République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderahmane Mira de Béjaia

Faculté des Sciences Exactes
Départements d'Informatique
Ecole Doctorale Réseaux et Systèmes Distribués

Mémoire de Magistère
En Informatique

Option
Réseaux et Systèmes Distribués

Thème

Modélisation de la propagation de l'attaque Eclipse dans un réseau eDonkey

Présenté par

Inas NAKMOUCHE

Devant le jury composé de :

Président :	**KERKAR Moussa**	**Professeur**	**U. A. Mira, Béjaia, Algérie**
Rapporteur :	**BOUABDALLAH Abdelmadjid**	**Professeur**	**UTC de Compiègne, France**
Examinateurs :	**MELIT Ali**	**M.C.**	**U. de Jijel, Algérie**
	CHALLAL Yacine	**M.C.**	**UTC de Compiègne, France**
Invité :	**TOUAZI Djoudi**	**M.A**	**U. A. Mira, Béjaia, Algérie**

Promotion 2005-2006

RÉSUMÉ

Le monde informatique a été révolutionné par l'apparition des réseaux complètement décentralisés entre autres le pair-à-pair (P2P). L'évolution récente de ces réseaux et le nombre grandissant de leurs utilisateurs permet de les considérer comme le véhicule potentiel des différentes attaques pour réaliser une propagation rapide. L'objet de ce présent mémoire est la proposition d'un modèle de réalisation de l'attaque Eclipse dans un réseau eDonkey. Pour cela, nous nous sommes servis de l'attaque Ver. Ce dernier va exploiter la vulnérabilité de l'architecture héritée afin de se propager et de contaminer les serveurs ainsi que leurs clients en prenant pour cible les ports **UDP**, et ce, dans le but de supprimer les serveurs sains de la liste de serveurs des clients. Ainsi, il parviendra à subdiviser le réseau eDonkey en un ensemble de sous-réseaux isolés.

Mots clés : réseaux pair-à-pair, eDonkey, attaque Eclipse, propagation de Ver.

ABSTRACT

Computer world was revolutionized by the appearance of completely decentralized networks like peer-to-peer (P2P). The recent evolution of these networks and the number growing of their users allows considering them as the potential vehicle of the various attacks to achieve fast propagation. The object of this present report is the proposition Eclipse attack realization model in eDonkey network. For it, we used the Worm attack. This last one is going to exploit the vulnerability of the inherited architecture to propagate and to contaminate the eDonkey's servers and their clients by taking for target **UDP** ports, and that, with the aim of removal the correct servers of the client's server list. So, it will succeed in subdividing the eDonkey network into a set of isolated sub-networks.

Keywords : peer-to-peer networks, eDonkey, Eclipse attack, Worm propagation.

Dédicaces

A mes parents qui m'ont suivi, accompagné depuis mon jeune âge durant toutes mes études,

A mes sœurs et mes frères pour leur soutien et leur encouragements,

A ma grand-mère « Mamani » que DIEU la garde.

Remerciements

Je tiens tout d’abord à exprimer mes vifs remerciements à mon encadrant Abdelmadjid BOUABDALLAH, Professeur à l'UTC (Compiègne, France) pour m’avoir proposer ce sujet ainsi que ses conseils et ses orientations.

Je remercie Hani Ragab Hassen, Docteur à l'UTC (Compiègne, France) pour son aide, ses encouragements et la documentation qu’il m’a fournie.

Je remercie les membres de jury d’avoir voulu accepter de juger mon présent mémoire de magister.

Mes remerciements chaleureux iront à Mr. Kamel TARI, le Chef de département d'informatique et responsable de l'école doctorale, que je considère comme mon « parrain » pour son soutien moral, professionnel et sa grande générosité.

Je remercie également Mr. Djoudi TOUAZI responsable du centre de calcul, pour sa disponibilité permanente et son apport inépuisable.

Je remercie M^{elle}. Lamia Hamza, Mr. Ibrahim Zeaabta (IEEF/ Français) et Mr. Elhadi Dendene pour tout l’aide qu’ils m’ont apporté.

Enfin, je remercie toute mes collègues et en particulier Samera, Asma, sans oublier mes copines de chambre pour leur convivialité ainsi que tout ceux et celles qui ont contribué de prés ou de loin à la réalisation de ce travail.

« Que DIEU les bénisse tous »

LISTE DES ABVÉRIATIONS

AIM	AOL Instant Messages
AOL	America Online
AES	Advanced Encryption Standard
BBS	Bulletin board system
BDD	Base De Donnée
BFS:	Breadeth First Search
CPU	Central Process Unit
CAN	Content Adressable Network
CA	Certificat Authority
DNS	Domaine Name Service
DDNS	Distributed Domaine Name Service
DOS:	Denial Of Service
DDoS	Distributed Denial Of Service
DHT	Distributed Hash Table
DES	Data Encryption Standard
3DES	Le Triple DES
HTTP	Hypertext Transfer Protocol
ICQ	I Seek You
IP	Internet Protocol
IRC	Internet Relay Chat
ISP	Internet service provider
MSN	The MicroSoft Network
MD4	Message Digest 4
MD5	Message Digest 5
MFTP	Multisource File Transfer Protocol
MAC	Message Authentication Code
MitM	Man In The Middle
OpenBSD	Berkeley Software Distribution

P2P	Peer To Peer
PKI	Public Key Infrastructure
QoS	Quality of Service
RSA	Rivest Shamir Adleman
SHA1	Secure Hash Algorithm1
SDSI	Simple Distributed Security Infrastructure
TTL	Time To Live
TCP	Transmission Control Protocol
URL	Uniform Resource Locator
UDP	User Datagram Protocol

Table des matières

Liste des abréviations i

Table des matières iii

Liste des figures vii

Liste des tableaux ix

Introduction générale 1

1 Les réseaux pair-à-pair ……...4

1.1 Introduction …….. …..4

1.2 Généralité sur le pair-à-pair ……5

1.2.1 Définition……..5

1.2.2 Genèse……..5

1.2.3 Caractéristiques des systèmes P2P…….7

1.2.4 Objectifs de système P2P……..8

1.2.5 Comparaison entre P2P et client/serveur…….8

1.2.6 Avantages et Inconvénients du modèle P2P…….....9

1.2.7 Domaines d'application……..10

1.3 Les différents niveaux de décentralisation des réseaux P2P……..11

1.3.1 Modèle pur……..11

1.3.2 Modèle hybride……...……12

1.3.3 Modèle centralisé……..13

1.4 La structure d'un système pair à pair…..……...13

1.4.1 Architecture non structurée……..…...13

1.4.2 Architecture structurée…….…..14

1.4.3 Architecture légèrement structurée……..14

1.5 Les mécanismes de découvertes dans les systèmes P2P……...14

1.5.1 La recherche dans le système P2P non structuré……...15

1.5.1.1 Localisation à l'aveugle……..…..15

1.5.1.2 Localisation via des index……..…...16
1.5.2 La recherche dans le système P2P structuré……..…...17
1.5.3 La recherche dans le système P2P légèrement structuré.……18
1.6 Conclusion…….……18

2 Quelques protocoles et architectures P2P 19

2.1 Introduction……..19
2.2 P2P structures……...20
2.2.1 CAN ‘‘Content Addressable Network’’……...20
2.2.2 Chord……....21
2.2.3 Pastry……22
2.2.4 Tapestry…….....23
2.3 P2P non structurés……24
2.3.1 Le protocole FastTrack……..…….…24
2.3.2 Le protocole eMule/eDonkey…..……..…….....26
2.3.2.1 Fonctionnement général…….....26
2.3.2.2 Les serveurs eMule…….…….……..27
2.3.2.3 Les clients eMule……..……27
2.3.2.4 La connexion client / serveur……..….28
2.3.2.5 La connexion client / client…….....30
2.3.2.6 La liste de serveurs……..…..31
2.3.3 Le protocole BitTorrent……..……..32
2.3.3.1 Conception et architecture BitTorrent……..…….32
2.3.3.2 Principe de fonctionnement…….....33
2.4 Conclusion…….…..34

3 La sécurité dans les réseaux P2P 36

3.1 Introduction……..36
3.2 Définition de la sécurité…….....37

3.2.1 La sécurité proactive……....37
3.2.2 La sécurité réactive……..38
3.3 Aspects de sécurité dans les réseaux P2P..……...39
3.3.1 Les services de sécurité……....40
3.3.2 Les mécanismes de sécurité……...41
3.4 D'autres fonctions de sécurité…….....43
3.4.1 Chiffrement symétrique……....43
3.4.2 Chiffrement asymétrique……..43
3.4.3 L'audit du système……...44
3.3.3 Fonction de hachage……44
3.4.4 Signature numérique…….....44
3.3.4 Infrastructures de confiance..……45
3.3.5 Autorités de confiance...……...45
3.3.6 Conclusion…….....46

4 Vulnérabilités des systèmes P2P 47
4.1 Introduction……..47
4.2 Attaques Générales et Défenses……...48
4.2.1 L'attaque de dénie de service (DoS)…….....49
4.2.2 L'attaque Man-in-the-Middle (MitM)…….50
4.2.3 L'attaque du Ver (Worm)……52
4.2.4 Le facteur humain…….....54
4.3 Les attaques spécifiques aux réseaux P2P…….54
4.3.1 L'attaque rationnelle……54
4.3.2 L'attaque sybil…….....55
4.3.3 L'attaque Eclipse……57
4.3.4 L'attaque d'empoisonnement……59
4.4 Conclusion……60

5 Un modèle de réalisation de l'attaque Eclipse dans un réseau eDonkey 61

5.1 Introduction…….....61

5.2 Le protocole de communication dans un réseau eDonkey.....62

5.2.1 Description générale de message……62

5.2.2 Communication TCP client / serveur……..63

5.2.3 Communication UDP client / serveur……..64

5.3 Propagation de ver sur réseau P2P…….....64

5.4 Proposition d'un modèle de réalisation de l'attaque Eclipse dans un réseau eDonkey……...66

5.4.1 La description du modèle de réalisation de l'attaque Eclipse dans le réseau eDonkey…….....66

5.4.1.1 Hypothèse……66

5.4.1.2 Fonctionnement de base……67

5.4.1.3 Déroulement de l'attaque et sa propagation……...68

5.5 Conclusion…….....71

Conclusion générale et perspectives……73

Bibliographie…….....75

Annexe A……83

LISTE DES FIGURES

Figure 1.1 Taxonomie des applications P2P……..……...10
Figure 1.2 Le modèle pur…...……...12
Figure 1.3 Le modèle hybride……....12
Figure 1.4 Le modèle centralisé…….....13

Figure 2.1 Exemple de l'espace virtuel de CAN, en 2-dimensions, avant et après l'ajout du nœud Z……...21
Figure 2.2 La topologie FastTrack…..……25
Figure 2.3 Structure générale du réseau eMule…….27
Figure 2.4 Début de dialogue entre le client et le serveur……..…….29
Figure 2.5 Dialogue entre le client et le serveur lors d'une recherche…….…..30
Figure 2.6 Connexion TCP client/client de demande de fichier……...30
Figure 2.7 Un modèle d'une file d'attente de téléchargement d'un peer……..…..31
Figure 2.8 L'architecture BitTorrent et les composants principaux…….…33

Figure 3.1 Mécanismes utilisés par les services de sécurité……38

Figure 4.1 Système de classification de couche……48
Figure 4.2 Une attaque de DDoS……..……..49
Figure 4.3 La tarification est une méthode de limiter la vitesse des demandes……...50
Figure 4.4 L'attaque Man in the Middle……..……...51
Figure 4.5 La propagation du Ver…….……53
Figure 4.6 Exemple d'une attaque sybil contre le réseau Pastry……...56
Figure 4.7 Exemple de l'attaque Eclipse, qui divise le réseau en deux parties……58

Figure 5.1 Propagation de Ver sur le réseau P2P…….65

Figure 5.2 Un exemple de la topologie de réseau eDonkey avec 3 serveurs où (i={attaquant, 1, 2} et j varie de 1à 4)……..67

Figure 5.3 Le serveur attaquant donne une adresse aléatoire de source (client)……..70

Figure 5.4 Séquence UDP de test de présence de serveur attaquant……..70

LISTE DES TABLEAUX

Tableau 5.1 L'en-tête de message eDonkey…… ……62

Tableau 5.2 Un message List-of-servers…..…..…….63

Tableau 5.3 Le message Server-status…...…….64

Tableau 5.4 Message List-of-servers envoyé par le serveur attaquant……..……69

Tableau 5.5 Exemple de message négatif « Search-request » envoyé par le serveur attaquant……..69

Tableau 5.6 Exemple de message négatif « Found-sources » envoyé par le serveur attaquant……...……...69

INTRODUCTION GÉNÉRALE

Au cours de la dernière décennie, le domaine des systèmes pairs-à-pairs (peer-to-peer ou P2P) a connu un succès remarquable. De plus en plus, la largeur de la bande est consommée par ces systèmes et le nombre d'utilisateurs sur ces réseaux augmente. Son principe est de connecter directement deux machines du réseau dès lors qu'elles ont des informations communes qui les intéressent. Désormais, cette connexion permet une relation d'égal à égal entre les deux postes. Il n'est donc plus indispensable de passer par un tiers (serveur central) pour dialoguer avec une machine distante. Les systèmes pair-à-pair sont maintenant une des applications d'Internet les plus populaires et une source très significative de trafic d'Internet.

