eDonkey.

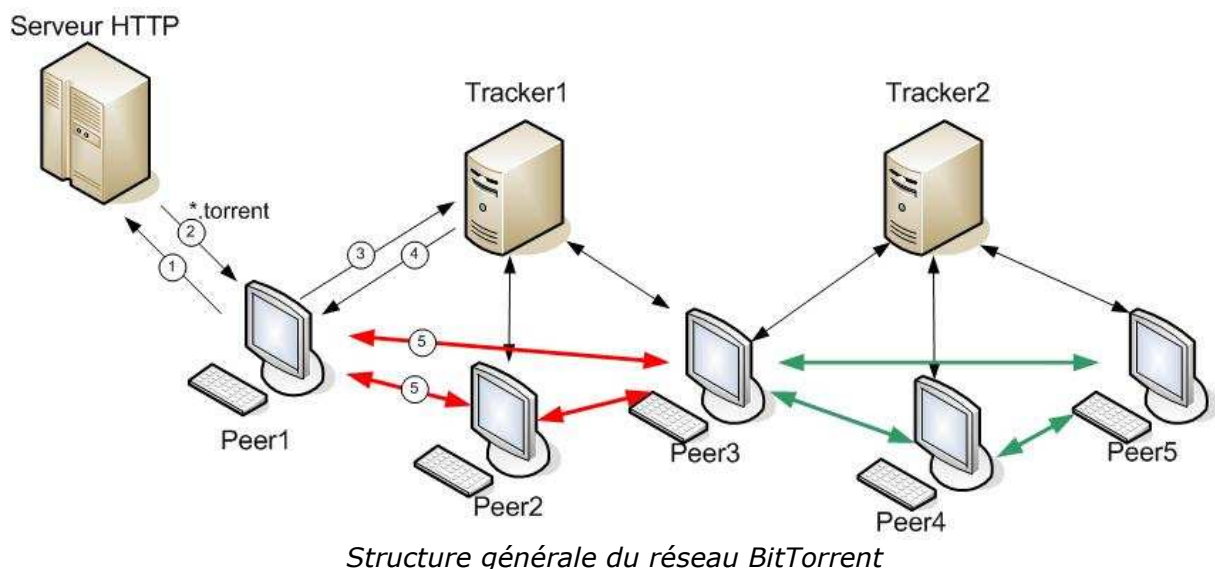
2.2 BitTorrent

Le réseau BitTorrent possède aujourd'hui une communauté d'utilisateurs très étendu, même si son architecture ne permet pas de déterminer son nombre exact. Les échanges de fichier à travers ce protocole représentaient déjà près de 30% du trafic sur internet en 2004¹³.

2.2.1 Principe de fonctionnement

Le terme BitTorrent désigne à la fois le protocole utilisé pour le partage et les logiciels clients qui l'implémentent. Le réseau BitTorrent, inventé par le programmeur Bram Cohen, a un mode de fonctionnement assez proche du réseau eDonkey.

Pour partager un fichier, les clients commencent par créer un fichier "**torrent**" (suffix *.torrent*) qui contient des « métadonnées » sur le fichier, notamment le HASH File. Il contient aussi l'adresse URL du tracker qu'il faut interroger pour pouvoir télécharger le fichier.



Pour télécharger un fichier, le client BitTorrent (programme client) recherche le *.torrent* correspondant (sur un serveur web par exemple – *flèche 1 sur le schéma*) et le télécharge (*flèche 2*). Il utilise ensuite l'URL contenu dans le "*.torrent*" pour contacter le tracker responsable du fichier pour lui demander la liste des pairs partageant actuellement le fichier (*flèche 3*). Le tracker renvoie alors une liste d'adresse IP à contacter (*flèche 4*). Dès lors, le client BitTorrent contacte directement ces adresses et commence le téléchargement (*requêtes 5*). La liste d'IP est constituée simplement à partir des adresses des personnes qui téléchargent le fichier désiré en même temps que nous : notre client télécharge le fichier sur les autres Pairs, et ceux-ci téléchargent chez nous.

¹³ http://www.cachelogic.com/home/pages/studies/2005_06.php
/ MISC N°25 - Mai Juin 2006 - p24 à p47

2.2.2 Terminologie

Torrent

Un torrent est un fichier, codé en ASCII, portant l'extension ".torrent" et permettant d'identifier les partages des utilisateurs sur le réseau. Ces partages peuvent être aussi bien des fichiers (film, musique, image cd) que des répertoires. Le fichier torrent est essentiel pour établir les connexions. Il contient en effet les informations sur le tracker référençant le fichier, des informations sur le fichier lui-même : son nom, sa taille, le hash SHA1 du fichier permettant de l'identifier, mais aussi la taille des fragments du fichier et leur hash afin de vérifier que l'intégrité des informations est bien conservée.

Tracker

Le tracker est une sorte de serveur permettant de mettre en relation les pairs. Il peut être considéré comme un annuaire mettant en relation un fichier avec les clients qui le téléchargent.

N'importe qui est susceptible de créer son propre tracker. Pour cela, il faut avant tout posséder une adresse IP fixe ou une URL qui permettra de rediriger vers le tracker. Cette **adresse fixe** est en effet nécessaire pour les connexions, l'adresse du tracker étant en effet enregistré dans le fichier torrent.

Il suffit ensuite de télécharger les sources BitTorrent ainsi que le logiciel d'installation de l'interpréteur python (utilisé par BitTorrent), et d'ouvrir les bons ports dans le firewall du serveur. Le tracker n'a plus qu'à être lancé avec une commande du type « c:\source\bttrack.py --port 6969 --dfile dstate ».

Il ne reste alors plus qu'à tester le tracker à l'aide d'un navigateur Internet¹⁴ (le protocole utilisé étant http, cela facilite le test).

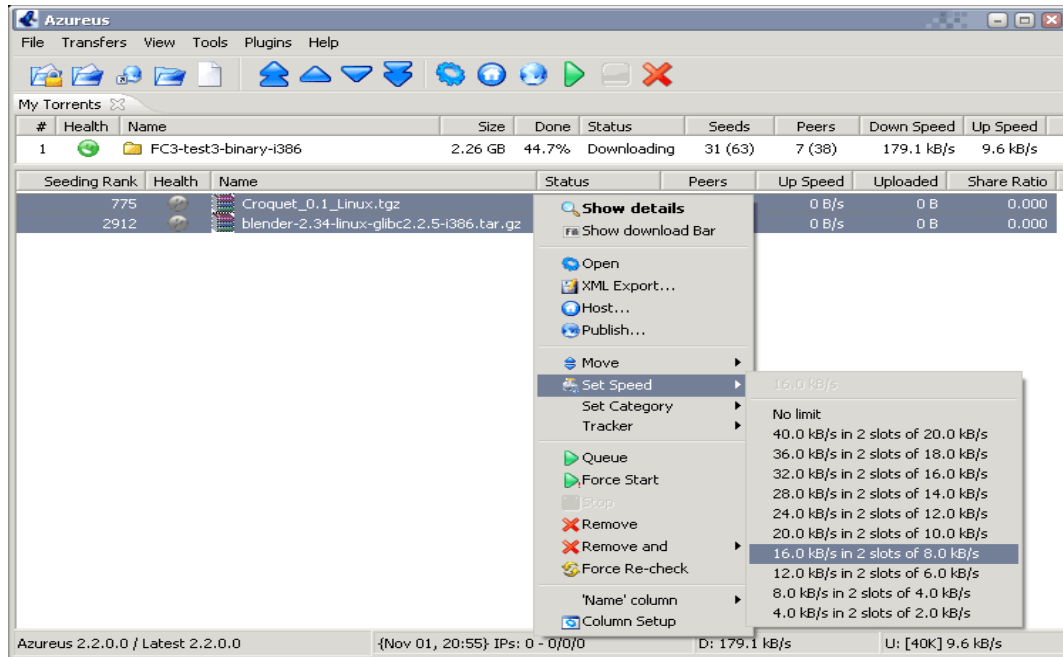
Les personnes connaissant l'adresse du tracker peuvent alors s'y connecter facilement grâce aux informations contenues dans le fichier torrent.

Client BitTorrent

Un client BitTorrent est un logiciel permettant, à partir d'un fichier ".torrent", de se connecter au tracker référençant un fichier afin d'effectuer le téléchargement souhaité. Celui-ci gère donc l'ensemble des échanges : il initialise les communications, télécharge les parties du fichier détenue par d'autres pairs, et envoie aux autres les parties qu'il possède.

Il existe de nombreux clients BitTorrent fonctionnant sous différents langages et diverses plates-formes (windows, linux, mac, etc.).

¹⁴ <http://www.emutorrent.com/faire-un-tracker.php>



Client BitTorrent Azureus, écrit en Java

Seeder (seed signifie graine en anglais)

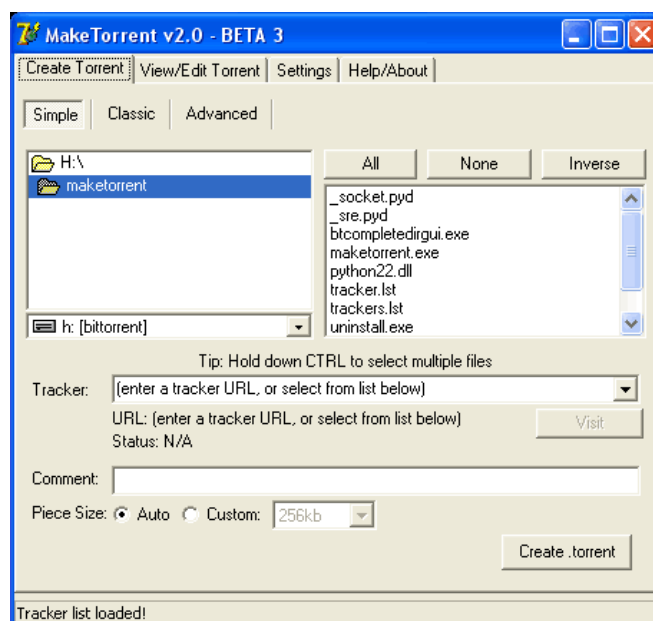
Dans le vocabulaire BitTorrent, une personne qui possède l'intégralité d'un fichier et qui continue de le partager est appelée seeder.

Leecher (leech signifie sangsue en anglais)

Un leecher est une personne qui ne possède pas un fichier dans sa totalité et qui continue donc à essayer de le télécharger.

2.2.3 Création d'un torrent

La création d'un torrent se fait assez simplement à l'aide d'un outil tel que *MakeTorrent*. Il suffit de sélectionner le fichier ou le répertoire que l'on souhaite partager et choisir un tracker.



Création d'un fichier .torrent à l'aide du logiciel MakeTorrent

Il faut ensuite choisir la taille des fragments. En général de 256 ko, cette taille peut augmenter si le fichier est très important. Plus les fragments sont petits, plus le téléchargement sera efficace. L'outil se charge alors de calculer la somme SHA1 des fragments et de générer le *.torrent*. Il ne reste alors qu'à l'enregistrer sur le web pour que d'autres personnes puissent connaître son existence.

Enfin, pour amorcer le partage, il suffit d'utiliser un client BitTorrent comme si on souhaitait télécharger le fichier. Le client vérifiera que le fichier que l'on souhaite télécharger est le même que celui présent sur notre disque. Il se connectera alors au tracker et commencera le partage du fichier. On sera alors considéré comme le seed originel et les autres clients pourront se connecter à nous pour télécharger le fichier.

Voici un exemple de contenu d'un fichier *.torrent* pour un fichier vidéo portant le nom "Black_Lagoon_-_20.avi" :

```
d8:announce34:http://seedy.mine.nu:6969/announce18:azureus_propertiesdl7:
dht_backup_enable1ee7:comment0:13:comment.utf-80:10:
created by15:Azureus/2.5.0.013:creation datei1164774262e8:
encoding5:UTF-84:infod4:ed2k16:0_ê~o_î_C`_ "x8ôÐ6:lengthi179998720e4:name47:
Black_Lagoon_-_20.avi10:name.utf-847:Black_Lagoon_-_20.avi12:
piece lengthi393216e6:pieces9160:%_@;ö-y%¿>ª-rò\ etc.
```

Les champs sont séparés par le délimiteur ":", et sont précédés du nombre de caractères à lire. Les principaux champs sont:

announce: "**http://seedy.mine.nu:6969/announce**" correspond à l'adresse du tracker qui doit être contacté pour pouvoir télécharger le fichier.

created by: "**Azureus**" correspond au client utilisé pour créer le torrent.

creation date: "**1164774262**" correspond au timestamp de la date de création

length: "**179998720**" : la taille du fichier en byte.

name: "**Black_Lagoon_-_20.avi**": le nom du fichier.

piece length: "**393216**": la taille de chaque partie du fichier, issue de la fragmentation.


pieces: "... " correspond à la concaténation du code SHA-1 sur 160 bits de chaque fragment.

2.2.4 Indexation des torrents et recherche d'informations

Un des points faibles du protocole BitTorrent est qu'il ne possède aucune fonction d'indexation. Il n'est donc pas possible de rechercher des fichiers directement avec les clients BitTorrent. La recherche de fichier s'effectue par une application extérieure. Il existe de nombreux sites internet proposant des liens vers des torrents, et certains navigateurs comme Firefox intègrent des extensions permettant d'effectuer rapidement une recherche globale sur ces sites.

La publication sur le web est en effet la seule façon de faire connaître rapidement la présence des fichiers.

Un utilisateur souhaitant télécharger un fichier commence donc par le rechercher sur le web, à travers des sites spécialisés ou non comme par exemple sur le site mininova¹⁵, un des plus connu actuellement (voir image ci-dessous) et télécharge le *.torrent*, puis donne celui-ci à son client BitTorrent qui se chargera du téléchargement.



The screenshot shows the mininova website interface. At the top, there is a search bar and navigation links like 'Home', 'Browse categories', 'Upload a torrent', 'Advanced search', 'Community', 'Search cloud', 'Statistics', and 'FAQ'. Below the navigation, there is a section for 'Newly popular torrents' with links for 'today' and 'yesterday'. A dropdown menu for 'Jump to category' is set to '- Select one -'. The main content area is titled 'Anime' and features a table of torrent listings with columns for 'Added', 'Name', 'Size', 'Seeds', and 'Leechers'.

Added	Name	Size	Seeds	Leechers
01:10	[Kyuu] Busou Renkin - 11[AA923A49] avi	169.87 MB	1632	1029
01:18	[Nipponsei] DEATH NOTE Original Soundtrack (320 kbps) zip	142.83 MB	367	232
21:59	[umai-Doremi]_Kenichi_04_[04302579].avi	169.81 MB	1199	1623
22:03	[umai-Doremi]_Kenichi_05_[FDEBCB0F].avi	166.77 MB	1190	1650
01:10	[gg] Code Geass 11 [4DB89E961] mkv	171.16 MB	1868	989
01:08	[Saizen] Prince of Tennis - National Championship Chapter - 07 [DVD][3840D6D0] mp4	174.68 MB	72	644
22:09	Saint Seya Meikai Hen Hades Inferno OAV 20-XVID.AC3 By Wil2761	345.02 MB	3	1
22:12	Saint Seya Meikai Hen Hades Inferno OAV 21-XVID.AC3 By Wil2761	345.1 MB	0	0
01:11	[Yoroshiku] Kekkaishi - 05 [003402E7] avi	223.12 MB	419	169
00:09	[asv] Code Geass 11 [A5946B76] VOSTFR mkv	186.31 MB	46	100

Le protocole BitTorrent ne propose pas de moteur de recherche. Des sites spécialisés comme mininova.org sont donc apparus pour remédier à ce problème

Ce procédé a tout de même des avantages puisqu'il peut permettre de diffuser un gros fichier à un groupe précis d'utilisateurs, sans que des personnes extérieures en aient connaissance. En effet, il suffit de créer un *.torrent* à partir du fichier que l'on souhaite diffuser et de l'envoyer à nos amis (par mail par exemple). Ainsi, seules les personnes concernées auront connaissances du fichier et pourront venir le télécharger. De plus, l'envoi du fichier sera globalement plus rapide que si on devait l'envoyer à chaque personne une par une.

¹⁵ <http://www.mininova.org/>

2.2.5 Téléchargement des fichiers : connexion au tracker

Pour pouvoir télécharger un fichier, le client lit dans le *.torrent* l'adresse du tracker en charge de ce fichier. Il se connecte ensuite à celui-ci par l'intermédiaire du **protocole HTTP (ou HTTPS)** en envoyant une requête « **GET** ».

Cette requête permet de fournir au tracker différentes données sur le client et sur le fichier recherché. On trouve notamment :

- ⇒ **info_hash** : le hash SHA1 du fichier, qui permet de l'identifier de façon unique
- ⇒ **peer_id** : ID unique sur 20 bits du client, généré au démarrage.
- ⇒ **Port** : le numéro de port utilisé par le client pour l'échange d'information. En général, il s'agit des ports de 6881 à 6889.
- ⇒ **Adresse IP** : ce champ est optionnel car il peut être facilement retrouvé à partir de la requête HTTP envoyée par le client. L'adresse IP peut cependant être utile si l'ordinateur se trouve derrière un serveur proxy ou si le client et le tracker se trouvent derrière une même passerelle NAT.
- ⇒ Le **nombre d'adresses de pairs** souhaité. Par défaut, le tracker renverra une liste de 50 pairs maximum, choisis aléatoirement.
- ⇒ Les statistiques du client : nombre de bits envoyés et reçus

En réponse, le tracker renvoie **la liste des leechers et des seeders** partageant actuellement le fichier demandé (ou un message d'erreur le cas échéant).

Pour chaque pair, le client BitTorrent recevra alors l'**ID**, son **adresse IP** et le **numéro de port** sur lequel il écoute. Il se chargera alors de contacter les différentes sources possibles.

Le client reçoit également **l'intervalle de temps** qu'il doit attendre pour contacter le tracker, ainsi que le **tracker ID** qu'il devra renvoyer dans ses prochaines requêtes.

Le client BitTorrent informe en effet le tracker, à des intervalles de temps réguliers, de son état dans le téléchargement (ce qu'il a téléchargé, reçu, etc.) et reçoit du tracker une liste actualisée des pairs. Ces données ont un but purement statistique.

2.2.6 Téléchargement des fichiers : connexion aux pairs

Une fois la liste des pairs obtenue, le client va chercher à se connecter à ceux-ci. BitTorrent utilise la même technique qu'eDonkey pour gérer l'envoi des fichiers : ceux-ci sont découpés en fragments qui peuvent être envoyés dans n'importe quel ordre.

Cette technique permet de faire du **multisourcing** : il est possible de télécharger plusieurs morceaux simultanément à partir de plusieurs pairs différents. De même, après avoir obtenu un morceau, le client servira de source pour l'envoyer vers d'autres utilisateurs.

Il est bien entendu possible de stopper son téléchargement et de le reprendre là où on l'avait laissé.