Le premier réseau pair-à-pair dans l'utilisation répandue était le système de serveur de news *Usnet* auxquels les pairs (ou peers) communiquent entre eux pour propager des articles de news *Usnet* sur le réseau entier *Usnet*. En outre, le *DNS*, les protocoles de routage (*RIP*, *OSPF*) sont des types de vieux modèles P2P. L'émergence du P2P peut être vue comme une renaissance du modèle originel d'Internet au niveau application.

Nous connaissons actuellement surtout le P2P par des logiciels comme Napster, Gnutella, eDonkey et plus récemment Bittorent pour l'échange possible de fichiers de musique ou de film de manière facile. Ceci ne constitue qu'une infime face des protocoles et des utilisations de P2P qui proposent une manière forte et équitable de collaborer en vue d'accroître le potentiel du réseau.

Une définition simple de pair-à-pair est donne dans **[107]**: « *P2P est une classe d'applications qui profitent de stockage de ressources, des cycles, de contenu, de la présence humaine disponible aux bords de l'Internet. Parce qu'ayant accès à ces ressources décentralisées signifie de fonctionner dans un environnement de connectivité instable et des adresses IP imprévisibles, les nœuds pair-à-pairs doivent fonctionner à l'extérieur du DNS et avoir une autonomie significative ou total de serveurs centraux* ». Autrement dit, le P2P est un système spécial distribué sur la couche d'application où chaque paire de nœuds peers peut se communiquer par le protocole de cheminement dans des couches P2P.

Pour comprendre clairement le P2P, nous pouvons comparer le modèle P2P avec le modèle client/serveur (C/S), chaque nœud prend le rôle ou bien de client ou bien de serveur. La ressource comme l'espace de stockage ou la capacité de calcul peut être partagée entre le client et le serveur. Néanmoins, le modèle C/S classique ne parvient pas à tirer la maximum de ces ressources du réseau (souvent mal exploitées) et le serveur dans ce modèle est en fait un point de contrôle central. En revanche, le modèle P2P est fondé sur un concept décentralisé et la nécessité d'un serveur central devient obsolète, car chaque ordinateur que l'on nomme également « peer » fonctionne à la fois comme serveur et comme client (servent).

Avec l'apparition des réseaux complètement décentralisés comme peer-to-peer de nombreux problèmes de sécurité sont inévitablement apparus 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 ayant droits.

Ces difficultés sont à l'origine du développement permanent de peer-to-peer. On assiste aujourd'hui à l'apparition de protocoles « cryptés » et « anonymes ».

Peer-to-Peer offre aux attaquants une façon plus efficace et performante pour accomplir leurs attaques. Mais parallèlement à ces attaques, les défenses et protections se développent.

Les réseaux P2P fournissent un accueil idéal aux différentes attaques (DDoS, Ver, Sybil, Eclipse,…) en faisant de leurs vulnérabilités des proies. Dans ce cadre, notre étude vise à proposer un modèle de réalisation de l'attaque Eclipse dans un réseau hybride eDonkey. Pour cela, nous avons utilisé l'attaque Ver pour monter et propager l'attaque Eclipse et donc parvenir à subdiviser le réseau eDonkey en différents sous réseaux isolés.

Le reste du mémoire est structuré comme suit : Le chapitre 1 est consacré à la présentation des réseaux pair-à-pair et leurs caractéristiques, leur objectifs, leurs avantages et inconvénients, les domaines d'application, les différentes architectures sur lesquelles ils reposent et les mécanismes de découverte spécifique aux différentes structures du système pair-à-pair. Le chapitre 2 présente quelques protocoles et architectures pair-à-pair, pour les systèmes symétriques qui utilisent une technique de répartition de l'information particulière : les tables de hachage distribuée (DHT), on a *CAN, Chord, Pastry* et *Tapastry*. De plus, ce chapitre étudie les protocoles les plus utilisés sur Internet comme Fastrack, eDonkey/eMule et Bittorent. Dans le chapitre 3, nous présentons la sécurité dans les réseaux pair-à-pair et les divers services et mécanismes de sécurité déjà proposés pour sécuriser ces réseaux pair-à-pair. Une vue générale sur les différentes attaques communes et leurs contres mesures dans les réseaux pair-à-pair actuels est détaillée au chapitre 4. Nous avons délimité les points essentiels qui vont nous aider, par la suite, à proposer un modèle de réalisation de l'attaque Eclipse dans un réseau eDonkey dans le chapitre 5. Nous terminons notre travail de recherche par une conclusion et quelques perspectives, dans le dernier chapitre.

Chapitre 1

Les réseaux pairs à pairs

1.1 Introduction

Le pair à pair se limite souvent dans les esprits à un type d'échange de fichiers aux frontières de la légalité. La forte médiatisation de téléchargement illicite de musique ou vidéo illustre à elle seule une forte tendance actuelle à confondre pair à pair avec piratage, c'est-à-dire, confondre une technologie neutre par définition avec l'usage qui peut en être fait. A travers les attaques subies par les technologies P2P, c'est non seulement toute une technologie que l'on cherche à museler, mais ce sont aussi des dizaines de solutions innovantes offertes par le P2P que l'on menace, alors que ces derniers pourraient bien constituer une des plus grandes mutations à venir de l'Internet et de son usage. Son architecture originale a bouleversé les schémas centralisés classiques qui dominaient sur

Internet et contribue de plus en plus à modifier les comportements et habitudes des internautes.

1.2 Généralité sur le pair-à-pair

« Peer to Peer » signifie littéralement « pair à pair ». Ce concept introduit ainsi une relation d'égal à égal entre deux ordinateurs. Les réseaux pairs à pairs se greffent au réseau public qu'est l'Internet. Ils présentent donc une topologie virtuelle (overlay network). Cette partie traite l'aspect technique du P2P, il s'agit tout d'abord de définir précisément la technologie P2P, de mettre en lumière l'historique de ce que certains auteurs définissent comme "une révolution", les caractéristiques, les objectifs, les différences entre le P2P et l'architecture client/serveur dite "traditionnelle" qui domine actuellement Internet, quelques avantages et inconvénients et en fin, les types d'applications P2P (*applications parallèles, application de gestion des fichiers et contenus et application de collaboration*).

1.2.1 Définition

Les réseaux pair-à-pair (P2P) sont très largement traités dans la littérature **[1]**, ainsi plusieurs définitions ont été soumises par la communauté P2P **[2] [3] [4] [5] [6]**. Dès lors, un réseau pair-à-pair, d'égal à égal ou encore de point à point est un réseau dans lequel tous les éléments connectés jouent à peu près le même rôle. Les nœuds pairs peuvent être des clients, des serveurs ou même les deux à la fois.

Avec le pair-à-pair, on sort de l'architecture centrale client/serveur, où le client passif se voit fournir des contenus par des portails centralisés pour entrer dans une sphère où l'utilisateur est actif en mutualisant les ressources dont-il dispose. Ces ressources ne sont donc pas seulement les données mais aussi la puissance de calcul de l'ordinateur, sa bande passante, ses capacités de stockage et de mémoire. La force de ce système est donc de se nourrir de la puissance qu'il crée lui-même.

1.2.2 Genèse

Ce que la plupart d'entre nous ignorent c'est que l'ancêtre d'Internet plus connu sous le nom d'*ARPAnet*, a été conçu initialement comme un système P2P permettant le partage de ressources informatiques à travers les Etats-Unis. La plupart des sites ARPAnet étaient

donc connectés non comme de clients/serveur mais d'égal à égal **[3]**. L'émergence du pair-à-pair peut être vue comme une renaissance du modèle original d'Internet au niveau application.

En 1979, Usenet permettait aux personnes qui se connectent à un seul serveur de news d'avoir accès à tous les serveurs connectés ensemble. C'est une collection de notes ou de messages utilisateurs soumise à divers sujets, qui sont signalés aux serveurs sur un réseau. Chaque collection de notes soumise est connue sous le nom de newsgroup.

En 1984, fut créé un réseau décentralisé appelé *Fidonet*, permettant l'échange de messages entre les utilisateurs à travers « Bulletin Board System (*BBS*)». Il ressemble à Usenet, et les BBS sont considérés comme des peers.

Le premier usage moderne de peer-to-peer fut révélé avec le lancement du logiciel de messagerie appelé ICQ[1] qui permettait à ses utilisateurs d'échanger des messages directement sans passer par un serveur central. Aujourd'hui, ce type d'application est introduit dans le système d'exploitation Microsoft Windows XP avec MSN messenger. C'est l'application P2P la plus utilisée dans le monde.

En 1999, l'apparition de *Napster* et son logiciel de partage de musiques a rendu le P2P très populaire. Un téléchargement 24h/24h et 7j/7j conduit à une poursuite judiciaire à cause au téléchargement illicite, d'où l'arrêt net du Napster en 2002.

Par la suite de nombreuses applications apparaissent, relatives à divers domaines : communication, collaboration,…etc. **[7] [2]**, parmi lesquelles *Gnutella* (version 0.4) **[8] [9],** qui est une approche plus décentralisée pour rendre plus difficile le suivi des traces.

[1] **ICQ** est un système propriétaire de messagerie instantanée et de visioconférence de la société Mirabilis. Le nom ICQ vient de « *I Seek You* » qui signifie« je te cherche ».

1.2.3 Caractéristiques des systèmes P2P

Comme les ressources ont une connectivité instable ou des adresses IPs variables, elles fonctionnent d'une manière autonome, indépendamment de systèmes centraux (e.g, DNS). Un vrai système P2P se reconnaît en répondant aux questions suivantes :

- Est-ce que le système permet à chaque pair de se connecter de manière intermittente avec des adresses IPs variables ?
- Est-ce que le système donne à chaque pair une autonomie significative ?
- Qui possède les ressources qui font tourner le système ?

Un système P2P se caractérise par plusieurs aspects :

- *Décentralisation* ; l'absence du contrôle centralisé, ni aucune vue globale de tous les pairs dans le réseau.
- *Dynamisme* ; l'aptitude de supporter le dynamisme des pairs, c'est-à-dire la mobilité des pairs ne doit pas contrarier le fonctionnement du système.
- *Scalabilité* ; la possibilité d'augmenter le nombre d'utilisateurs et l'aptitude de gérer un tel nombre d'utilisateurs.
- *Tolérance aux fautes* ; la capacité de maintenir la connexion (offre de services) malgré la présence des fautes ainsi que l'intermittence des nœuds.
- *Sécurité* ; l'aptitude d'un système à gérer et à protéger des informations sensibles et vulnérables.
- *Anonymat* ; la possibilité de permettre une résistance aux censures et la liberté de discuter sans crainte de persécution.
- *Connectivité ad hoc* ; la tolérance d'établir une communication sans aucune architecture préexistante, car les systèmes P2P cherchent à faire communiquer les différents nœuds dans les différents types de réseaux (réseaux filaire, mobile, ad hoc)
- *Robustesse* ; l'aptitude à garder la stabilité du système malgré la présence de défaillance (lorsqu'un pair défaillie ou quitte le système).
- *Un réseau virtuel* ; les pairs participant à un service P2P forment souvent un réseau virtuel appelé *overlay* **[10]** conçu au dessus de la couche transport. Généralement, un tel réseau virtuel présente une propriété de transparence qui s'applique à plusieurs points **:**

  (1) tout d'abord, il permet de faire abstraction des différences de nature des pairs dont les caractéristiques sont différentes sur les plans :

- matériel, avec par exemple des stations de travail, des téléphones mobiles, des assistants personnels.
- logiciel, au niveau des systèmes exploitation et langage de programmation.
- de la communication avec l'utilisation des technologies et piles protocolaires différentes.

(2) La transparence s'applique aussi sur le routage effectué au niveau sous-jacent : deux voisins de la topologie virtuelle ne le sont pas forcément physiquement. Enfin, concernant les ressources, un overlay rend transparent leur accès, leur localisation et leur duplication.

1.2.4 Objectifs de système P2P

Le modèle P2P étant très général, ses objectifs sont variés **[11]** :

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

1.2.5 Comparaison entre P2P et client/serveur

Conceptuellement P2P est une alternative au modèle client/serveur **[12]**, qui est constitué d'un serveur ou d'un cluster de serveurs et beaucoup de clients. Dans le modèle

pair-à-pair pur, tous les participants sont des servents[2]. Il y a une différence entre les deux modèles.

Pair-à-Pair :

- Auto-organisé
- Evolution dynamique
- Découverte des pairs
- Flux distribué
- Symétrie du réseau
- Adressage dynamique au niveau application
- Nœuds autonomes
- Attaques difficiles (mobilité, anonymat)

Client/Serveur :

- Administration centralisée
- Configuration statique
- Consultation de tables
- Flux centralisé
- Asymétrie du réseau
- Adressage statique (IP)
- Nœuds dépendants
- Attaques plus simples

1.2.6 Avantages et inconvénients du modèle P2P

Il s'avère assez difficile de définir les avantages et les inconvénients de l'architecture pair-à-pair. Tout dépend en fait du contexte dans lequel la technologie est exploitée.

Avantages du P2P :

- Echanges plus rapides car plus directs,
- Optimisation de l'utilisation de la bande passante du réseau (équilibrage de la charge),
- Maintenance et coût réduit (ressources réparties),
- Résistance aux pannes, grâce à la réplication des ressources,
- Extensibilité (passage à l'échelle),
- Utilisation des ressources non utilisées (CPU, Stockage…).