Le protocole est basé sur une connexion TCP. Chaque pair garde une table regroupant des informations sur tous les pairs auxquels il est connecté, qu'ils envoient des informations ou non. A chaque fois que le client a fini de télécharger un morceau, il l'annonce à tous ses contacts, ce qui permet de garder en mémoire l'état du téléchargement de tous les pairs et de leur demander les parties qu'ils possèdent et qui nous intéressent. La spécification originale prévoyait qu'un client cherche à obtenir les différents segments d'un fichier de façon aléatoire. Cependant, la plupart des clients actuels cherchent **les parties les plus rares en priorité**.

Contrairement à des logiciels comme eMule, le client ne gère **pas de file d'attente**. Le système de téléchargement est basé sur un système de récompense ressemblant à l'algorithme du "**tit-for-tat**¹⁶". Le principe est simple : j'envoie des données aux personnes qui m'en envoient également.

Ainsi, on accordera la priorité aux requêtes des pairs nous apportant le plus, c'est à dire ceux qui participent le plus à notre téléchargement.

Cette technique du "donnant-donnant" permet de lutter efficacement contre les leechers qui ne font que télécharger sans jamais envoyer de données aux autres.

Le protocole prend tout de même en compte une partie des demandes de pairs qui ne nous envoient pas encore de données. En effet, un problème se poserait si on appliquait l'algorithme du tit-for-tat à la lettre : qui commencerait à partager ses données aux autres ? Les pairs resteraient bloqués tant que l'un d'eux ne se décide à prendre l'initiative et à envoyer des données. De même, lorsqu'un leecher commence le téléchargement d'un fichier, il n'a aucune partie à proposer aux autres. Il risquerait donc d'être bloqué éternellement.

Afin de palier à cet inconvénient, le client BitTorrent réservera tout de même une petite partie de la bande passante pour initialiser des connexions vers des pairs ne le profitant pas, c'est à dire qui ne lui envoient aucune donnée. Cet algorithme, appelé "**optimistic unchoking**" à donc pour but de donner une chance aux nouveaux arrivant et de découvrir de nouveaux partenaires avec qui partager les données, en prenant l'initiative de l'envoi.

Lorsqu'un client refuse d'envoyer des données à un autre pair, on dit qu'il le "**choke**". Cela arrive si le pair n'est pas profitable (s'il ne nous envoie aucune donnée en retour) ou s'il nous est moins profitable que d'autres (la bande passante étant limitée, il faut faire des choix). Plus aucune requête du pair "choké" ne sera acceptée.

L'opération inverse, consistant à accepter les requêtes du pair, est appelée "**unchoke**". Ainsi, le client évalue les autres pairs toutes les 10 secondes, pour déterminer à qui il va envoyer des données et effectue des opérations de "choke" et "unchoke" en fonction du résultat de l'évaluation (nombre de données qu'on lui a envoyé, etc.). Il n'y a donc pas de file d'attente, le partage étant géré uniquement en fonction du crédit des pairs.

¹⁶ http://en.wikipedia.org/wiki/Tit_for_tat

Le Super-seeding :

Cet algorithme permet à un seed (client possédant l'intégralité d'un fichier), d'envoyer le plus vite possible un fichier. Le but est de permettre à des pairs limités en bande passante de partager des gros fichiers.

Contrairement à un seed normal, les parties du fichier ne seront pas envoyées de façon aléatoire. Le client va faire croire aux différents pairs qu'il ne possède qu'un seul fragment du fichier. Quand l'un d'eux se connectera à lui, il enverra cette partie. Une fois le téléchargement de cette partie fini, le super-seed n'enverra plus de morceaux du fichier avant d'avoir vu la partie précédemment envoyée sur au moins un autre pair, ce qui lui permet de s'assurer que la partie est bien diffusée.

Cette technique est 200%¹⁷ plus efficace que l'algorithme utilisé par un seed normal. Il n'est cependant pas recommandé de l'utiliser s'il existe déjà plusieurs seed pour le fichier.

Il est à noter qu'en général, sur un même ordinateur, il y a un client ouvert, sur une fenêtre distincte, par fichier téléchargé. Si cette fenêtre est fermée (en fin de téléchargement par exemple), le fichier n'est plus partagé, ce qui explique la courte durée de vie des fichiers sur le réseau.

2.2.7 BitTorrent et les autres P2P

Le réseau BitTorrent est un réseau **Peer-to-Peer hybride** du fait de l'utilisation des trackers. En cela, **il ressemble beaucoup au principe utilisé par le protocole eDonkey**: les pairs se connectent à des serveurs (des trackers dans le cas de BitTorrent) qui permettent de les mettre en relation.

Cependant, dans le réseau eDonkey, les pairs peuvent se connecter à n'importe quel serveur: si un utilisateur du serveur choisi partage un fichier qui les intéresse, ils pourront alors se connecter à lui. BitTorrent, quant à lui, oblige les utilisateurs à se connecter à un tracker précis, désigné dans le fichier *.torrent*.

Du fait de l'architecture semi-centralisé, le protocole BitTorrent souffre des mêmes faiblesses qu'eDonkey, à savoir le **risque de suppression d'un serveur**. Le risque est encore plus important du fait que tous les clients partageant le même fichier se connecteront au même serveur: si le tracker est coupé, il n'y aura plus la possibilité de télécharger le fichier sans modifier le fichier torrent renseignant sur l'adresse du tracker (et donc le remplacer sur tous les sites l'indexant). Pour pallier à ce problème, les *.torrent* peuvent aujourd'hui contenir plusieurs adresses de tracker, à utiliser en cas de panne. De même, il ne faudra pas non plus oublier de prendre en compte les problèmes liés aux ressources du tracker (bande passante, capacité de traitement). Quand des milliers de personnes veulent le fichier, il faut pouvoir leur répondre.

Une différence majeure entre BitTorrent et les autres protocoles Peer-to-Peer est qu'il ne possède **aucun moteur de recherche intégré**. Il faut donc utiliser un logiciel intermédiaire (site web, blogs, etc) pour trouver les *.torrent*. Cependant, de plus en plus de navigateurs web peuvent intégrer un client BitTorrent et un mini moteur de recherche de fichier *.torrent*, rendant le téléchargement très facile.

¹⁷ http://wiki.theory.org/BitTorrentSpecification#Super_Seeding

Ce point peut également être vu comme un avantage pour le réseau. Les réseaux Peer-to-Peer souffrent globalement des fichiers truqués ou contenant des virus. Des réseaux comme FastTrack ou eDonkey facilitent leur propagation, les applications utilisant leur propre moteur de recherche recherchant les fichiers sur les ordinateurs des clients. Les fichiers sont ainsi partagés indéfiniment entre les utilisateurs ne se rendant pas compte que le fichier est truqué. Il reste toujours des traces du fichier sur un des pairs du réseau, qui est successible de l'envoyer à d'autres pairs. Sur un réseau BitTorrent, il suffit de localiser le fichier torrent et de le **supprimer du site web pour le rendre inaccessible aux différents pairs**, ce qui évite la trop grande propagation du fichier truqué.

Cependant, ces sites web peuvent eux-mêmes faire l'objet d'attaques comme ça a été le cas pour le site **suprnova.org**¹⁸, un des sites les plus utilisés pour l'indexation des fichiers torrent qui a été fermé suite aux actions des sociétés gérant les droits d'auteurs. Cependant, après une petite période de doute, ces sites sont très vite remplacés par d'autres ce qui permet au réseau de continuer à fonctionner malgré les attaques incessantes.

Enfin, contrairement aux autres réseaux actuellement populaires tels que FastTrack et eDonkey, BitTorrent ne gère **pas de files de d'attente** de téléchargement. Les pairs sont évalués toutes les 10 secondes et le client choisi de fournir des données à ceux qui lui en offrent le plus en retour.

Ce système permet d'accroître la **vitesse des téléchargements**, contrairement à eDonkey où il faut souvent plusieurs jours pour télécharger de gros fichiers.

Cette rapidité des téléchargements provoque aussi une **obsolescence rapide des fichiers** partagés. En effet, une fois le fichier téléchargé, les utilisateurs ferment la fenêtre du client, stoppant ainsi le partage. Ainsi, la durée de vie de partage des documents n'est en général que de quelques mois. Le réseau favorise donc le partage de fichier récent et très demandé.

On notera cependant, que la vitesse initiale de téléchargement est souvent faible étant donné que le client n'a rien à uploader. Il faut donc attendre un certain temps pour voir ses débits augmenter et compter sur l'algorithme d' "optimistic unchoking".

¹⁸ Voir <http://www.essentielpc.com/actualites/511-suprnova-bittorrent.html>

2.2.8 Développement de BitTorrent

Différentes extensions, officielles ou non, sont prévues dans le développement de BitTorrent.

Parmi celles-ci, on peut citer les **connexions cryptées**¹⁹, permettant la création de liaisons cryptées entre les pairs, protégeant ainsi le contenu des messages des regards extérieurs.

On pourra aussi noter la présence d'une extension permettant le **WebSeeding** autorisant un serveur HTTP à servir de "seed" pour les téléchargements et ainsi combiner le protocole BitTorrent avec le protocole HTTP.

Enfin, une autre extension importante vise à supprimer un point faible du protocole: les trackers qui centralisent le réseau. Les nouvelles versions de BitTorrent supportent des connexions sans tracker. Le client implémente alors une **table DHT**²⁰ pour identifier et stocker la liste des différents clients partageant le fichier qui l'intéresse. Cette DHT, appelée "Khashmir" est basé sur le protocole **kademlia**.