Inconvénients du P2P :

- La QoS (qualité de service) n'est pas garantie (ligne peu fiable, débit peu élevé, …)

[2] Servent : c'est un nœud qui joue les deux rôles **serv**eur et cli**ent**. D'où ce mot composé.

- La sécurité, c'est un vrai problème pour les réseaux P2P (crackers, virus, DDOS, confidentialité, authentification, …),
- Possibilité d'avoir des contenus fallacieux (consistance, contradiction),
- Difficile d'appliquer la loi *wild wild web* (droit d'auteur, contenu immoral, pornographie infantile).

1.2.7 Domaines d'application

Pour le grand public, le terme P2P peut parfois se confondre avec les systèmes d'échange de fichiers musicaux. En réalité, ce terme englobe également d'autres applications. Trois principales classes d'applications P2P ont émergé **[13]** : *applications parallèles, application de gestion des fichiers et contenus et application de collaboration*, (figure 1.1).

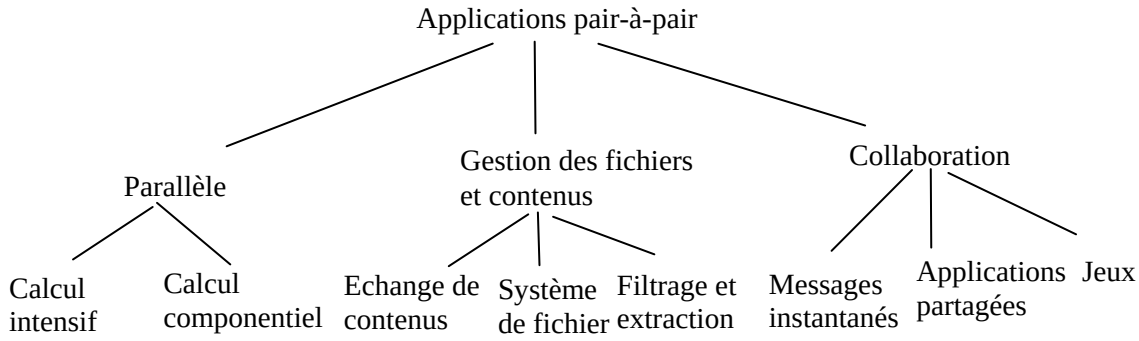


Figure 1.1 : *Taxonomie des applications P2P.*

- *Calcul parallèle* : la possibilité de pouvoir utiliser la puissance de calcul de millions de machines connectées à l'Internet. Il intéresse fortement les acteurs de la communauté du calcul parallèle. Les applications parallèles P2P divisent les gros calculs intensifs en petits calculs plus au moins indépendants et repartis sur plusieurs pairs. Divers projets sont mis en œuvre de grands calculs grâce au P2P. On peut citer par exemple : *Seti@home* **[14]** qui consiste à analyser des signaux extraterrestres, *XtremWeb* **[15]** qui permet le calcul scientifique à travers le Web.
- *Gestion des fichiers et de contenus* : c'est l'application la plus connue de P2P. Elle consiste à créer des communautés de pairs partageant des fichiers stockés sur leurs machines comme par exemple, *Napster* **[16]**, *Gnutella* **[17]** et *Morpheus*, *Kazaa*,

eMule, *Bittorrent* **[18]**. Ils permettent aux pairs de chercher et télécharger des fichiers que d'autres pairs ont rendus disponibles. La plupart des applications actuelles ne prévoient pas la fiabilité et comptent sur l'utilisateur pour faire des choix intelligents concernant la localisation des fichiers recherchés et réessayer quand les téléchargements échouent. Ces applications se focalisent aussi sur l'utilisation des espaces de stockage inutilisés comme un serveur pour des utilisateurs. Elles doivent assurer la fiabilité en employant les techniques traditionnelles des BDD, comme la réplication par exemple. Nous en citons à titre d'exemple : *Freenet* **[19]**, *OCeanStore*, *CFS*.

- *Collaboration* : les applications P2P collaboratives permettent aux utilisateurs de collaborer en temps réel sans compter sur un serveur central pour rassembler et retransmettre l'information. L'email et les News sont les exemples les plus classiques. Ce domaine propose différents services de : (1) communication ; dont on trouve la messagerie instantanée comme : *ICQ*, *AIM*, *AOL Instant Messenger*, *MSN Messenger*, *Yahoo Messenger*, *Skype*…etc, (2) applications partagées qui permettent aux particuliers ou entreprises d'agir réciproquement (l'observation et l'édition de la même information se fait simultanément), et (3) les jeux P2P qui sont un autre exemple d'application P2P faite en collaboration n'exigeant aucun serveur central.

1.3 Les différents niveaux de décentralisation des réseaux P2P

Le P2P se classe dans la branche des systèmes distribués de la classification des systèmes informatiques **[13]**. Néanmoins, le degré de décentralisation est variable et représente une distribution plus au moins forte des tâches accomplies. Les différentes propositions de classification des infrastructures P2P en fonction de leur degré de décentralisation scindent le modèle P2P en deux **[13] [3]** ou trois **[21] [5]**.

1.3.1 Modèle pur

C'est un système totalement décentralisé dont les nœuds peers communiquent de manière symétrique et possèdent le même rôle. Il se caractérise par le fait que la suppression d'un pair n'affecte pas le service offert **[3]**.

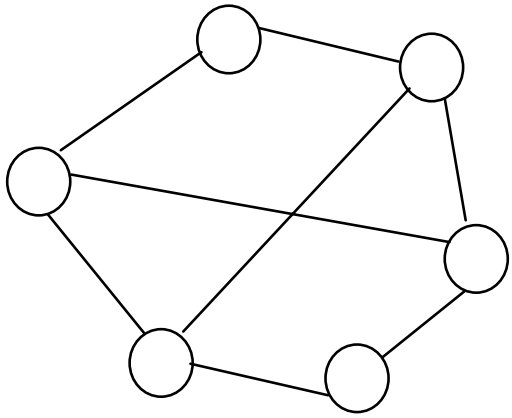


Figure 1.2 : *Le modèle pur*.

1.3.2 Modèle hybride

C'est un modèle P2P décentralisé dont lequel certains pairs jouent un rôle particulier. Ils exécutent des fonctions différentes des autres pairs rendant ainsi les pairs non équivalents en apportant un certain degré de centralisation. Ces nœuds particuliers sont généralement appelés *super-pairs* ou *ultra-pairs* dont le rôle est l'organisation fonctionnelle des pairs. Par exemple, *Kazaa*, les utilise pour la découverte et la localisation des ressources.

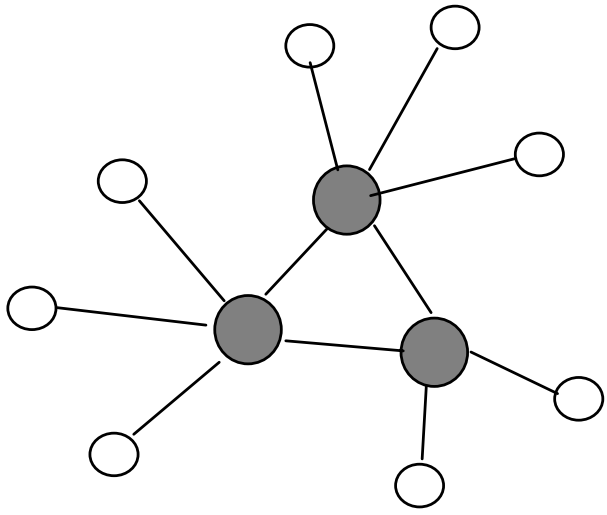


Figure 1.3 : *Le modèle hybride*.

1.3.3 Modèle centralisé

C'est un modèle P2P distribué dans lequel un serveur dédié est utilisé pour assurer les fonctions de découverte de localisation de ressources. Une fois ces fonctions effectuées, le pair interagit directement avec les autres pairs. Ainsi, cette interaction fait la différence entre le modèle P2P centralisé et le client/serveur. Néanmoins, certaines classifications considèrent le modèle centralisé étant comme un cas particulier du modèle hybride contenant un seul super-pair et d'autres le voient comme la composition des modèles purs et centralisés, en ce sens que, chaque super-pair agit comme une entité centrale pour les pairs qui lui sont connectés et l'organisation des super-pair suit le modèle pur.

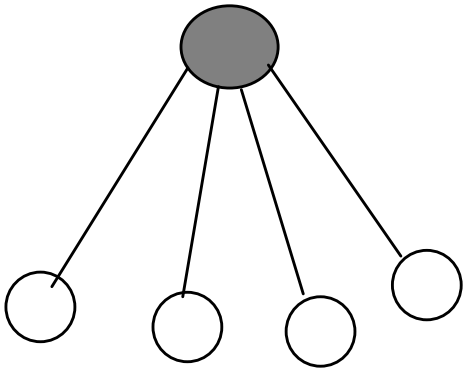


Figure 1.4 : *le modèle centralisé.*

1.4 La structure d'un système pair à pair

Les systèmes pairs à pairs constituent un réseau de pairs fortement dynamique avec une topologie complexe. Cette topologie crée un réseau virtuel au-dessus du réseau physique. Les systèmes P2P peuvent être différenciés par le degré auquel ces réseaux virtuels contiennent une certaine structure, ou sont crées de manière ad hoc. Ici, la structure se réfère à la manière de localiser le contenu du réseau en respectant la topologie du réseau **[22]**. Il y a principalement trois différentes architectures pour les systèmes P2P.

1.4.1 Architecture non structurée

Les architectures non structurées ont été développées en respectant complètement le paradigme pair à pair. Dans ces réseaux, l'emplacement des données ou fichier est complètement indépendant de la topologie virtuelle, puisqu'il n'y a aucune information de

savoir quels sont les nœuds susceptibles d'avoir les données appropriées, donc la recherche se fait d'une manière aléatoire en traitant et demandant différents pairs s'ils ont toutes les données qui conviennent à la requête demandée. Néanmoins, il est difficile de trouver les fichiers désirés sans une large inondation de requêtes. Le temps de recherche n'est pas garanti. Pour cela, ils sont considérés comme des systèmes non scalables **[22]**.

1.4.2 Architecture structurée

Dans les systèmes P2P structurés comme *Chord* **[23]**, *Pastry* **[24]**, *Tapstry* **[25]** et *CAN* **[26]**, la topologie du réseau virtuel est hermétiquement contrôlée et les fichiers ou leurs pointeurs sont placés de façon précise dans des emplacements spécifiques. Cette architecture améliore le processus de localisation des données, en garantissant la localisation d'une ressource disponible sur le réseau. Le principe de cette approche repose sur l'utilisation d'une table de hachage distribuée (DHT). Ceci repose sur le principe classique de hachage en fournissant les services d'insertion d'un couple clé/donnée et de récupération de la donnée associée à une clé.

1.4.3 Architecture légèrement structurée

Dans P2P légèrement structuré, la structure overlay n'est pas strictement spécifiée. Elle est formée en se basant soit sur des allusions, soit sur des probabilités. Dans *Freenet* **[19]** et *Symphony* **[27]**, leur réseau overlay est structuré et la localisation des données n'est pas précisément déterminée.

1.5 Les mécanismes de découvertes dans les systèmes P2P

Les réseaux pairs à pairs sont des réseaux overlay, distribués, dynamiques sans aucune organisation hiérarchique. La pertinence d'un système P2P réside en sa capacité à localiser les ressources de façon efficace quelle que soit la taille du réseau. Ainsi, ces systèmes doivent reposer sur des méthodes efficaces de découverte de ressources désirées.

Différents articles **[28] [29] [30] [31]** ont étudié les techniques de découvertes dans les réseaux P2P. Les systèmes P2P les plus existants supportent la simple recherche d'objets par la clef ou l'identificateur. Ainsi les pairs ne sont donc plus représentés par leurs adresses physiques (de niveau transport) mais par un identifiant défini dans le cadre

de l'overlay. Les techniques de localisation sont généralement classées selon les différentes structures du système pair à pair.

1.5.1 La recherche dans le système P2P non structuré

Dans les systèmes P2P non structurés, il n'existe aucune règle définissant ni où se trouve la donnée stockée, ni l'ordre des nœuds les uns par rapport aux autres. Deux stratégies ont été définies pour la recherche dans un système P2P non structuré : des approches dites à l'aveugle (en anglais *Blinder search*) et des approches basées sur des index (en anglais *informed search*). Une taxonomie des différentes techniques existantes est présentée dans **[30]**. Nous présentons ci-dessous les stratégies de localisation les plus utilisées **[32]**.

1.5.1.1 Localisation à l'aveugle

Consiste à propager la requête à un nombre suffisant de nœuds pour satisfaire la demande, ce principe est représenté de différentes manières :

- *Inondation simple (flooding)*, cette approche est basée sur un simple envoi des messages initiés ou reçus à tous les voisins spécifiés dans la table de voisinage du pair considéré. Chaque message initial est défini avec un TTL[3] qui sera décrémenté au fur et à mesure. Dès que le TTL du message arrive à zéro, la propagation de ce dernier est alors stoppée. La version originale du *Gnutella* utilise cette technique d'inondation simple (appelé : BFS, *Breadth First Search*). Pour trouver les fichiers dans le réseau, une étude **[33]** de cette approche a montré que cette stratégie n'est pas scalable, du fait de la charge induite sur le réseau par le grand nombre de messages échangés. Différentes politiques d'inondation ont été mises en œuvre pour améliorer la recherche dans les réseaux P2P **[34] [32]**.
- *Inondation graduelle (Gradual flooding)*, comme son nom l'indique, cette approche est aussi basée sur le principe d'inondation, mais dans ce cas, le TTL des messages sont gérés différemment. Le principe est d'attribuer aux messages un TTL initial de petite valeur. Si la recherche est infructueuse, alors le pair augmente la valeur du TTL et recommence une inondation avec ce nouveau TTL. Selon **[35]** l'inondation

[3] TTL : Time To Leave.

graduelle ou *expanding ring* **[36]** s'avère plus efficace dans le cas de forte réplication des données.

- *Marche aléatoire (Random walk),* le principe d'une marche aléatoire **[37]**, consiste à transmettre les messages à un ou plusieurs voisins choisis aléatoirement à chaque étape, jusqu'à ce que les données soient localisées. La charge du réseau est clairement réduite mais les délais de retour augmentent de manière significative. En augmentant le nombre de marcheurs[4] l'augmentation des délais peut être contrôlée. Multiples marches aléatoires exigent un mécanisme pour terminer la marche. Deux méthodes se présentent : le *TTL* et la *vérification*. Cet algorithme de localisation supporte le passage à l'échelle.

1.5.1.2 Localisation via des index

Aussi appelée recherche informée, où la recherche se fait en utilisant des informations sur la localisation des documents :

- *Routing Indice (RI),* l'ajout d'information sémantique permet aussi d'améliorer les performances de localisation des ressources. L'approche par RI **[38]** se base sur l'étiquetage des ressources par des thématiques (domaines scientifique, musical,…). Cette vision thématique des pairs permet d'enrichir sémantiquement les tables de routage sur chaque pair. Ainsi, chaque pair dispose du nombre de ressources accessibles à chacun de ses voisins. L'information stockée est agrégée et groupée par thème. Les requêtes considérées par RI sont basées sur le contenu du fichier (au lieu du nom ou l'identificateur du fichier). Une requête portant sur un thème donné sera donc transmise uniquement sur les voisins permettant l'accès aux ressources du thème. Le but de RI est de faciliter au nœud la sélection des meilleurs voisins pour faire suivre les requêtes. En général, un bon voisin est celui par lequel beaucoup de documents peuvent être rapidement trouvés.
- *Raccourcis basés sur l'Internet*, c'est un principe encore puissant qui permet à résoudre le problème de passage à l'échelle du processus d'inondation par l'exploitation de l'intérêt des utilisateurs pour localiser les données **[39]**. Si un pair a une partie particulière du contenu par laquelle on est intéressé, il est très possible qu'il y ait d'autres articles qui nous intéressent aussi. En partant du postulat qu'un

[4] Une marche est un message envoyé par un nœud demandeur.

pair retournant des données pertinentes est considéré comme pertinent, l'algorithme consiste à choisir aléatoirement un pair parmi ces pairs pour ensuite le rajouter dans la liste de raccourcis[5]. Les raccourcis sont triés en fonction de leur utilité selon le principe qu'un lien fréquemment utilisé sera conservé par rapport à un lien qui ne sera pas utilisé. La construction des raccourcis se fait par apprentissage basé sur les requêtes passées. Un message initié par un pair est partagé dans un premier temps le long des raccourcis et si aucune donnée n'est trouvée, alors le message est propagé selon le principe d'inondation. Pour améliorer les performances, les messages passant par les raccourcis effectuent un second rebond à travers les raccourcis des pairs sur lesquels ils ont été envoyés. De cette façon, la probabilité de localiser une donnée est augmentée.

1.5.2 La recherche dans le système P2P structuré

Dans un système strictement structuré, le rapport voisin entre les pairs et les emplacements de données est strictement défini. La recherche dans de tels systèmes est donc déterminée par l'architecture particulière de réseau. Parmi ces systèmes structurés, certains implémentent une table de hachage distribué (DHT) en utilisant différentes structures de données : *insert(hash,data)* qui insère une donnée dans la DHT et *lookup(hash)* qui retourne la donnée stockée en *hash*. Quelques systèmes DHT pair à pair ont des structures virtuelles plates (*flat overlay DHT*), d'autres ont des structures virtuelles hiérarchiques (*hierarchical overlay DHT*) **[29]**. DHT consiste à attribuer des identifiants aux données dans le même espace que celui de l'identifiant du pair. Ceci est numériquement plus proche de celui de l'identifiant de la donnée est responsable. Cela signifie qu'il connaît l'emplacement de cette donnée. Ainsi, sa localisation revient à acheminer une requête vers le pair dont l'identifiant est le plus proche. Le DHT, au sein de lequel sont répartis les pairs permet de le faire de manière efficace. Généralement, en *log(n)* sauts logiques dans un réseau constitué de *n* pairs. La localisation dans ces systèmes est certaine. Les systèmes les plus connus sont Chord **[23]**, PGrid **[40]**, CAN **[26]**. Avec Chord, le réseau logique est représenté sur un *anneau*. PGrid exploite une représentation en *arbre* et CAN utilise un espace *D-dimensionnel*.

[5] Pour la raison du passage à l'échelle la liste est d'une taille fixe.

1.5.3 La recherche dans le système P2P légèrement structuré

La recherche dans les systèmes P2P légèrement structurés dépend de la structure virtuelle du réseau et la manière dont les données sont stockées. Par exemple dans Freenet, les données sont stockées sur la base des allusions, employées pour la construction du système virtuel. Donc la recherche dans ce réseau est aussi basée sur des allusions. Il existe d'autres techniques de recherche dans un tel système. Ils sont présentés dans **[29]**.

1.6 Conclusion

Cette présentation générale du pair à pair avait pour but de montrer les différents aspects de cette technologie. Les systèmes pairs à pairs, constituent une alternative intéressante et importante au système traditionnel. Le P2P connaît toujours un vrai succès auprès du grand public grâce aux logiciels de partage et de communication. Néanmoins, le pair à pair se développe également auprès de différentes applications comme le calcul distribué et le travail collaboratif…etc.

Différents protocoles et architectures se sont succédés à cause de l'évolution technique (amélioration du débit et des logiciels), juridique (la fermeture de Napster a poussé à la décentralisation des réseaux) et sociologique (changement des comportements des utilisateurs et utilisation consciente et massive). Ainsi, on a vu quelques mécanismes de découverte spécifiques aux différentes structures du système P2P.

Chapitre **2**

Quelques protocoles et architectures P2P

2.1 Introduction

Pour qu'un réseau pair à pair existe et fonctionne, il faut tout d'abord que chaque participant (internaute) installe sur son ordinateur un logiciel qui lui permet de se connecter au réseau. Il faut aussi, qu'il soit doté de règles de fonctionnement communes à tous : c'est ce qu'on appelle *le protocole* du réseau. C'est lui qui détermine comment et selon quel format les ordinateurs s'échangent des informations. Par exemple, *qui possède tel ou tel fichier ?comment initier un transfert de données ?* Les protocoles identifient l'ordinateur qui propose le fichier et contrôlent le transfert des données. Selon les différentes classes d'architectures du réseau P2P, on trouve par exemple pour les réseaux virtuels P2P structurés : CAN, Tapestry, Chord, Pastry, Kademlia et Viceroy **[41] [42]**. En outre, pour

les réseaux virtuels P2P non structuré : Freenet, Gnutella, FastTrack **[43]**, Kazaa **[44]**, Overnet **[45]**, eDonkey **[46]**, BitTorrent **[47]**.

Eng Keong et al **[48]** ont présenté une expertise et une comparaison de divers réseaux P2P structuré et non structuré. Cependant, les systèmes décentralisés non structurés sont généralement les plus utilisés sur Internet. Il ne s'agit pas ici de présenter une liste exhaustive des réseaux P2P accessible via l'Internet mais plutôt de montrer les plus importantes, en particulier, Fastrack, eMule/eDonkey et BitTorrent qui sont des P2P non structurés. Mais tout d'abord, on présente quelques exemples de DHT qui sont des P2P structurés (protocoles de deuxième génération) et qui ont des topologies « connues ».

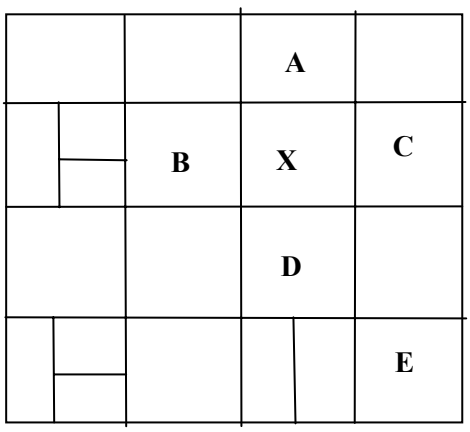
2.2 P2P structurés

Les quatre premières tables de hachage distribuées sont apparues au même moment en 2001 sous l'impulsion de différents groupes de recherche : Chord, Pastry, CAN et Tapestry. Par la suite les recherches à ce sujet n'ont pas cessé, et d'autres DHT sont apparues comme par exemple Kademlia. Ces réseaux implémentent une DHT. Cette dernière représente un espace virtuel, l'espace des clefs *keyIds*. Chaque nœud (PC) est repéré par un identifiant unique $nodeId \in keyIds$, de la même façon, chaque ressource (fichier, abonnement, …) est identifiée de manière unique par un identifiant de clé *key*, tel que $key \in keyIds$ (dans le cas d'un fichier, la clé est souvent l'empreinte SHA1 de ce fichier). Pour accéder à une ressource, un nœud doit connaître la clé de cette ressource. Ce nœud utilise alors la DHT qui implémente de façon distribuée une fonction liant la clé à la ressources. Dans ces protocoles, les pairs s'organisent dans une structure bien définie (topologie connue comme anneau, hyper-cube et tore) qui est employée pour le cheminement des requêtes.

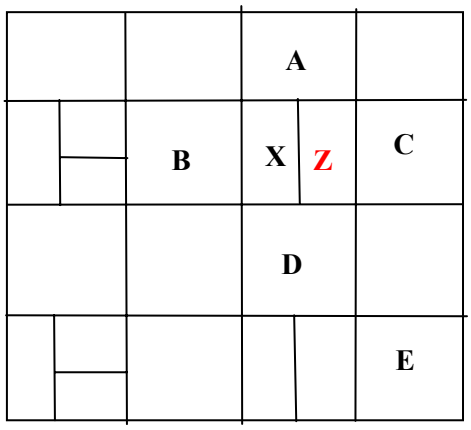
2.2.1 CAN ''*Content Addressable Network*''

Can est une infrastructure P2P décentralisée et distribuée qui procure des algorithmes de routage pair à pair implémentant une table de hachage distribuée (DHT). Il est conçu pour être scalable, tolérant aux fautes et auto-organisé.

Can repose sur la topologie en tore. Les clés sont associées à des cordonnées dans un espace de *d* dimension. L'espace à *d* dimensions est découpé en zone occupée par un noeud. Des objets sont hébergés par les noeuds sous forme d'ensemble clé K /valeur V. La valeur correspond à l'identité (l'adresse IP le plus souvent) du noeud *n* possédant la ressource représentée par la clé (qui est obtenue grâce à un algorithme de cryptage) et chaque nœud est responsable d'une zone de clés. Les nœuds peers couvrent en permanence l'ensemble de l'espace de clés. Ainsi, lorsqu'un nouveau noeud entre, les zones sont partagées, comme montre la figure 2.1 **[48]**. Cela permet d'avoir une table de routage de taille fixe. Mais le temps d'acheminement ainsi que le temps de recherche grandissent plus rapidement. Cependant, certains nœuds peuvent être soumis à beaucoup de requêtes. Des mécanismes de réplication peuvent s'ajouter afin de résoudre ces problèmes dus aux surcharges.



Avant : Le voisinage du nœud X = {A, B, C, D}



A*près* : Le voisinage du nœud X = {A, B, D, Z} et le voisinage du nouveau nœud Z = {A, C, D, X}

Figure 2.1 : *Exemple de l'espace virtuel de CAN, en 2 dimensions, avant et après l'ajout du nœud Z.*

2.2.2 Chord

Chord est utilisé dans différents types d'application : CFS (collaborative file system), un système de fichier distribué à l'échelle d'Internet, ConChord qui utilise CFS afin de fournir une infrastructure distribuée pour la délivrance de certificats de sécurité SDSI

(Simple Distributed Security Infrastructure) et enfin dans DDNS (Distributed Domain Name System) qui propose une implémentation P2P du DNS (Domaine Name Service).

Chord repose sur une topologie en anneau, un nœud Chord a la connaissance de son prédécesseur et du nœud suivant. Une fonction de hachage régulière (*consistent hashing*) génère une clé pour chaque nœud à partir de son adresse IP. Ensuite, chaque nœud est placé dans l'anneau de manière à ordonner les clés par ordre croissant.

Chaque ressource d'identifiant *id* est assignée au nœud possédant le premier identifiant (ou clé) supérieur à *id*. L'espace d'Ids est circulaire et unidirectionnel, les nœuds forment une liste simplement chaînée, chacun ne connaissant que son successeur. Les messages sont donc routés le long de l'anneau, tant que l'identifiant du nœud courant est inférieur à l'identifiant recherché. Dés qu'un nœud reçoit une requête pour une ressource d'identifiant inférieur à lui, il sait qu'il est responsable de cette ressource. Ce routage « *naïf* » est long. La *Finger-table* est donc ajoutée, il s'agit de raccourcis dans le routage. Cette table de *160* entrées. Chaque nœud *n* se voit doté d'une entrée vers les nœuds (appelés *fingers*) de clé suivant $n+2^{i-1}$ avec $1 < i < 160$.