Chaque client génère un "Node ID" unique, correspondant en général au Hash SHA-1 de son adresse IP. Il se connecte ensuite au réseau en utilisant l'adresse IP d'un pair déjà présent. Les données sont ensuite retrouvées en utilisant le principe "clé => valeur". La clé correspond au fichier demandé et la valeur correspond à l'ID du client possédant ce fichier. Pour obtenir ces valeurs, le client envoie le Hash SHA-1 du fichier recherché sur le réseau. Si les voisins n'ont pas le fichier, ils propagent la demande sur le réseau, sinon ils lui répondent.

¹⁹ Voir: http://www.azureuswiki.com/index.php/Message_Stream_Encryption

²⁰ Distributed Hash Table - http://www.bittorrent.org/Draft_DHT_protocol.html

3 Identification des risques

Le développement massif des Peer-to-Peer, souvent gratuits, est accompagné de nombreux problèmes de sécurité pour les usagers. En effet, ces logiciels sont utilisés, la plupart du temps, à une très grande échelle, tout le monde pouvant se connecter à tout le monde, ouvrant ainsi la porte à des individus mal intentionnés. Il est cependant à noter qu'une partie des attaques contre les réseaux Peer-to-Peer sont organisées par les sociétés de droits d'auteur souhaitant protéger les fichiers dont le contenu est régi par la loi du copyright.

3.1 Virus et Spyware

Les P2P sont très connus comme étant une des sources permettant la propagation des virus et spyware²¹ à l'échelle mondiale. En effet, des millions d'utilisateurs sont connectés entre eux à travers les réseaux P2P, ce qui constitue un facteur très intéressant pour les créateurs de virus, ceux-ci pouvant se propager très facilement d'utilisateur en utilisateur.

Parmi ceux-ci, on trouve notamment les chevaux de Troie qui sont contenus dans des fichiers quelconques susceptibles d'être téléchargés par des utilisateurs. Lorsque ceux-ci installent le fichier sur leur ordinateur, le Trojan s'implante en même temps et crée des « portes dérobées » sur l'ordinateur.

Ce problème n'est pas spécifique aux P2P, mais il constitue un des problèmes majeurs de ces réseaux puisqu'ils permettent de les véhiculer très facilement. Il est d'autant plus important de bien les surveiller dans les grandes entreprises: un virus infectant un des pc de l'entreprise se propagera très rapidement sur tous les postes de travail, risquant de paralyser le réseau de la société. La "back door" pourrait également être utilisée pour accéder au contenu des dossiers de l'entreprise, provoquant une fuite d'information irréversible pour celle-ci.

Dans le cadre des particuliers, les back-doors peuvent permettre de prendre le contrôle des ordinateurs. On peut en faire différents usages, et notamment s'en servir pour attaquer des serveurs distants ou prendre le contrôle d'autres machines tout en restant caché puisque les attaques ne viennent pas directement de nous (les ordinateurs infectés servent alors de proxy).

Ainsi, « en 2002 le virus Duload se propage via KaZaa quelques mois après l'apparition et la diffusion par le même biais du cheval de Troie K0wBot et du ver Benjamin [...] En avril 2005, le vers Nopir-B (se propageant par les réseaux P2P) s'attaque aux fichiers MP3 [...] »²².

Enfin, on peut également citer le cas où le logiciel P2P lui-même est infecté comme par exemple le logiciel KaZaA dans lequel le spyware « Cydoor²³ » était installé en même temps que le programme lui-même, dans la version officielle. Ce spyware, utilisé par la société du même nom à des fins commerciales, permettait de télécharger des publicités sur le pc de l'utilisateur ou encore de récupérer certaines données sur lui.

²¹ Spyware : logiciel espion s'installant sur les ordinateurs, à l'insu de leurs utilisateurs, et récoltant des données sur ceux-ci pour ensuite les envoyer aux personnes qui pourront les exploiter.

²² MISC N°25 - Mai Juin 2006 - p24 à p47

²³ Voir : <http://en.wikipedia.org/wiki/Cydoor>

3.2 Fichiers truqués

Poison Peers :

Le réseau Peer-to-Peer comme BitTorrent introduisent le problème des "Poison Peers": un pair fait croire au tracker qu'il a un fichier complet. Celui-ci va donner l'IP du pair malicieux à d'autres pairs, qui vont se connecter à lui. Le "faux seeder" va alors pouvoir **envoyer des morceaux de parties erronées** (aussi appelé « **polluting attacks** »).

Certes, les clients vont détecter ces parties truquées une fois qu'ils auront le morceau complet (grâce à la somme MD4 ou SHA1), mais si le nombre de « Poison Peers » est important, alors cela va encombrer le réseau et faire perdre beaucoup de temps et de bande passante aux différents pairs, ce qui va globalement réduire l'efficacité du réseau.

Ces attaques peuvent aussi prendre la forme de fichiers dont **le nom ou la description ne correspond pas au contenu réel du fichier**. Les fichiers sont en effet placés sur le réseau par les utilisateurs, qui fournissent eux-mêmes les informations le décrivant (titre, résumés, etc.).

Certains utilisateurs n'hésitent pas à placer des faux noms de fichiers. Cela leur permet par exemple d'augmenter leur score dans les files d'attente en utilisant un faux fichier dont le nom est très demandé (dernier film sorti par exemple), etc.

Les utilisateurs téléchargent ensuite le fichier pensant trouver le bon et finissent par se retrouver avec un « fake ». Ces attaques peuvent causer beaucoup de tord au réseau, le temps et les ressources perdus pouvant être très important.

Partage automatique des données:

Comme nous l'avons vu lors de l'étude du fonctionnement d'eDonkey et de BitTorrent, les réseaux Peer-to-Peer utilisent en général le fractionnement des fichiers pour accélérer les téléchargements: un pair peut commencer à envoyer des morceaux d'un fichier qu'il ne possède pas en totalité. Le problème de cette méthode est que l'utilisateur n'a pas la possibilité de visualiser ce fichier (du moins pas dans sa totalité) lors du téléchargement.

Si ce fichier est un « fake », l'utilisateur ne le saura que lorsqu'il sera complet et pourra alors le supprimer. Cependant, avant d'avoir le fichier complet, il aura déjà envoyé de nombreuses parties du fichier truqué à d'autres pairs, qui eux-mêmes l'auront propagés, etc. Le fichier sera ainsi envoyé de pair en pair et pourra très difficilement être totalement supprimé du réseau. Le mode de fonctionnement des Peer-to-Peer favorise donc la vitesse des téléchargements, mais par la même occasion, et il facilite également la propagation des fichiers truqués, des virus, etc.

La lutte contre les « fake » :

Comme nous l'avons vu dans notre analyse des deux principaux P2P actuels, il existe différentes façon de lutter contre les fichiers truqués.

Dans le cas de BitTorrent, le problème est plus simple : pour supprimer un fichier, il faut supprimer le *.torrent* du site qui l'indexe dès que le fichier est détecté comme faux, ce qui le rendra inaccessible à de nouveaux téléchargement. Cependant, ce n'est pas aussi simple que ça quand on sait qu'il existe de nombreux site d'indexation de fichier torrent, et que cela se fait de plus en plus de manière automatique.

Dans le cas d'eDonkey, la solution envisagée repose principalement sur le bénévolat des utilisateurs, qui ont la possibilité de **commenter et d'évaluer un fichier**. Très peu d'utilisateurs ont cependant le réflexe de visualiser ces commentaires, et les pirates ont eux-mêmes la possibilité de commenter leur fichier pour faire croire que la qualité est excellente et que c'est le bon fichier.

Les **fonctions de prévisualisation**, notamment utilisés par le client eMule peuvent aussi permettre de réduire le « coût » d'un fichier truqué en le détectant et le supprimant avant la fin du téléchargement. Encore faut t'il pour cela posséder la première partie du fichier afin de lancer la visualisation... (le téléchargement de la première et dernière partie est prioritaire par défaut dans le configuration d'eMule afin de permettre cette visualisation).

Dans tout les cas, ces solutions ne permettent pas de pleinement stopper la propagation des fichiers truqués dans un délai suffisamment court pour empêcher une propagation mondiale.

Les solutions envisagées pour le problème des fichiers truqués sont principalement basées sur les actions des utilisateurs qui doivent rester attentifs à ce qu'ils téléchargent et partagent (regarder les commentaires, visualiser le document dès que possible, etc.). Ceci ne concerne malheureusement qu'une petite partie des usagers.

De plus, nous l'avons déjà expliqué lors de l'étude d'eDonkey, les utilisateurs ont très rarement le réflexe de prévenir les autres utilisateurs. Lorsqu'ils détectent un « fake », ils se contentent en général de détruire le fichier. Or, pour pouvoir commenter un fichier (et donc prévenir qu'il est truqué), il faut le posséder dans sa liste de partage...

Le problème des fichiers truqués reste donc un problème majeur des Peer-to-Peer et aucune « solution miracle » n'a encore été trouvée.

Les principaux « pirates » utilisant ces techniques sont toujours les sociétés de droits d'auteurs qui tentent de dissuader les utilisateurs de télécharger des contenus protégés en ralentissant / bloquant le téléchargement des vrais fichiers. Des entreprises se sont même spécialisées dans ce genre d'attaque avec des logiciels comme « overpeer²⁴ » ou « P2P Overflow ».

²⁴ Voir <http://www.zdnet.fr/actualites/internet/0,39020774,39294599,00.htm>

3.3 *Failles de sécurité*

Faille dans le code source :

Les logiciels utilisés pour les échanges P2P sont pour la plupart "open source". Le code source des clients/serveurs est très souvent disponible sur internet. N'importe qui peut donc y accéder et l'analyser, ce qui en général est considéré comme plus sécurisé car les codes peuvent ainsi être étudiés par la communauté.