Chord à des avantages comme : la simplicité, supporte le passage à l'échelle et répond la plupart du temps correctement aux requêtes même dans une phase de récupération de défaillance (un nœud quitte le réseau) et de stabilisation (lors de l'ajout d'un noeud).

2.2.3 Pastry

Pastry reprend la structure circulaire de Chord à quelques détails près. Les identifiants de nœud sont choisis aléatoirement et codés sur 128 bits, alors que les identifiants de ressources sont codés sur 160 bits. Pastry fournit un système de routage et suggère des contraintes pour un système de transfert et de réplication efficace, assurant la persistance des données présentes dans la DHT.

Les données sont sous la responsabilité du nœud ayant l'identifiant le plus proche numériquement. Le routage s'effectue en propageant le message vers un nœud partageant un plus grand préfixe commun avec la ressource, ou à défaut partageant un préfixe de

même taille et plus proche numériquement de la ressource ciblée. Chaque nœud possède un *leaf set* et une *table de routage*. Le *leaf set* contient *L* voisins dans l'espace logique, typiquement 16 ou 32. Il sert à donner au nœud une vision précise de sa localité, tout en permettant à certains nœuds de défaillir sans briser le réseau. Ce *leaf set* est maintenu à jour de manière active. La table de routage quant à elle est une table par préfixe plus précise pour les nœuds proches dans virtuel. Elle est mise à jour de manière passive, en complétant les trous des nœuds défaillants et en remplaçant les entrées par des nœuds plus proches.

Pour router un message, un nœud *n* vérifie d'abord si la ressource *r* est sous le contrôle de son *leaf set*. Si tel est le cas, *n* transmet directement *r* au nœud responsable. Sinon, *n* cherche dans sa table de routage un nœud partageant un préfixe plus long avec *r* pour lui transmettre le message. Si *n* ne trouve pas de nœud satisfaisant, alors il transmet le message à un nœud partageant un préfixe de même taille que lui avec *r*, mais qui est plus proche numériquement (un tel nœud doit exister dans le *leaf set*).

2.2.4 Tapestry

Tapestry est basé sur l'algorithme de Plaxton. Il a la particularité de gérer la localisation d'objets et les noeuds en même temps **[48]**. Il réparti de manière uniforme les objets et les noeuds sur le réseau en utilisant un algorithme de hachage (SHA-1 par exemple).

Tapestry ressemble beaucoup à Pastry. Néanmoins, l'acheminement ici se base sur le suffixe au lieu de préfixe. Cependant, chaque noeud *n* maintient une table de routage répartie en différents niveaux. Chaque niveau *l* de la table contient une liste de voisins de *n* dont les *l* premiers chiffres sont communs à ceux de *n*. Chaque noeud possède un identifiant écrit sous forme hexadécimale d'une longueur de quatre chiffres. Un noeud d'identifiant 0325 fait parvenir un message à un noeud d'identifiant 4598. Il va alors consulter le premier niveau de sa table de routage et y chercher un voisin dont le dernier chiffre est 8, i.e. $***8$, puis lui faire suivre ce message. A son tour, ce noeud va consulter le second niveau de sa table de routage et chercher un voisin dont l'identifiant se termine par 98, i.e. $**98$, et lui faire suivre le message. Ce processus se répète jusqu'à l'atteinte du noeud considéré.

Tapestry est présenté comme une amélioration de l'algorithme de Plaxton qui est très statique. Tout en gardant son intérêt principal est qui sa méthode de routage. Il possède la capacité de gérer la mobilité des objets. Comme les trois premiers protocoles, Tapestry utilise la fonction de réplication afin de rendre le système plus robuste.

2.3 P2P non structurés

Dans cette section nous allons présenter les protocoles P2P non strucutré les plus couramment utilisées, à savoir le réseau FastTrack, eMule/eDonkey et le protocole BitTorrent.

2.3.1 Le protocole FastTrack

FastTrack est un protocole pair à pair de partage de fichier qui supporte la recherche de métas-données. Il est venu comme une solution aux problèmes que Napster et Gnutella (version 0.4) faisaient face **[49] [50]**.

Il est une propriété appartenant à la société américaine *Sherman Networks*[1] , d'où très peu de détails sont connus du protocole réel utilisé. Ses clients les plus connus du grand public sont *Kazaa*, *Morpheus*, *Grokster* et *iMech*.

L'architecture de FastTrack suit un système 2-tiérs (niveaux) dans lequel le premier niveau se compose de super-pair (SP). Ce sont des pairs avec une haute largeur de bande passante et espace disque. Le deuxième niveau se compose de simples pairs qui ont de faibles capacités (par exemple une faible connexion par modem). Ces pairs sont nommés pairs ordinaires (OP) (figure 2.2).

Un sous-ensemble de nœuds ordinaires se relie à un pair de premier niveau et transmet les métas donnés des fichiers qu'ils partagent aux super-pairs. Chaque super-pair indexe alors les fichiers des pairs ordinaires qui lui sont liés comme le faisait autrefois le serveur central du réseau Napster. En revanche entre les super-pairs, le système continu a fonctionné comme le réseau Gnutella (version 0.4). Le réseau FastTrack est dynamique et

[1] http://www.sharmannetworks.com

toujours changeant. En effet les super-pairs peuvent librement joindre ou quitter le réseau. Afin d'assurer la disponibilité constante du réseau, il est nécessaire de dédier un pair disponible (ou plusieurs pairs) pour la surveillance et le maintient du réseau. Un tel pair s'appelle un nœud d'amorçage, en anglais « *bootstrapping node* » **[51] [49]**.

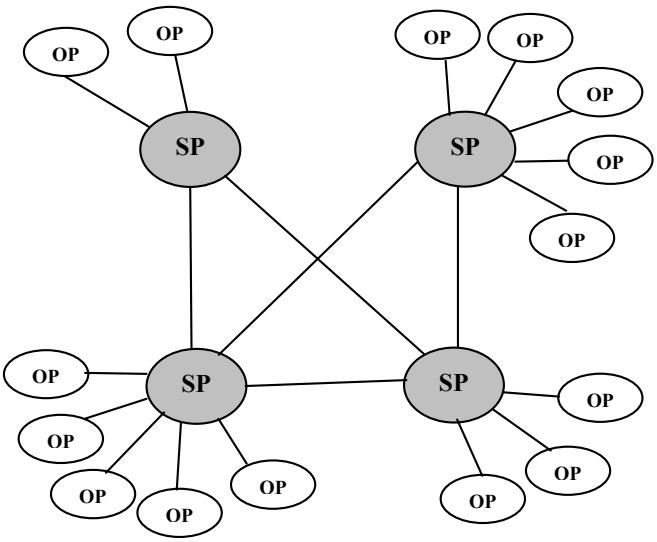


Figure 2.2 *: La topologie FastTrack.*

Quand un client FastTrack, par exemple Kazaa, est exécuté sur un pair, il entrera d'abord en contact avec le nœud d'amorçage. Ce dernier, détermine alors si ce pair particulier est qualifié pour être un super-pair. Si c'est le cas, il l'équipera de quelques adresses IP d'autres super-pairs. Sinon, il est qualifié pour être seulement un pair ordinaire et il lui répondra en fournissant l'adresse IP d'un des super-pairs.

La découverte de ressources est réalisée par la diffusion entre les super-pairs. Quand un nœud de deuxième niveau (OP) fait une demande, elle est orientée d'abord à son propre super-pair (SP) qui diffusera à son tour la même demande à tous les autres super-pairs auxquels il est actuellement relié. Ceci est répété jusqu'à ce que le TTL de cette demande atteigne zéro. Chaque super-pair qui a reçu la requête mène une recherche à travers sa base de données indexée qui contient l'information de tous les fichiers partagés par ses nœuds reliés. Une fois la donnée désirée est trouvée, une réponse sera renvoyée à travers le même chemin où la requête recherchée a été propagée jusqu'à ce quelle atteigne le nœud original qui l'a émis. Cette méthode d'acheminement de réponse ressemble à celle du réseau

Gnutella. Ainsi, il n'est pas garanti de recevoir l'intégralité des résultats car si un des nœuds de la chaîne se déconnecte, les résultats de cette partie ne peuvent arriver. Ce problème se produit moins sur FastTrack. En effet les super-pairs étant censés être des nœuds rapides (connexion à haut débit) les chemins sont normalement plus fiables et plus stables. Le téléchargement sur FastTrack est le même que sur Gnutella. Une fois l'endroit du fichier trouvé, le client qui veut le fichier se relie au client qui accueille ce fichier et envoie une demande http. Le client accueillant le fichier désiré interprète la demande et envoie une réponse http.

Malheureusement, bien que Kazaa a prit dès 2001 la suite de Napster comme référence incontournable pour le grand public (quand on souhaite décrire un moyen gratuit d'échange des fichiers musicaux ou vidéo), les attaques judiciaires contre le téléchargement illégal sur kazaa poussèrent de plus en plus les utilisateurs à le délaisser pour d'autre logiciel tel eMule **[52]**.

2.3.2 Le protocole eMule/eDonkey

Le réseau eDonkey **[46]** (aussi bien connu pour un de ses clients, eMule) était, en octobre 2005, le réseau pair à pair de partage de fichier le plus populaire avec 2.9 millions d'utilisateurs quotidiens **[94]**. Nous décrivons ci-dessous, le protocole eMule **[52]** qui est une application populaire de partage de fichiers basée sur le protocole eDonkey. Le réseau eMule est peuplé par plusieurs centaines de serveurs eMule et des millions de clients.

2.3.2.1 Fonctionnement général

eMule fonctionne dans un mode décentralisé. Il appartient à la classe des architectures P2P hybride et comprend deux applications : le client eMule (ou pair) et le serveur eMule (figure 2.3). Le client eMule partage et télécharge des fichiers. Le serveur eMule fonctionne comme un serveur d'index pour les emplacements de fichier et distribue les adresses d'autres serveurs aux clients.

eMule fournit des particularités avancées, comme la recherche basée sur des méta-données de fichier, les téléchargement simultanés d'un fichier de différentes sources, le partage de téléchargement partiel et la détection de corruption.

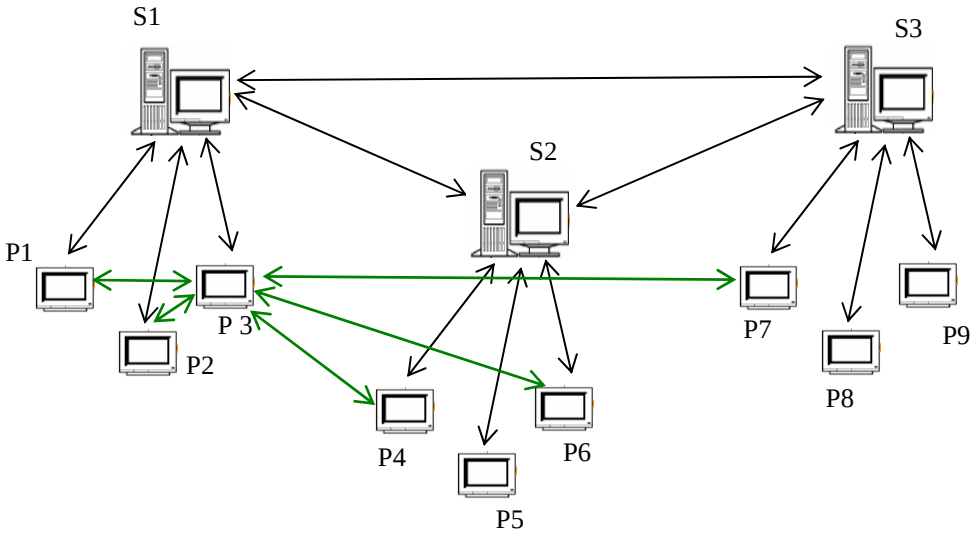


Figure 2.3 : *Structure générale du réseau eMule.*

2.3.2.2 Les serveurs eMule

Les serveurs eMule sont les points d'entrée du réseau. Ils permettent aux pairs de se faire connaître sur le réseau et d'en rencontrer d'autres. Les serveurs utilisent une base de données interne dans laquelle ils stockent l'information sur des clients et des fichiers. Ils agissent comme des index centralisés pour le stockage de l'information sur l'emplacement de fichiers des clients connectés et ainsi de leur proposer un moteur de recherches. Une fonction supplémentaire du serveur qui établit un rapprochement (jouer le rôle d'un pont) entre deux clients qui se relient par un pare-feu et ne peuvent pas accepter les connexions entrantes, cette fonction augmente considérablement la charge du serveur **[52]**. eMule utilise UDP pour augmenter les capacités d'un client.

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 manifeste auprès de plusieurs autres qui vont signaler à leur tour l'arrivée d'un nouveau serveur.

2.3.2.3 Les clients eMule

Le client eMule est sûrement l'élément le plus important dans le réseau car c'est lui qui assure réellement la fonction « égal à égal » **[95]**. Son rôle est de :

- Se connecter à un serveur pour indiquer ce qu'il partage en utilisant une simple connexion TCP. Cependant, chaque client eMule est préconfiguré avec une liste de serveurs et une liste de fichiers partagés sur son système de fichiers local.
- Emettre des requêtes de recherche de fichiers.
- Demander les sources possibles pour un fichier demandé.
- Télécharger les fichiers demandés sur les sources.
- Vérifier l'intégralité et la corruption des fichiers.
- Envoyer les fichiers demandés par les autres clients.
- Gérer une file d'attente des clients.

2.3.2.4 La connexion client / serveur

La connexion client / serveur présente deux objectifs principaux : (1) *cherchant de nouveaux fichiers pour télécharger*. (2) *découvrant de nouvelles sources pour les fichiers qu'ils sont en trains de se télécharger* **[94] [96]**. Tout d'abord, chaque client introduit une connexion TCP avec exactement un seul serveur. Le serveur attribut au client un ID qui sera employé pour l'identifier dans le reste de sa session avec ce serveur. Un *highID* qui correspond à l'adresse IP du client sous forme d'entier. Un *lowID* est un numéro attribué par le serveur pour identifier le client n'ayant pas d'IP publique (ou directement accessible).

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 des fichiers qu'il désire partager et demande ensuite une mise à jour de sa liste de serveurs. Le serveur envoi son statut, sa version de serveur puis sa liste de serveurs connus (figure 2.4).

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. 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. Cette recherche de fichier est initiée par l'utilisateur **[52]**. L'utilisateur peut alors choisir le fichier qui lui convient dans la liste. Le client peut alors

demander au serveur les sources ayant ce fichier. Le serveur va lui retourner la liste des sources correspondant pour pouvoir télécharger (figure 2.5). Les clients emploient aussi des messages UDP pour propager leurs requêtes à d'autres serveurs puisque aucune fonctionnalité de diffusion n'est disponible entre des serveurs. Le but des messages UDP est d'*améliorer la recherche de fichier, la recherche de source* et finalement, *les tests de présence des serveurs* (s'assurer que tous les serveurs eMule dans la liste de serveur du client sont valables) **[52] [94]**.

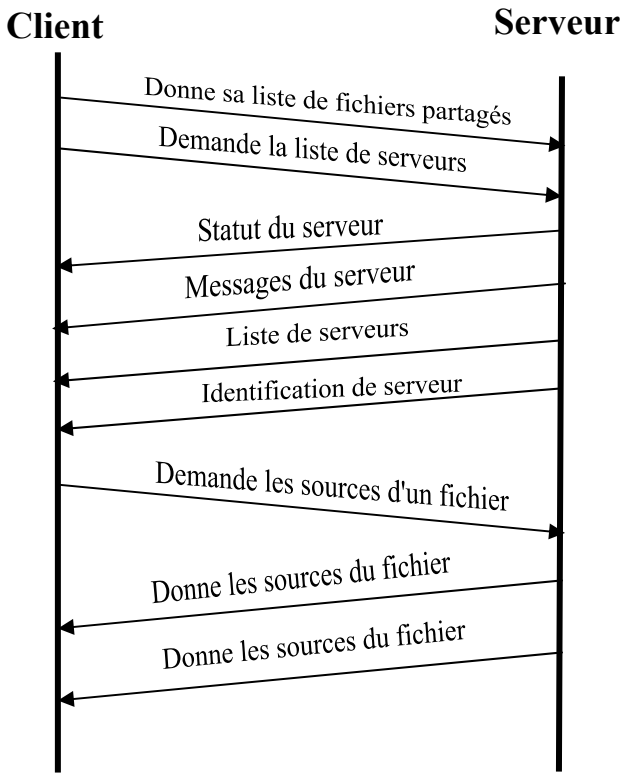


Figure 2.4 : *Début de dialogue entre le client et le serveur.*

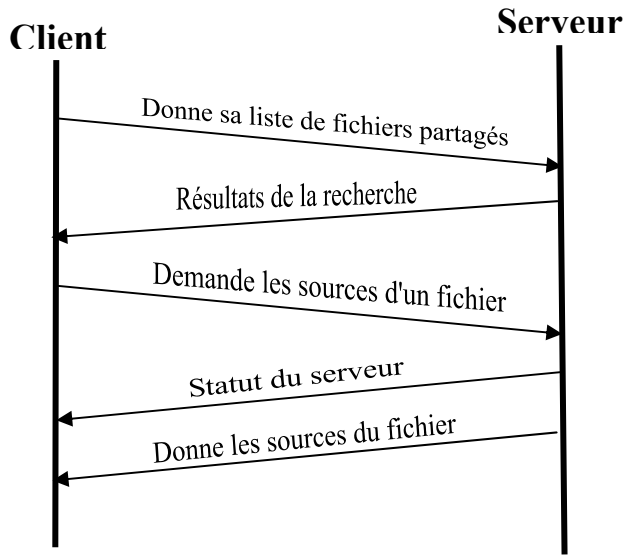


Figure 2.5 : *Dialogue entre le client et le serveur lors d'une recherche.*

2.3.2.5 La connexion client / client

Une fois que le client eMule récupère la liste des pairs (sources) ayant le fichier désiré, le client va essayer de se connecter à ceux-ci (connexion TCP) afin de télécharger le fichier (figure 2.6). Le fichier est divisé en parties de 9.28 Mo, dont est calculé pour chacune d'elles une somme MD4 **[97]**. Et grâce au protocole **MFTP** (*Multisource File Transfer Protocol*) **[96]**, un client peut à la fois télécharger un fichier à partir de plusieurs sources et aussi partager les parties déjà téléchargées.

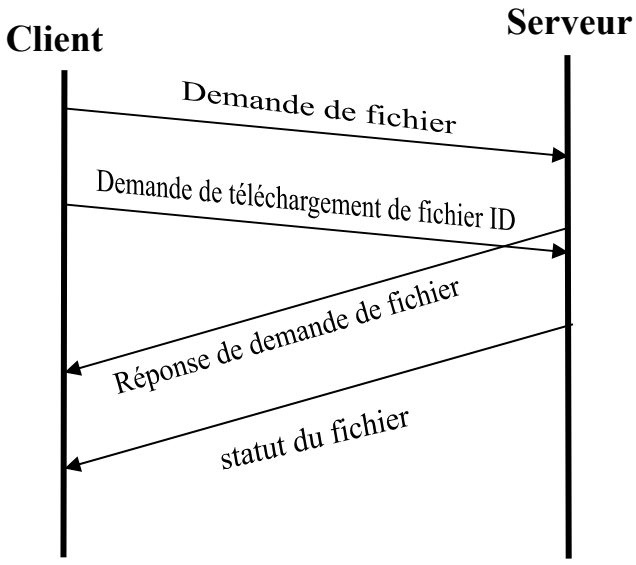


Figure 2.6 : *Connexion TCP client / client de demande de fichier.*

Lorsqu'un client souhaite télécharger une partie du fichier sur un autre client, il est placé dans une file d'attente de téléchargement (figure 2.7). Si la file est vide, alors le client 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 quelle est pleine. Des messages sont émis en UDP périodiquement entre les pairs pour connaître leur position dans la file d'attente.

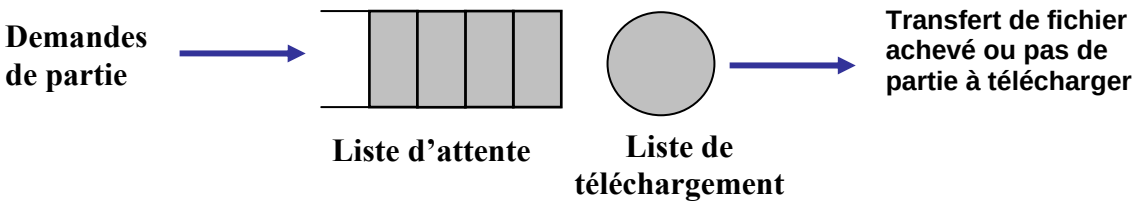


Figure 2.7 : *Un modèle d'une file d'attente de téléchargement d'un peer.*

Un système de crédit est mis en place pour aller plus vite dans la file. Il 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. Pour empêcher la personnification, eMule assure le système de crédit en utilisant la cryptographie à clef publique *RSA*.

2.3.2.6 La liste des serveurs

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 numéro de port) sur lesquels il est possible de se connecter. Cette liste est enregistrée 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 disponibles pour se connecter. Mais le dynamisme du réseau implique un mécanisme de mise à jour de cette liste **[52] [98].**

Lorsqu'un client se connecte, le serveur lui envoie sa liste de serveurs. Le client peut alors mettre à jour sa liste. Il effectue régulièrement des *tests de présence* des serveurs

(requête de *Keep-alive* 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 **[52]**.

2.3.3 Le protocole BitTorrent

En raison de la nécessité d'avoir une bande passante plus grande pour pouvoir échanger et partager des fichiers de plus en plus grands, le réseau BitTorrent **[53] [54]** a été imaginé et crée par *Bram Cohen* en 2001 **[54]**. BitTorrent est une application pair-à-pair qui permet la distribution rapide et efficace de grands fichiers. Ici, nous présentons une vue d'ensemble sommaire. Pour plus de détails **[47] [99]**.

2.3.3.1 Conception et architecture BitTorrent

Ce protocole crée un réseau virtuel de pair en utilisant cinq composants principaux : le *fichier .torrent*, un *site web*, le *serveur tracker*, les clients *seeders*, les clients *leechers* (figure 2.8).

BitTorrent est classé parmi les réseaux pairs à pairs non structurés. Il a une architecture hybride du fait de l'utilisation des *trackers* qui permettent de mettre les pairs en relation. Il ressemble beaucoup au principe utilisé par le protocole eMule/eDonkey. Cependant, les pairs, de ce dernier, peuvent se connecter à n'importe quel serveur auquel son utilisateur partage un fichier qui les intéresse. Quant à BitTorrent, il oblige les utilisateurs à se connecter à un *tracker* précis, désigné dans le fichier *.torrent*.

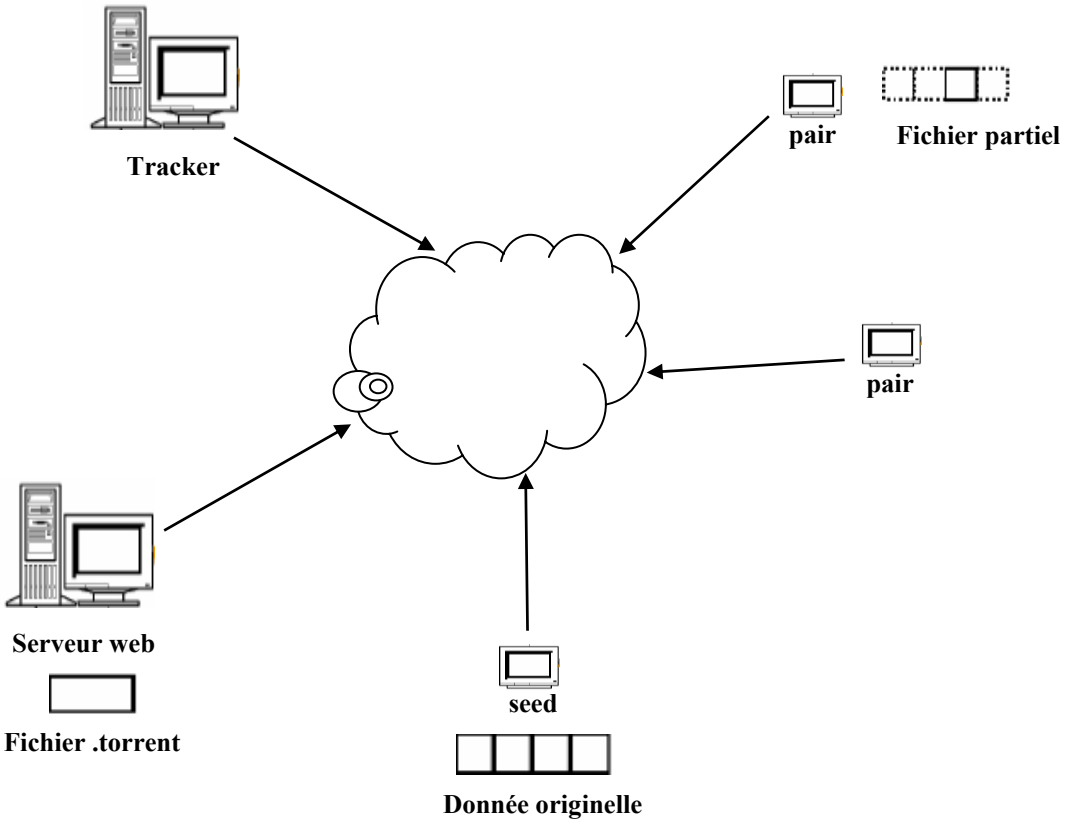


Figure 2.8 : *L'architecture BitTorrent et les composants principaux*.

2.3.3.2 Principe de fonctionnement

Pour partager un fichier, les clients commencent par créer un fichier « *torrent* » qui contient des « *méta donnée* » sur le fichier à partagé : son *nom*, sa *taille*, le *hash SHA1 du fichier* (permettant de l'identifier), *la taille des fragments du fichier* et leurs *hashs* (pour vérifier que l'intégrité des information est bien conservée) et également *l'adresse IP* (ou l'URL) *d'un tracker* qu'il faut interroger pour pouvoir télécharger le fichier.