Certaines personnes mal intentionnées ont cependant, par la même occasion, la possibilité de découvrir des failles de sécurité dans le code source et de les exploiter. Dans le cas d'un code source fermé, la retro-ingénierie (reverse engineering) permet de découvrir les problèmes de sécurité par l'analyse du code machine. Ces logiciels étant présent à des millions d'exemplaires à travers le monde, l'exploitation de ces failles est très intéressante pour les pirates car elle peut leur permettre de prendre la main sur de nombreuses machines.

Comme tout programme informatique accessible par le réseau, les applications Peer-to-Peer sont sujettes à des attaques de type stack/buffer overflow, etc., que ce soit à cause de défaillance dans leur propre code ou dans les bibliothèques qu'ils utilisent (voir le paragraphe 2.1.8 d'eDonkey, relatif aux problèmes d'utilisation de bibliothèque externe).

3.4 *Spoofing ip : Restrictions des IP / Usurpation d'identité*

De plus en plus, les logiciels P2P implémentent la restriction de plages IP pour se connecter et partager.

En effet, pour éviter que les autorités se connectent au réseau et tente ainsi de perturber son fonctionnement ou de récupérer des données sur les utilisateurs, les clients et serveurs P2P autorisent uniquement les plages d'IP des FAI, des universités, et de quelques réseaux d'entreprises. Plusieurs listes sont mises à jour par des communautés et disponibles²⁵ sur internet pour bloquer les adresses connues à éviter.

Activation du filtrage d'IP dans eMule :

eMule contient donc une option permettant de filtrer des adresses « dangereuses ». Pour cela, il faut aller dans le menu "*préférences*", "*sécurité*" et ajouter dans le champ « *Update From URL* » le chemin vers un fichier contenant les plages IP. Une liste est disponible à l'adresse suivante :

<http://www.bluetack.info/nipfilter.dat.gz>

L'utilisation de *PeerGuardian*²⁶ se développe aussi. Il permet d'interdire, directement dans le système d'exploitation, la connexion vers des adresses néfastes (typiquement les organismes anti-P2P). Un autre logiciel disponible est *ProtoWall*. Pour Azureus (Client BitTorrent), il existe un Plugin *SafePeer* qui effectue lui aussi le filtrage d'IP pour le logiciel P2P.

²⁵ Liste de plages d'adresses IP : <http://test.blocklist.org/>

Logiciel de mise à jour de la liste : BlockList Manager

²⁶ <http://phoenixlabs.org/pg2/>

Ces dispositions obligent les autorités à masquer leur réelle adresse internet pour surveiller les réseaux (spoofing ip), ce qui offre à l'utilisateur la possibilité de se retourner contre eux en cas de litige (utilisation d'une fausse identité).

Ces techniques d'usurpation d'identité ne sont cependant pas utilisées uniquement par les autorités, mais peuvent, à l'inverse, jouer contre elles. Ainsi, certains pirates n'hésitent pas à se faire passer pour d'autres personnes. Par exemple, cela peut leur permettre de télécharger des fichiers avec une fausse adresse évitant ainsi d'être détecté par les sociétés de droits d'auteurs espionnant le réseau, ou encore permettre d'attaquer un serveur (dénie de service, etc.) en restant « caché ».

Dans un autre genre, le spoofing IP pourrait être utilisé pour récupérer la notoriété d'un utilisateur. Les clients P2P utilisent en général des systèmes de crédits basés principalement sur le taux d'envoi de données par les utilisateurs. Se faire passer pour un utilisateur ayant acquis beaucoup de crédits permettrait de télécharger plus rapidement, car nos demandes seront prioritaires, sans avoir à émettre vers les autres pairs.

Enfin, cette technique pourrait permettre du nuire au réseau. Par exemple, sur le réseau eDonkey, lorsqu'un utilisateur « abuse » des ressources, c'est à dire si il effectue de trop nombreuses requêtes vers les serveurs par exemple, il sera placé sur une liste noire et donc banni du réseau. De même, s'il effectue plus d'une requête toutes les dix minutes pour un même fichier, il sera banni par le pair qu'il interroge de façon trop agressive²⁷. Une attaque pouvant être envisagée consisterait alors à se faire passer pour d'autres pairs en usurpant leur adresse IP et les utiliser pour effectuer des nombreuses requêtes vers d'autres membres du réseau. Ces membres seraient alors bannis les uns après les autres du réseau, ce qui lancé à grande échelle pourrait fortement nuire au réseau.

NOTE : *Nous n'avons cependant trouvé aucun document affirmant ou infirmant cette possibilité d'attaque.*

Autre utilisation du filtrage :

Le filtrage d'adresse IP non pas pour unique but de se protéger des autorités. Il peut aussi être utilisé pour former une communauté, par exemple une communauté francophone²⁸. Cela peut permettre de partager des fichiers avec uniquement nos amis, ou améliorer les recherches en se connectant uniquement à des pairs utilisant la même langue, etc.

²⁷ <http://www.emule-speed.com/tutoriaux/emuletotale/banissementclient.htm>

²⁸ <http://lugdunum2k.free.fr/francophone.shtml>

3.5 Utilisation des ressources / DDoS

Les réseaux P2P sont réputés pour être très gourmand en terme de ressource système (temps processeur, place disque, puissance de calcul) et en bande passante. Selon certaines sources, les échanges P2P représenterait 50 à 80% des échanges²⁹.

Pour les entreprises, cela peut entraîner de gros problèmes de **surcharge du réseau et ralentir l'ensemble des connexions** (en particulier si les utilisateurs n'ont pas limités leur bande passante en upload).

C'est le cas notamment avec l'utilisation du *logiciel Skype*. Ce logiciel (gratuit), très pratique pour passer des appels téléphonique en utilisant la voix sur IP, utilise la technologie P2P, dans laquelle chaque pair peut agir comme serveur, pour ses communications, mais également comme relais pour le routage et le transport des appels des autres pairs³⁰. Ainsi, même inactif (pas d'appel en cours), le logiciel va continuer à consommer des ressources, qui peuvent être importantes, notamment en terme de bande passante, pour les entreprises. Ce problème a mené certaines de celles-ci à le bannir complètement de ses postes de travail.

De plus, l'utilisation de la bande passante dans les réseaux de partage de fichier est en général très peu optimisée : les données sont envoyées au cas par cas. Ainsi, un client répond aux demandes des autres pairs les unes après les autres, alors qu'il y a peut être 100 clients qui demande la même chose. Une optimisation des réseaux P2P pourrait donc être l'utilisation d'échanges multicast pour le partage de données.

Plusieurs logiciels se sont développés pour tenter de réduire l'utilisation des ressources, notamment en termes de bande passante. En voici deux exemples :

BitTyrant³¹

Basé sur le logiciel BitTorrent *Azureus*, BitTyrant prétend offrir des téléchargements jusqu'à 70 % plus rapides que son aîné. Il a été imaginé par des étudiants et chercheurs de l'Université de Washington, qui ont ajouté à Azureus un système de ratio qui favorise les taux de téléchargement de ceux qui uploadent le plus.

PeerCache

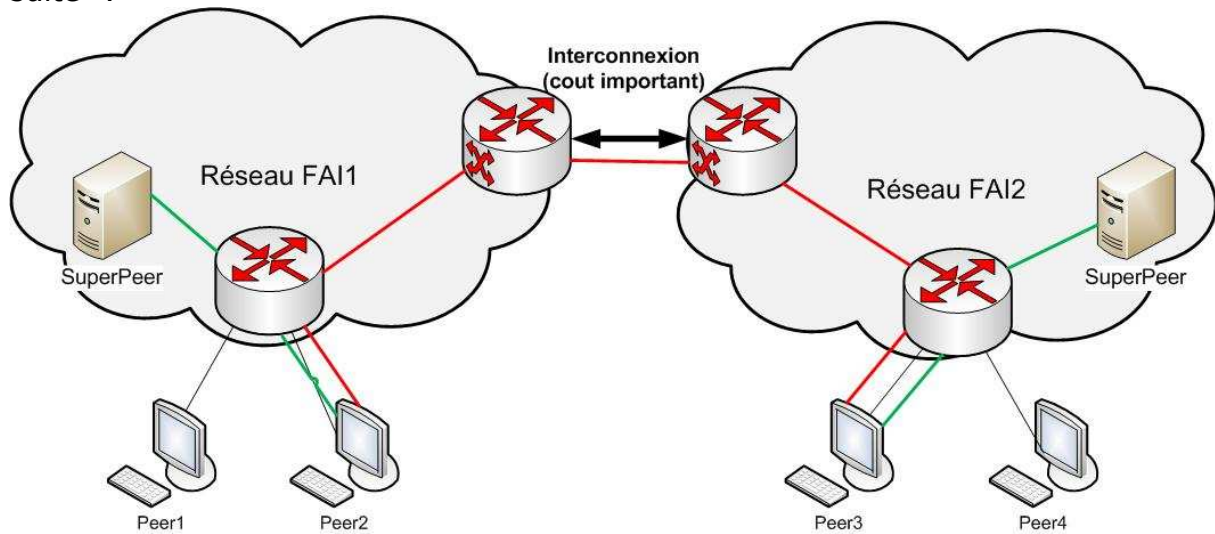
Afin de limiter le trafic sortant du réseau, il est possible d'utiliser un serveur cache pour P2P, agissant comme une sorte de proxy. Pour le réseau FastTrack, il existe *PeerCache*. Cependant, ce type de serveur est principalement utile pour les FAI afin de concentrer le trafic dans son réseau. Le problème se pose de la légalité du cache stocké sur un tel serveur. Les échanges Peer-to-Peer sont en effet, en majeure partie, réputé illégaux. Le fait qu'un fournisseur d'accès améliore l'efficacité des échanges P2P pose donc certains problèmes.