Lorsque un client BitTorrent veut télécharger un fichier, il cherche d'abord le fichier «.*torrent* »correspondant à travers des sites web, par email ou canal IRC. Contrairement à FastTrack et eMule/eDonkey qui possèdent leur propre moteur de recherche. En suite, le client se sert de l'adresse qui se trouve dans le fichier «.*torrent* » pour contacter le *tracker* responsable du fichier, en envoyant une requête « *GET* » par l'intermédiaire du protocole HTTP **[55]**. Cette requête permet de fournir au *tracker* différentes données sur le client et

sur le fichier recherché. Le *tracker* renvoie une liste des leechers[2] et des seeders[3] partageant actuellement le fichier demandé (ou message d'erreur dans le cas échéant). Une fois cette liste des pairs obtenue, le client va chercher à se connecter à ceux-ci et il commence le téléchargement. La technique est la même que pour le réseau eMule/eDonkey. Les fichiers sont découpés en fragments qui peuvent être envoyés dans n'importe quel ordre. Cela, permet de faire le téléchargement multiple à partir de plusieurs pairs différents. De même, après avoir reçu une partie du fichier, le client servira de source pour l'envoyer vers d'autres utilisateurs. En effet le client BitTorrent informe le tracker à des intervalles de temps réguliers de son état du téléchargement et reçoit du tracker une liste actualisée des pairs.

Contrairement aux autres réseaux tels que FastTrack et eMule/eDonkey, le client BitTorrent ne gère pas de file d'attente. Le système de téléchargement est basé sur un système de récompense ressemblant à l'algorithme *Tit-For-Tat* **[56]** : le principe est simple, « j'envoie des données aux personnes qui m'envoient également ». Ainsi, la priorité s'accordera aux requêtes des pairs qui apportent le plus. Cette technique du « donnant-donnant » incite à la coopération entre les pairs et permet de lutter efficacement contre les *leechers* qui ne font que télécharger sans jamais envoyer de données aux autres

2.4 Conclusion

On a vu quelques exemples de système pair à pair qui se base sur les tables de hachage distribuées (CAN, Chord, Pastry, Tpastry). Ces systèmes structurés présentent un grand nombre de similitudes, la principale différence réside dans l'algorithme de routage utilisé. On a présenté également trois protocoles P2P particuliers (FastTrack, eMule/eDonkey et BitTorrent) qu'ils ont marqué une étape importante dans le monde des protocoles P2P.

Néanmoins, ces protocoles présentent des faiblesses. En effet, un réseau P2P est constitué de pairs inconnus et donc potentiellement dangereux. D'autre part, ce type de réseau est un moyen de diffusion de virus et de vers. Ainsi, un problème de sécurité se

[2] Leechers : c'est un pair qui ne possède pas le fichier en totalité et souhaitant encore le téléchargé.
[3] Seeder : est un client partageant un fichier complet.

pose, un état de l'art sur la sécurité dans ces réseaux fera l'objet des deux chapitres suivants.

Chapitre 3

La sécurité dans les réseaux P2P

3.1 Introduction

La popularité croissante des systèmes P2P et plus particulièrement les systèmes P2P de partage de fichier comme *eDonkey*, *Kazaa* et *Bittorent* a récemment suscité beaucoup d'intérêt à la communauté de l'Internet parce qu'ils présentent une infrastructure totalement distribuée et décentralisée. Ce sont des systèmes ouverts qui ont plusieurs avantages : l'*adaptation*, l'*auto-organisation*, l'*équilibrage de charge*, la *tolérance aux fautes*, la *disponibilité* à travers une forte réputation, le *passage à l'échelle* et l'*aptitude de mettre en commun* et d'utiliser une grande quantité de ressources. Mais cela à quel coût en terme d'intimité et de sécurité ?

Dans ce chapitre, nous commençons d'abord par définir plus précisément la sécurité. Ensuite nous introduisons les notions de sécurité présentée dans le système informatique. Puis, nous présentons les services et les mécanismes de sécurité déjà proposés pour sécuriser les réseaux pair-à-pair.

3.2 Définition de la sécurité

La sécurité informatique consiste à garantir aux données et aux services les cinq propriétés suivantes : *confidentialité*, *intégrité*, *disponibilité*, *authenticité* et la *non répudiation* **[57]**. Sur chaque système, une politique de sécurité définit les propriétés attendues et les moyens à utiliser pour les mettre en œuvre. Dans cette section, nous décrivons les mécanismes de sécurité habituels utilisés dans un environnement centralisé. L'ensemble de ces mécanismes représente ce que l'on est en droit d'attendre d'un système sécurisé. Nous distinguons deux types de mécanismes de sécurité : *la sécurité proactive* et *la sécurité réactive*.

3.2.1 La sécurité proactive

Elle regroupe l'ensemble des mesures de protection de l'information **[57]** :

- *La confidentialité* : consiste à empêcher des utilisateurs non autorisés d'accéder à une information. D'une autre manière, c'est la protection du trafic circulant dans l'analyse.
- *L'intégrité* : est la propriété d'une information à ne pas être altérée.
- *La disponibilité* : est la garantie que l'information soit accessible.
- *L'authentification* : permet de garantir l'identité de l'utilisateur.
- *La non répudiation* : empêche l'utilisateur à nier une action faite.

Ces services reposent sur des mécanismes de *chiffrement* (symétrique ou asymétrique), de *signature cryptographique*, de *contrôle d'accès*, de *contrôle de routage* et de *tiers de confiance* (figure 3.1).

	Chiffrement	*signature*	*Contrôle d'accès*	*Contrôle de routage*	*Tires de confiance*
Authentification	X	X			
Autorisation			X		
Confidentialité	X			X	
Intégrité	X	X			
Non répudiation		X			X

Figure 3.1 : *Mécanismes utilisés par les services de sécurité.*

3.2.2 La sécurité réactive

La sécurité réactive consiste en la détection de comportements malveillants ou anormaux. Ici, nous présentons les antivirus et les systèmes de détection d'intrusion. Les antivirus visent à détecter des fichiers contenants un code malicieux. De tels fichiers utilisent généralement une faille du système pour exécuter le code malicieux et se propager. Les antivirus reposent principalement sur une base de signatures et vérifient que les fichiers ne présentent pas ces signatures. Des mécanismes d'analyse heuristiques sont également présents cherchant à détecter des comportements suspects. Néanmoins et étant donné la grande variété de virus, ces heuristiques sont peu efficaces.

Les systèmes de détection d'intrusion **[100]** essayent de détecter les comportements anormaux du système. Pour cela, il existe deux possibilités :

- *La détection d'un comportement malveillant* : il s'agit de la modélisation des comportements d'attaques. La solution couramment envisagée repose sur une base de signatures avec les problèmes inhérents à cette méthode. En effet, toute attaque récente n'est pas détectée et la mise à jour de cette base est un travail exigé.
- *La détection d'un comportement non prévu* : il s'agit de modéliser le comportement normal du système. La plupart des approches proposant ce modèle repose sur une phase d'apprentissage, qui ne peut être ni exhaustive (certains comportements nouveaux peuvent ne pas se produire), ni garantie (des attaques peuvent avoir lieu pendant ce temps). Dans le cas d'attaques se produisant pendant la phase d'apprentissage, elles sont prises comme un comportement normal et ne sont pas

détectées. Donc, la non exhaustivité génère de fausses alertes et la non garantie peut provoquer des intrusions non détectées.

3.3 Aspects de sécurité dans les réseaux P2P

Les réseaux P2P sont des réseaux totalement distribués et décentralisés. Le passage à l'échelle de ces systèmes, leur nature ouverte et le manque du contrôle centralisé présentent de nouveaux problèmes. Parmi ceux là, *la sécurité*. Cependant, le problème de sécurité de ces systèmes n'est apparu que très récemment (premiers travaux en 2002). Dans ce domaine de sécurité des P2P, il y a 5 services de sécurité pertinents : *disponibilité, authenticité de fichier, anonymat, contrôle d'accès* et *l'équité* **[58]**.

L'environnement où le pair doit fonctionner est hostile et dans lequel n'importe quel pair peut joindre librement le réseau. Ce dernier peut être un pair malveillant dont on ne peut pas avoir confiance pour acheminer les requêtes ou avoir des réponses correctes.

Afin de rendre ce type de réseau plus sécurisé, plusieurs solutions ont été proposées. Ces solutions essayent d'implémenter les services de sécurité, elles peuvent être réparties en trois catégories **[58]** : *identité*, *confiance/réputation* et *incitation*.

1). *L'identité* : les solutions apportées dans cette catégorie se focalisent à réaliser les deux propriétés de sécurité *l'anonymat* et le *contrôle d'accès.*

2). *La confiance/réputation* : les solutions apportées dans cette catégorie se focalisent à effectuer la disponibilité et *l'authenticité du fichier* en utilisant le système de confiance **[69]**.

3). *L'incitation* : les solutions apportées dans cette catégorie s'occupent à établir la propriété de sécurité *l'équité* (en anglais : « *fair trading* ») et la *disponibilité des pairs* en sollicitant les différentes manières **[69]** afin d'inciter les pairs à contribuer au système.

3.3.1 Les services de sécurité

Certaines définitions des services de sécurité ont été présentées par *N.Daswani et.al* dans **[59]**. Ils ont indiqué qu'il est important de développer des techniques qui pourront *prévenir*, *détecter*, *gérer* et permet de *se remettre* des attaques.

- *La disponibilité* : est définie comme la disponibilité des fichiers et des pairs dans le système. Chaque nœud doit être capable de communiquer avec les autres pairs et est capable d'offrir l'accès aux ressources avec qui il collabore.
- *L'authenticité du fichier* : l'authenticité du fichier peut être définie de multiples façons. Quatre définitions sont données **:**
  (1) *Oldest document* « Ancien document ». Avec cette approche, on a confiance à la plus ancienne réponse d'une requête pour l'authentifier.
  (2) *Expert-based* « repose sur des experts », dans ce cas les experts donnent leur opinion sur l'authenticité du fichier.
  (3) *Voted-based* « l'approche basée sur des votes », différents experts donnent leurs opinions sur l'authenticité du fichier puis ces avis rassemblés nous informent sur l'authenticité.
  (4) *Reputation-based* « l'approche basée sur la réputation », elle est similaire à l'approche précédente à la différence que cette dernière prend en considération la réputation des experts.
- *Le contrôle d'accès* **[60]** : il permet de limiter l'accès des pairs aux ressources, sauf à ceux qui en ont droit.
- *L'anonymat* : il rend difficile aux pairs de découvrir qui a créé un fichier, qui l'a stocké, qui en a l'accès et quels sont les documents stockés sur un pair. Les objectifs de cette anonymisation sont à la fois la lutte contre la censure et la crainte des autorités. Le premier réseau de ce type à être proposé est *Freenet*[1]. Il masque les communications par des mécanismes de Proxy et chiffre les liens pour éviter l'écoute. En revanche, ceux-ci sont gourmands en ressource et ralentissent fortement le trafic, et tout le monde n'a pas un Proxy sous la main pour se connecter... Plus récemment, *Mute* **[61]** et *Ants* **[62]** ont proposé l'anonymisation totale des identifiants, c'est-à-dire qu'aucun nœud ne peut faire la correspondance entre un identifiant et une adresse IP. Les mécanismes de routage sont basés sur des algorithmes de fourmis (des agents actifs) et l'anonymat des requêtes et des réponses est garanti. Il existe d'autre réseaux chiffrés et anonymes comme : *Share*[2], et *Kameleon*[3]. 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

[1] http://freenetproject.org/whatis.html
[2] http://fr.wikipedia.org/wiki/Share
[3] http://kameleon.sourceforge.net/

développer très rapidement. Pour contourner ce problème d'anonymat, il existe une autre solution qui est le *réseau d'amis* (en anglais : *friend-to- friend*) avec des relations de confiance. L'utilisateur permet à ses amis de joindre son réseau « privé » et ainsi partager ses données uniquement avec eux. C'est le principe des logiciels comme *Waste*[4] et *Darkne*t. L'inconvénient d'une telle solution est qu'il faut quelques pairs avec de larges bandes passantes pour obtenir de bonnes performances.

- *L'équité* : les systèmes P2P reposent sur la coopération de tous les participants. Il est important d'empêcher que certains nœuds profitent plus qu'ils ne coopèrent **[63]**, cela signifie qu'un pair qui bénéficie de réseau doit contribuer aussi à cela. La bande passante et l'espace disque sont les deux principaux exemples de ressources limitées. Alors que la bande passante semble encore difficile à partager équitablement. Des solutions ont été proposées dans **[64]** pour l'espace disque.

3.3.2 Les mécanismes de sécurité

Un réseau P2P étant constitué de pairs inconnus et donc potentiellement dangereux, il convient de mettre en œuvre différents mécanismes de sécurité. Les mécanismes de sécurité traditionnels qui protègent les systèmes et les données, ainsi les systèmes comme les pare-feux ne peuvent pas protéger les systèmes pair-à-pair puisqu'ils sont essentiellement et totalement distribués. Autrement dit, de tels mécanismes peuvent interdire la communication pair-à-pair. Donc de nouveaux concepts de sécurités sont requis pour se protéger des intrus et des attaques en permettant toujours l'interaction et le traitement distribué dans les systèmes pair-à-pair **[2] [64]**.

Parmi cela, on trouve *la cryptographie* qui protège les données par le *chiffrage* avec clés multiples (clé public/clé privée). Pour protéger *IPR* (des Droits de Propriété intellectuelle) en permettant toujours l'utilisation publique de contenu, les signatures peuvent être ajoutées aux données, par exemple des techniques comme *watermarking* et *steganography*. Dans l'environnement de calcul distribué, un code est exécuté sur des ordinateurs. D'une part, ces systèmes d'ordinateur doivent être protégés du code malveillant qui peut corrompre la machine traitant. D'autre part, des résultats et des données sensibles étant traitées doivent être sécurisés. Il y a quelques méthodes étant développées pour

[4] http://waste.sourceforge.net/

s'occuper de ces questions, par exemple *sandboxing*. Et pour éviter le phénomène de pure consommation, une solution a été proposée : l'utilisation du *système de crédit* **[65]**.