²⁹ MISC N°25 - Mai Juin 2006 - p24 à p47

³⁰ <http://www.reseaux-telecoms.net/actualites/lire-quand-skype-consomme-de-la-bande-passante-dans-votre-dos-5158.html>

³¹ http://www.ratiatum.com/log1097_BiTyrant.html

Wanadoo NL a testé la solution mais des problèmes légaux sont survenus de suite³².



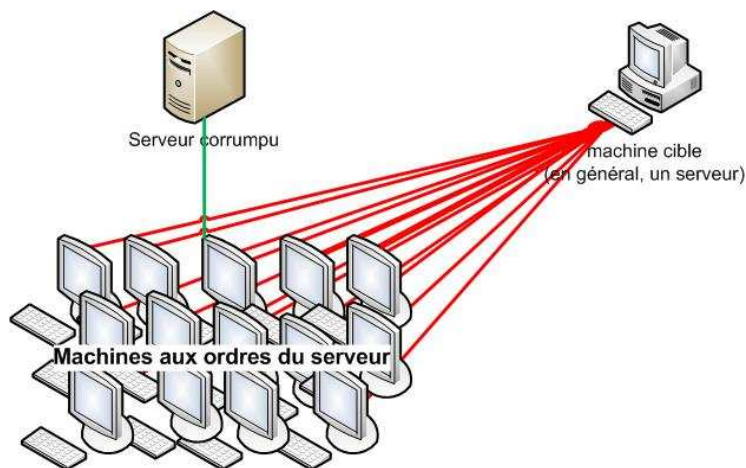
Principe de PeerCache

*En rouge, le flux P2P sans peerCache a un cout important à cause de l'interconnexion.
En vert, le flux P2P avec peerCache a un cout plus faible car le trafic reste au sein de son réseau.*

Attaque par dénis de services :

Les réseaux P2P demandant énormément de ressources et impliquant des milliers d'utilisateurs, ils peuvent être détournés pour lancer des attaques par dénis de services (DoS) et dénis de services distribués (DDoS).

Comme nous l'avons déjà expliqué dans le paragraphe 2.1.8 concernant les problèmes sur eDonkey, les réseaux P2P peuvent être utilisés pour lancer des attaques distribués vers une cible. Par exemple, un serveur eDonkey (ou un tracker BitTorrent) spécialement préparé peut donner les mêmes adresses cibles à tous les clients qui vont se connecter à eux.



Principe du déni de service distribué (DDoS)

Ainsi, des centaines de machines vont tenter de se connecter sur une même adresse en la surchargeant de demande et risquant de la saturer. Ces attaques peuvent être lancées sur n'importe quel type de matériel accessible par le réseau (serveur, poste de travail, routeur, etc.).

³² <http://www.transfert.net/Wanadoo-Pays-Bas-utilise-un>

Il est ainsi « facile » de s'attaquer aux réseaux de plusieurs entreprises. De plus, ces attaques sont « discrètes » du point de vue du pirate car il ne fait que fournir l'adresse cible. Jamais son adresse ne sera considérée comme attaquante sur le réseau de l'entreprise.

Les serveurs permettant d'établir les connexions P2P (les trackers pour BitTorrent) peuvent eux-mêmes être la cible de DDoS. Comme nous l'avons déjà répété plusieurs fois, ces serveurs sont un énorme point des réseaux centralisés et hybrides. Des attaques par DDoS, causées par la présence de virus sur de nombreux poste de travail par exemple, peuvent ainsi surcharger les serveurs centralisant les connexions, les rendant complètement indisponible pour les utilisateurs normaux, ce qui peut provoquer une chute considérable des performances du réseau : « *The distributed denial-of-service (DDoS) attack on the BitTorrent infrastructure prevented some users from downloading files for up to 10 hours on Wednesday* »³³.

3.6 Actions en justice

La principale ennemie des réseaux P2P de partages de fichiers sont les sociétés de droits d'auteurs et les Majors Américains qui souhaitent protéger les copyrights.

Les réseaux P2P Hybrides, qui sont les plus populaires, ont maintes fois montrés leur résistance face aux attaques informatiques (DDoS, etc.) ou à la suppression de certains serveurs très populaire :

- ⇒ suppression des serveurs Razorback pour edk
- ⇒ suppression des sites d'indexations / tracker (ex : suprnova.org) pour BitTorrent

Cependant, les actions en justice finissent toujours par causer énormément de problèmes aux réseaux P2P, comme la montrée la mort de Napster, le premier P2P mondialement connu, où la fin prochaine de KaZaA.

Les nouvelles lois, comme par exemple la loi DADVSI³⁴ en France (transposition de la directive européenne 2001/29/CE), permettent aujourd'hui d'incriminer les utilisateurs des réseaux P2P.

Certains pays vont même plus loin en s'attaquant directement aux développeurs de logiciels P2P.

Ainsi, Un créateur de P2P japonais a été condamné en justice pour la création et la mise à disposition d'un logiciel de P2P facilitant l'échange de fichiers copyrightés. Ce chercheur de l'Université de Tokyo avait été arrêté en mai 2004 pour son projet de logiciel inspiré de Freenet. Ce type de condamnation peut rapidement freiner l'expansion du P2P³⁵.

³³ http://news.zdnet.com/2100-1009_22-5473754.html (2004)

³⁴ <http://fr.wikipedia.org/wiki/DADVSI>

³⁵ <http://akosh.pcinpact.com/actu/news/33513-winy-P2P-freenet.htm>

Digital Rights Management (DRM)

La gestion numérique des droits (en anglais *Digital Rights Management* ou *DRM*) a pour objectif de contrôler par des mesures techniques de protection l'utilisation qui est faite des œuvres numériques.

Les P2P sont souvent assimilés à des logiciels pour « pirater » des œuvres protégés par les droits d'auteurs. Ils pourraient ainsi être lavés de tout soupçon s'ils intégraient une solution de DRM dans leur implémentation³⁶. Cependant, la popularité du P2P serait surement moindre en perdant une grande partie des fichiers partagés, comme la montré l'échec de Napster après sa transformation en logiciel P2P de partage de musique légale.

3.1 Administration difficile

Un des soucis des systèmes d'informatiques dans les entreprises est la surveillance du réseau et de tout ce qui si passe. Ceci passe en général par une centralisation des accès sur des serveurs centraux qui peuvent être protégés et surveillés par le SI.

Or, malgré tous leurs avantages, les Peer-to-Peer sont créés justement pour ne pas avoir ces serveurs centraux. Cela complique considérablement les tâches d'administration ce qui explique que les P2P sont encore sous exploités dans les entreprises.

La solution du P2P, bien que plus économique et robuste que le modèle client/serveur (si le serveur tombe en panne, plus rien de marche...), reste difficilement contrôlable, notamment en terme de sécurité sur l'ensemble des données qui transite, et des autorisations d'accès.

N'importe qui peut devenir un pair sur le réseau, mais tout le monde ne peut pas forcément avoir accès à toutes les données. Des listes d'autorisations doivent donc être gérées et nécessitent donc l'utilisation d'un serveur central ou la configuration sur chaque machine...

Grâce aux nouvelles techniques permettant d'utiliser des serveurs proxy, ainsi que les principes d'offuscation et de cryptage, il est très difficile aujourd'hui pour les entreprises de bloquer les trafics P2P. Cela peut leur causer de gros problèmes au niveau de la **confidentialité des données**, qui peuvent être partagés en quelques secondes à l'autre bout de la planète.

Pour citer un exemple flagrant, « *En 2005 au Japon, des données sensibles sur une installation nucléaire se sont retrouvées sur les réseaux P2P parce qu'un consultant avait malencontreusement lancé sur son portable un logiciel de partage de fichiers [...] Dans la même veine, des données confidentielles sur les troupes américaines stationnées en Irak auraient transité via Gnutella...* »³⁷.

Même si des solutions de filtrage existent, elles ne sont jamais évidentes à mettre en place (quels logiciels doivent être bloqués ?, etc.).

³⁶ Peer-to-Peer Networking and Digital Rights Management: How Market Tools Can Solve Copyright Problems par Michael A. Einhorn and Bill Rosenblatt

<http://www.cato.org/pubs/pas/pa534.pdf>

³⁷ MISC N°25 - Mai Juin 2006 - p24 à p47

3.2 Absence d'anonymat

Contrairement à la problématique des réseaux informatiques d'entreprises qui ont besoin d'identifier leurs utilisateurs pour des raisons de sécurité, un des principaux problèmes des réseaux Peer-to-Peer « publics » présent actuellement sur le marché est l'absence d'anonymat des utilisateurs. Ce problème est devenu un des points majeur du développement des P2P, poussé principalement par les actions de surveillances des réseaux effectués par les sociétés de droits d'auteurs et la justice internationale.

3.2.1 Identification par adresse IP

Pour des raisons pratiques dues au fonctionnement des réseaux informatiques, les utilisateurs sont toujours identifiés par leur adresse IP ou un dérivé (les HightID sous eMule par exemple) qui est nécessaire pour pouvoir établir une connexion d'un pair à un autre.

De plus, ces IP sont, dans le cas des P2P populaires présentés précédemment, stockés sur les serveurs centralisant les connexions, liés au nom des fichiers partagés, afin de les fournir facilement aux différents pairs sur le réseau. Il est donc assez facile de déterminer qui est connecté et ce qu'il partage, d'où un manque total d'anonymat pour les utilisateurs.

Une solution consisterait à utiliser un serveur proxy pour se connecter au P2P. Cependant, ceux-ci sont gourmand en ressource et ralentissent fortement le trafic, et tout le monde n'a pas un proxy sous la main pour se connecter...

3.2.1 Identification des protocoles

Les firewalls « traditionnels » analysant les couches 3 (IP) et 4 (TCP, UDP) sont de moins en moins efficaces pour protéger les ordinateurs car beaucoup de P2P peuvent utiliser le port 80, ouvert la plupart du temps pour permettre l'accès aux serveurs web.

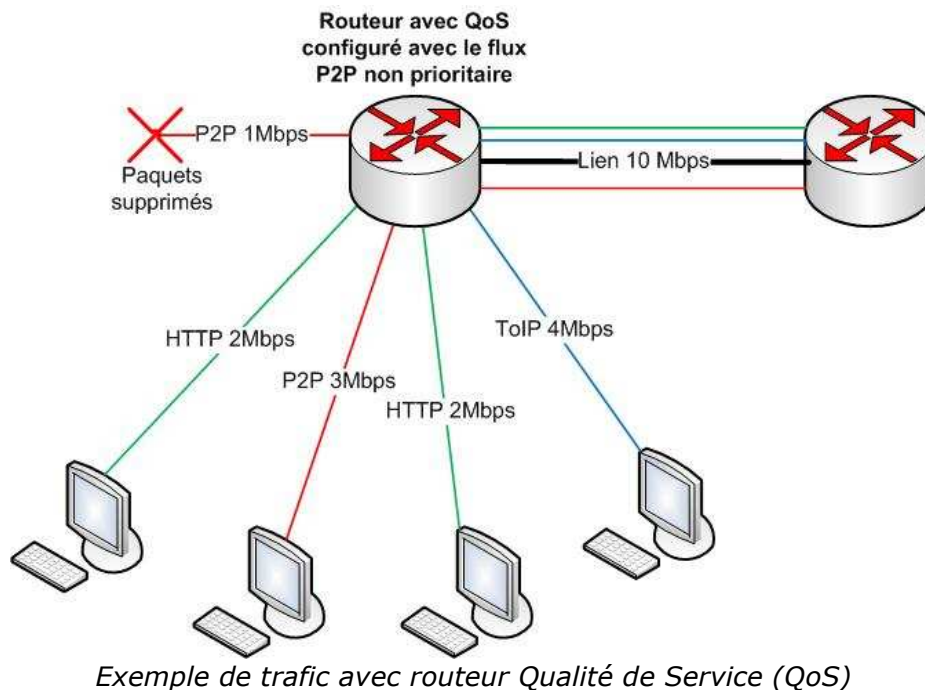
Une première solution consistait à bloquer les ports et les adresses IP des serveurs connus. Cependant, le réseau étant dynamique et le nombre de serveur important, il est très difficile de bloquer à coût sûr les connexions. Quant au numéro de port, ils sont facilement configurable sur de nombreux P2P.

Ce problème a aujourd'hui amené au développement de différents outils permettant d'**analyser le trafic réseau au niveau 7, c'est à dire au niveau applicatif**. Même si ils sont un peu plus gourmand en termes de ressources, ils sont de plus en plus efficaces.

Ces outils apparaissent de façon croissante dans les entreprises et permettent, entre autre, de filtrer le trafic Peer-to-Peer.

En effet, les trames utilisées pour les échanges P2P sont en général facilement identifiables, ce qui peut permettre de couper/bloquer facilement les connexions. Par exemple, dans le protocole BitTorrent, lorsqu'un pair se connecte à un tracker, il envoie une requête HTTP en indiquant "*User-Agent: BitTorrent*". Il est donc facile de bloquer ce type de transmission à l'aide d'une **analyse protocolaire**.

Les flux P2P étant très important, certains fournisseurs d'accès, tel que Free sur son réseau non dégroupé³⁸ utilise également des routeurs permettant de faire de la qualité de service en équilibrant les flux P2P par rapports aux autres services, voir les réduire au maximum, ce qui fait que peu ou pas de flux P2P peuvent traverser le réseau, bloquant ainsi ce moyen de communication aux utilisateurs.



Ces outils peuvent également effectuer une analyse de trafic afin de détecter un trafic anormal. Les logiciels P2P étant gourmand en termes de bande passante, ils peuvent également être détectés par ce moyen.

Exemple d'outils : L7-filter, IPP2P, P2PWALL, Snort

³⁸ <http://www.journaldufreenaute.fr/06/07/2006/le-filtrage-p2p-setend-en-zone-non-degroupee-ipadsl.html>

Cisco System Service Control :

http://www.cisco.com/cdc_content_elements/flash/csc/cisco_FINAL_no-loop.html

3.2.2 Solutions au problème de l'anonymat ?

L'amélioration des techniques de filtrage et d'identification des pairs sur le réseau ont amené parallèlement les P2P à se développer. Ainsi, différentes techniques d'offuscations du protocole (voir 2.1.8. Problèmes d'eDonkey et quelques solutions), de cryptage des connexions, etc., ont vu le jour pour tenter de remédier à ces problèmes.

Le principe utilisé par les P2P pour garder l'anonymat est de transiter des **flux chiffrés** par des pairs intermédiaires. En chiffrant les données, ces pairs ne sont pas conscients des données qui circulent. Le passage par des pairs intermédiaires permet de **dialoguer indirectement** et donc ne pas connaître l'identité du pair émetteur.

Cependant, le majeur inconvénient de cette méthode est la **diminution des performances**. En effet, le réseau passe d'un réseau pair à pair à un réseau pair à mandataires à pair. Un flux entre 2 pairs n'encombre plus seulement leurs deux bandes passantes, mais aussi les bandes passantes des mandataires. La bande passante maximum entre les deux pairs est égale au minimum des liens entre les pairs et mandataires et entre les liens entre les mandataires.

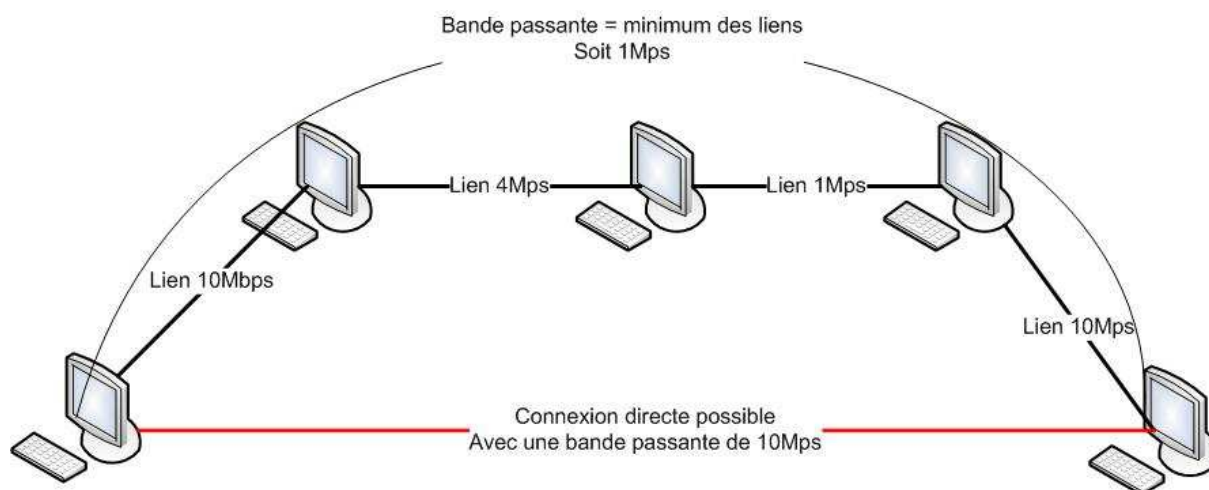


Schéma d'un réseau P2P avec mandataire

Les principaux réseaux chiffrés et anonymes sont *Ants*, *Freenet*³⁹, *Share*, et *Kameleon*. Ils ne sont pour le moment pas très populaires mais avec l'augmentation du nombre de procès pour téléchargement illégal, ces réseaux vont probablement se développer très rapidement.

Pour contourner ce problème d'anonymat, une autre solution est le **réseau d'amis** avec des relations de confiance. L'utilisateur permet à ses amis de joindre son réseau « privée » et ainsi de partager ses données uniquement avec eux. C'est le principe des logiciels comme *Waste*⁴⁰ et *Darknet*. L'inconvénient d'une telle solution est qu'il faut quelques pairs avec de grosse bande passante pour obtenir de bonnes performances.

³⁹ <http://freenetproject.org/whatis.html>

⁴⁰ <http://waste.sourceforge.net/>

4 Attaques spécifiques aux réseaux P2P

4.1 Sybil Attack

L'attaque Sybil est une attaque visant à éviter les systèmes de partages basés sur la réputation. Elle consiste à voler une identité (possédant beaucoup de crédit) ou à se forger de nombreuses identités pour casser le système de réputation⁴¹. Il est alors possible de se bâtir une très large influence sur le réseau, grâce à l'utilisation des différentes identités (chaque identité peut certifier que les autres sont des identités de confiance partageant beaucoup). Ainsi, le pair va pouvoir utiliser largement le réseau sans y apporter sa contribution. (Ex: download massif de données sans en partager.)

La résistance d'un système à une attaque de type Sybil dépend de la facilité avec laquelle il est possible de se créer des identités.

Le problème de la Sybil attacks peut être résolu en utilisant une autorité centrale de certification pour identifier les différents pairs sur le réseau, comme c'est le cas pour le réseau *Pastry*. Quand un pair malicieux est détecté, l'autorité peut prévenir tous les autres pairs. Cependant, on passe d'un réseau « pur » à un réseau « centralisé » ...

Dans le cas d'eDonkey, le problème de Sybil Attack est contré grâce au mécanisme de cryptographie asymétrique sous forme de défi/challenge et aussi résolu par une réputation locale et non globale, c'est-à-dire que chaque pair enregistre les pairs qui ont participé à ses besoins pour le gratifier plus tard.

4.2 Eclipse

L'attaque Eclipse est plus générale que l'attaque Sybil décrite précédemment et vise principalement les réseaux P2P structurés. Dans ces réseaux, chaque nœud maintient des pointeurs vers ses voisins.

L'attaque consiste alors à contrôler plusieurs pairs influant sur le réseau (une grande partie des voisins), pour ne pas rediriger le trafic vers les bons pairs (les pairs « sains »), et donc modifier l'utilisation normal du routage. L'attaquant cache alors les bons pairs sur le réseau, d'où le nom d'« attaque eclipse »⁴².

Cette attaque peut être lancée en utilisant l'attaque Sybil, c'est à dire en se forgeant de nombreuses identités sur le réseau, afin d'être présent comme voisin sur un maximum de nœud dans le réseau.

Une solution est l'utilisation d'une **autorité de certification** (CA) qui attribue et vérifie les identités du réseau. (cf. Bibliographie Secure Routing for structured P2P overlay network)

⁴¹ <http://www.cs.rice.edu/Conferences/IPTPS02/101.pdf>
<http://www.answers.com/topic/reputation-system>

⁴² Defending against eclipse attacks on overlay networks - Atul Singh Miguel Castro
Peter Druschel Antony Rowstron

4.1 **Pollution de DHT**

Les tables DHT (Distributed Hash Table) sont utilisées dans les réseaux structurés pour permettre l'acheminement des informations entre les différents pairs du réseau. Ces tables sont cependant basées sur la présomption que tous les nœuds sont des pairs de confiance.

Le principe de la pollution de DHT est de renvoyer des résultats erronés au demandeur. Cela peut concerner aussi bien des identifiants des fichiers, des IP et ports, qui seront falsifiés. Ainsi de propagation de pairs en pairs, l'ensemble du réseau va enregistrer et diffuser de mauvaises informations, le rendant complètement inopérant.

Un premier type d'attaque consiste à falsifier **les tables de routage**. Chaque pair participe au système de routage du réseau. Un nœud malicieux peut donc essayer d'introduire de fausses destinations, qui ne mèneront nulle part.

Les nœuds construisent leur table de routage avec les informations données par les autres nœuds. Si ces informations sont fausses, des nœuds sains risquent eux-mêmes d'effectuer un mauvais routage, ou encore de propager l'information erronée.

Ces attaques sont assez proches de l'attaque « eclipse » présentée dans le paragraphe précédent, sauf qu'ici, les nœuds ne sont pas forcément « éclipser », mais peuvent posséder de mauvaises informations et conduire à un mauvais fonctionnement de l'ensemble du réseau.

Ces attaques peuvent être contrées assez facilement par les pairs sains en testant si les destinations sont réellement accessibles avant de les ajouter dans leur table de routage. Les informations erronées données par des nœuds malicieux peuvent ainsi être détectées.

Lors de leur connexion dans le réseau, les nœuds sont aussi extrêmement vulnérables. Ils doivent en effet se connecter à d'autres pairs pour obtenir les informations nécessaires pour participer au réseau. Si un nœud malicieux parvient à être le contact des nouveaux arrivants, il va pouvoir leur fournir des informations erronées pour les faire entrer dans un réseau parallèle, contrôlé par le pirate.

Pour contrer cette attaque, les nœuds doivent garder des informations sur leur ancienne connexion afin de les comparer avec les données de la nouvelle. Il pourra ainsi détecter des anomalies.

Un autre type d'attaque consiste à fournir des informations lorsqu'on l'interroge **sur les données** dont il est responsable. Il peut par exemple dire qu'il possède certains fichiers, et refuser les connexions quand d'autres pairs souhaitent les obtenir, etc.

Un moyen de contrer les attaques sur les données est **la réplication** : si les données sont présentes sur plusieurs pairs, alors le client pourra interroger les réplicas pour savoir si la ressource est vraiment indisponible. Le but est d'éviter qu'un seul nœud soit responsable d'une ressource.

Dans le cas général, un réseau P2P sera fortement perturbé si plus de ¼ du réseau sont des pairs malicieux (cf. Bibliographie *Secure routing for structured peer-to-peer overlay networks*).

CONCLUSION

Les Peer-to-Peer sont aujourd'hui de plus en plus présents dans le monde informatique, que ce soit pour les simples utilisateurs à travers les réseaux d'échanges de fichiers, de messageries instantanés ou de voix sur IP, ou encore pour les entreprises, pour la mise en place d'applications réparties comme le partage d'agendas, etc.

D'une manière générale, les P2P actuelles souffrent de nombreux problèmes tels que les fichiers truqués, virus et autres... A cela s'ajoute le fait qu'il ne s'agit pas, la plupart du temps, de P2P « purs », mais de P2P « hybride », s'appuyant donc sur l'utilisation de serveurs centraux qui, malgré leurs avantages, constituent une véritable faiblesse.

L'utilisation de P2P comporte donc de nombreux risques, mais parallèlement à ces attaques, les défenses et protections se développent. Ainsi, face à l'amélioration du filtrage et de l'identification des pairs, les P2P offusqués, cryptés ou encore anonymes se développent et prennent de l'ampleur pour constituer les P2P de demain.

Face à ces améliorations, la meilleure solution pour éviter la fuite de données dans les entreprises consiste encore à éduquer ses employés.

Quant aux particuliers, il faut rappeler que le seul ordinateur réellement en sécurité est un ordinateur éteint ! Les risques inhérents aux P2P, bien que présents, restent marginal à la vue du nombre d'utilisateurs se connectant actuellement sur ces réseaux.

Bibliographie

Articles sur les réseaux P2P

<http://schuler.developpez.com/articles/p2p/>

<http://www.ratiatum.com>

<http://en.wikipedia.org/wiki/P2p>

MISC N°25 - Mai Juin 2006 - p24 à p47

A Survey and Comparison of Peer-to-Peer Overlay Network Schemes - Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma and Steven Lim - IEEE COMMUNICATIONS SURVEY AND TUTORIAL, MARCH 2004

A comparison of peer-to-peer architectures - Peter Backx, Tim Wauters, Bart Dhoedt, Piet Demeester - Broadband Communication Networks Group (IBCN),

A Survey of Peer-to-Peer Security Issues - Dan S. Wallach

Spam Attacks: P2P to the Rescue - Ernesto Damiani, S.De Capitani di Vimercati, Stefano Paraboschi, Pierangela Samarati, Andrea Tironi, Luca Zaniboni

Statistique sur l'utilisation P2P

<http://www.clubic.com/actualite-19921-toujours-plus-d-utilisateurs-de-peer-to-peer.html>

<http://www.oecd.org/dataoecd/55/57/32927686.pdf>

Différents réseaux Peer-to-Peer

⇒ **Quelques logiciels P2P de partage de fichier :**

BitTorrent : <http://www.bittorrent.com/>

Emule : <http://www.emule-project.net>

Freenet : <http://freenetproject.org/whatis.html>

KaZaA : <http://www.kazaa.com/fr/>

Napster : <http://www.napster.com>

Waste : <http://waste.sourceforge.net/>

⇒ **Projet SETI (partage de la puissance de calcul)**

<http://setiathome.free.fr/>

⇒ **Logiciel P2P de téléphonie sur IP**

Skype : <http://www.skype.com>

Fonctionnement d'eMule

⇒ **Système de crédit emule**

http://www.emule-project.net/home/perl/help.cgi?l=13&topic_id=202&rm=show_topic

⇒ **Serveur Lugdunum et sa configuration**

<http://lugdunum2k.free.fr/kiten.html>

⇒ **eMule, spécifications du protocole**

<http://www.cs.huji.ac.il/labs/danss/presentations/emule.pdf>

<http://sourceforge.net/projects/pdonkey/>

Fonctionnement de BitTorrent

⇒ **présentation de BitTorrent**

<http://harisson.free.fr/blog/index.php?2005/01/08/45-blog-web-et-bittorrent-petit-etat-des-lieux>
<http://developpeur.journaldunet.com/tutoriel/theo/060126-fonctionnement-bittorrent.shtml>
http://www.ratiatum.com/dossier1356_BitTorrent_Le_guide_complet.html
http://www.info.fundp.ac.be/~ven/CISma/FILES/40b3681836cb72004_dementen_gaetan.pdf

⇒ **protocole BitTorrent détaillé :**

<http://www.bittorrent.org/protocol.html>
<http://wiki.theory.org/BitTorrentSpecification>

Attaques contre les réseaux P2P :

⇒ **Spoofing**

P2P entrapment – incriminating peer-to-peer network users – septembre 2003

⇒ **Sybil attack**

<http://www.answers.com/topic/reputation-system>

The Sybil Attack - John R. Douceur Microsoft Research

(Article disponible sur : <http://www.cs.rice.edu/Conferences/IPTPS02/101.pdf>)

⇒ **Eclipse attack**

Defending against eclipse attacks on overlay networks - Atul Singh, Miguel Castro, Peter Druschel, Antony Rowstron - 2004

(Article disponible sur :

<http://portal.acm.org/citation.cfm?id=1133613&dl=ACM&coll=&CFID=15151515&CFTOKEN=6184618>)

Secure routing for structured peer-to-peer overlay networks - Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron and Dan S. Wallach

⇒ **Pollution DHT**

Security Considerations for Peer-to-Peer Distributed Hash Tables - Emil Sit and Robert Morris

Digital Right Management

⇒ **DRM**

Peer-to-Peer Networking and Digital Rights Management: How Market Tools Can Solve Copyright Problems - Michael A. Einhorn and Bill Rosenblatt