Manish Parashar et.al **[66]** ont présenté des techniques de sécurités qui protègent les applications P2P :

- *Cryptographie* : elle est utilisée pour chiffrer des messages envoyés entre les deux parties de communication afin qu'un espion ne puisse être capable de les décoder. Il y a plusieurs méthodes pour l'implémentation de la cryptographie. Le plus populaire est le chiffrement par clé public. Il est très utile dans les systèmes pair-à-pair parce qu'il peut protéger un message et assurer son intégrité. Il peut aussi être employé dans l'installation des canaux de communications virtuelles entre des pairs en utilisant un réseau 3-tiers comme l'Internet. On utilise le chiffrage avec la clé public pour répondre à beaucoup d'exigences de sécurité tel que : *la sécurité de l'intégrité des messages* et *la gestion des droits digitaux*.
- *Cartes à puce* : ce sont des cartes en plastique semblables aux cartes de crédit qui contiennent une puce de circuits intégrés dont la capacité de stockage de l'information et la puissance de traitement ne sont pas disponibles dans des cartes de raie magnétique. L'accouplement du microprocesseur avec un lecteur produit une carte dont la puissance permet plusieurs applications. Grâce à ces dispositifs de contrôle d'accès incorporant le chiffrage et l'authentification, les données sont disponibles seulement aux utilisateurs autorisés. Pour la gestion de réseau pair à pair, les cartes à puce permettent une gestion plus sûre et plus simple pour divers réseaux avec moins de dépenses significatives. Pour le contrôle d'accès, une carte unique est utilisée pour l'identification personnelle. Elle a aussi d'autres fonctionnalités intégrées.
- *Stéganography* : c'est le processus de cacher l'information de sécurité dans les données par opposition au chiffrage. Il aspire à rendre les données illisibles aux téléspectateurs non autorisés. Il est très utile parce qu'il peut fournir des données à une identité unique incorporée directement dans l'information. Ce processus est connu sous l'appellation *watermarking*[5]. Il peut être utilisé pour signer les duplications et protéger les droits de propriété intellectuelle.

[5] http://en.wikipedia.org/wiki/Digital_watermarking

3.4 D'autres fonctions de sécurité

Dans cette section, nous donnons des définitions de quelques fonctions de la sécurité dite classique qui sont utilisées dans la sécurité informatique ainsi la sécurité P2P et qu'ils se présentent dans ce mémoire.

3.4.1 Chiffrement symétrique

La cryptographie symétrique consiste à chiffrer et à déchiffrer un message en partageant la même clef secrète. Seuls les usagers qui ont cette clef peuvent déchiffrer les messages chiffrés par cette clef. Dans la cryptographie symétrique l'algorithme de chiffrement est le même que celui de déchiffrement, ainsi la taille de la clef joue un rôle important contre la cryptanalyse. Les crypto systèmes symétriques sont caractérisés en général par leur rapidité, soit lors de chiffrement ou lors de déchiffrement. Les algorithmes symétriques connus DES, 3-DES, AES.

3.4.2 Chiffrement asymétrique

Le chiffrement asymétrique ou à clef publique est apparu pour la première fois en 1976 à l'université de Stanford. Ce sont les deux chercheurs Diffie et Hellman **[101]** qui ont amené ce type de chiffrement à l'existence[6]. Dans le chiffrement asymétrique, il existe deux types de clefs, la première est la clef publique, elle est connue par tous les utilisateurs. La deuxième est la clef privée, elle est connue seulement par l'utilisateur qui la génère. On utilise en cryptographie asymétrique des fonctions à sens unique et à brèche secrète. Une telle fonction est difficile à inverser, à moins de posséder une information particulière, tenue secrète, nommée clé privée. L'exemple le plus connu de Fonction à brèche secrète est RSA **[102]**.

3.4.3 l'audit du système

C'est l'observation, l'enregistrement, l'analyse et la compréhension des événements importants ou anormaux qui vont concourir à reconstituer le fil de son histoire, après la constatation d'une panne ou d'une attaque. Dans la pratique, on enregistre, dans les

[6] http://fr.wikipedia.org/wiki/Cryptographie_asym%C3%A9trique

différents dispositifs de sécurité, des journaux infalsifiables qui seront des témoins de confiance chargés d'interpréter la trame des opérations et d'imputer la responsabilité d'une erreur ou d'un acte malveillant à son initiateur. Cette fonction d'audit, témoin de la mémoire du système, est déterminante dans un système. L'audibilité est une fonction qui consiste à pouvoir récupérer des preuves numériques incontestables, en cas de perquisition des données ou d'examen ultérieur des activités. Une approche retenue dans **[64]** propose un audit distribué pour l'espace disque.

3.4.4 Fonctions de hachage

Une fonction de hachage permet de calculer le résumé (une empreinte) d'un message de longueur arbitraire, tandis que son hachage est de longueur fixe. Parmi les fonctions de hachage les plus utilisées et standardisées, il existe la fonction MD4, MD5 (Message Digest 4 ou 5) qui donne une empreinte de 128 bits et SHA-1 (Secure Hash Algorithm-1) qui fournit des résumés de 160 bits. C'est en particulier cette fonction que l'on utilise pour valider l'intégrité d'un transfert de documents ou pour vérifier qu'un fichier n'a pas été corrompu, le résultat est appelé MAC (Message Authentication Code).

3.4.5 Signature numérique

Une signature numérique est une empreinte (d'un message) chiffrée par la clé privée de la source, cette empreinte chiffrée étant concaténée avec le message originel. La signature numérique permet de prouver au destinataire que le message n'a pas été altéré (l'intégrité du message), elle permet aussi de prouver que seule la source a envoyé le message (l'authentification de la source). On signe un document via des fonctions de hachage.

3.4.6 Infrastructures de confiance

Les systèmes de cryptage asymétrique permettent de fournir un canal sécurisé pour échanger les clés. Ainsi, la publication de la clé publique à grande échelle doit se faire en toute confiance pour assurer que la clé publique est bien celle de son propriétaire, le propriétaire de la clé est digne de confiance et la clé est toujours valide. Cependant, n'importe qui peut, à présent, engendrer un couple de clé privée/clé publique en récupérant un logiciel sur Internet. Aussi, lorsqu'une personne publie son nom et sa clé publique

associée, un pirate peut se glisser sous cet affichage et usurper la clé, en proposant sa propre clé publique. De cette façon, le pirate pourra décoder avec sa propre clé privée les messages destinés à l'interlocuteur véridique. C'est ce qu'on appelle une attaque par l'homme au milieu (Man In The Middle **[68]**).

Pour pallier ce problème, une PKI (Public Key Infrastructure) est utilisée. C'est l'ensemble des solutions techniques basées sur la cryptographie asymétrique et qui utilise une autorité de certification. Le rôle de l'infrastructure à clé publique est multiple et couvre notamment les champs suivants :

- Enregistrer les requêtes des clés en vérifiant l'identité des demandeurs.
- Générer les paires de clefs (clé privée/clé publique).
- Garantir la confidentialité des clefs privées correspondant aux clés publiques.
- Certifier l'association entre chaque utilisateur et sa clé publique.
- Révoquer des clés (en cas de perte par son propriétaire, expiration de sa date de validité ou de compromission).

3.4.7 Autorités de confiance

C'est un tiers de confiance appelée une autorité de certification (Certification Authority, abrégé CA). Il représente une entité ou un service qui permet de délivrer des certificats. La CA fait la liaison entre la clef publique et l'identité d'un client, en faisant signer la clé publique par cette autorité digne de confiance, qui va ainsi garantir que la clé publique appartient bien au bon interlocuteur. Elle permet aussi d'établir la confiance entre la clé publique et la clé privée.

3.5 Conclusion

La solution pour assurer la sécurité d'un réseau pair-à-pair se présente dans la conception des mécanismes qui assurent la disponibilité, l'authenticité de fichier, l'anonymat, l'équité et le contrôle d'accès. Dans ce chapitre, nous avons énuméré les différents services de sécurité P2P, puis nous avons cité les mécanismes de sécurité qui protège les applications P2P et d'autres fonctions et approches utilisées pour implémenter les propriétés de sécurité P2P.

Dans les systèmes actuels, la sécurité se résume principalement à l'anonymat. Il s'agit de la propriété caractérisant les réseaux dits « de troisième génération ». D'autres études sont apparues au sujet de la disponibilité et l'authentification dans des réseaux structurés.

Beaucoup de menaces et attaques sont dues aux faiblesses innées de l'architecture P2P. Ces faiblesses aboutissent aux problèmes de sécurité dans les réseaux P2P. Dans le chapitre qui suit nous allons aborder les différentes menaces et attaques traitées dans la littérature sur ces réseaux avec les solutions présentées pour contrer ces attaques.

Chapitre 4

Vulnérabilités des systèmes P2P

4.1 Introduction

L'adoption d'un système P2P implique une manière différente d'envisager la notion de sécurité. La nature distribuée du système P2P et le nombre croissant d'unités amenées à collaborer dans une telle perspective semblent avoir présenté une nouvelle variété de vulnérabilités menaçantes. Les vulnérabilités deviennent complexes en raison des buts architecturaux comme la distribution de la charge, la facilité de la recherche et la simplification de la reconfiguration. Les systèmes P2P sont plus vulnérables à des menaces bien connues, comme : *le spoofing, la violation de l'intimité* et l'attaque *Man-in-the-Middle*…etc.

Marling et Javed dans **[68]** ont présenté une étude sur ces nouvelles vulnérabilités spécifiques aux systèmes P2P. Des attaques spécifiques aux réseaux P2P et leurs solutions possibles ont fait l'objet d'études préalables dans différents articles **[69] [70] [71] [72] [73] [74]**.

Selon la classification présentée par Marling et Javed **[68]** l'attaque (ou la vulnérabilité) est subdivisée en trois couches (niveaux) : *la couche inférieure, la couche moyenne* et *la couche supérieure* (ou *la couche P2P*). Ces trois niveaux correspondent à ce qui est, en réalité, exploité dans l'attaque. Ils sont vulnérables à n'importe quel modèle de réseau, (figure 4.1)

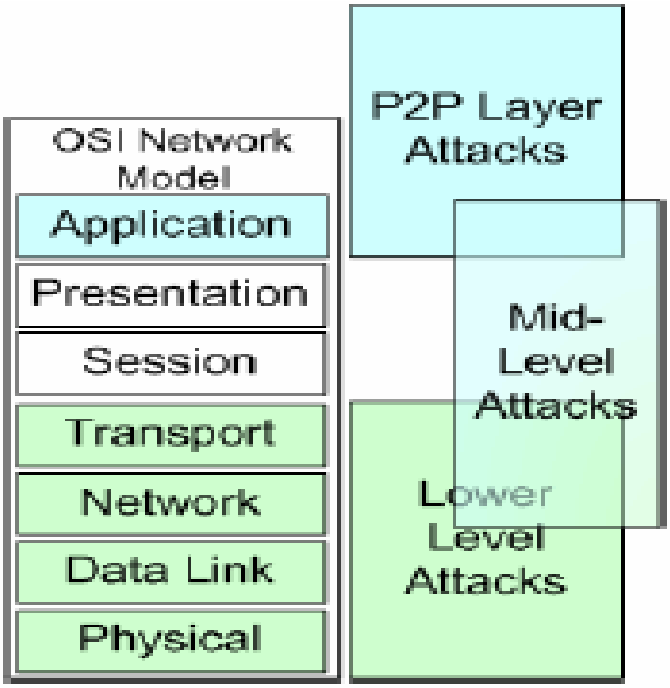


Figure 4.1 : *Système de classification de couche*.

4.2 Attaques Générales et Défenses

D'abord, nous exposons différentes vulnérabilités ou attaques trouvées dans les réseaux généraux qui ont été classées par Marling et Javed **[68]** dans les deux premières couches : *inférieure* et *moyenne*.

4.2.1 L'attaque de dénie de service (*DoS*)

Une attaque de dénie de service est une attaque sur un ordinateur ou un réseau qui cause la perte d'un service **[75]**. Il existe une infinité de formes et méthodes pour perpétrer une attaque de DoS. Dans le cas de réseaux P2P, la forme la plus commune d'une attaque de DoS est une tentative d'inonder le réseau avec des paquets simulés pour y empêcher des requêtes valides ou livrés des messages. Ainsi, s'arrêtent toutes les communications le long des chemins affectés d'où la diminution du trafic réseau. Une autre méthode consiste à noyer la victime par des calculs fastidieux de sorte qu'elle soit occupée pour répondre à toute autre requête. Les attaques DoS sont beaucoup plus efficaces si plusieurs hôtes sont impliqués. On parle alors d'une attaque DDoS (Dénie de service Distribué) **[67]**.

Dans ce type d'attaque, les ordinateurs attaquants sont souvent des ordinateurs individuels avec des connexions à large bande qui ont été infectés par un virus ou un torjan. L'attaquant peut alors contrôler ces machines à distance (qualifiées comme des zombies ou des esclaves) et emmener une attaque à n'importe quel hôte ou réseau (figure 4.2).

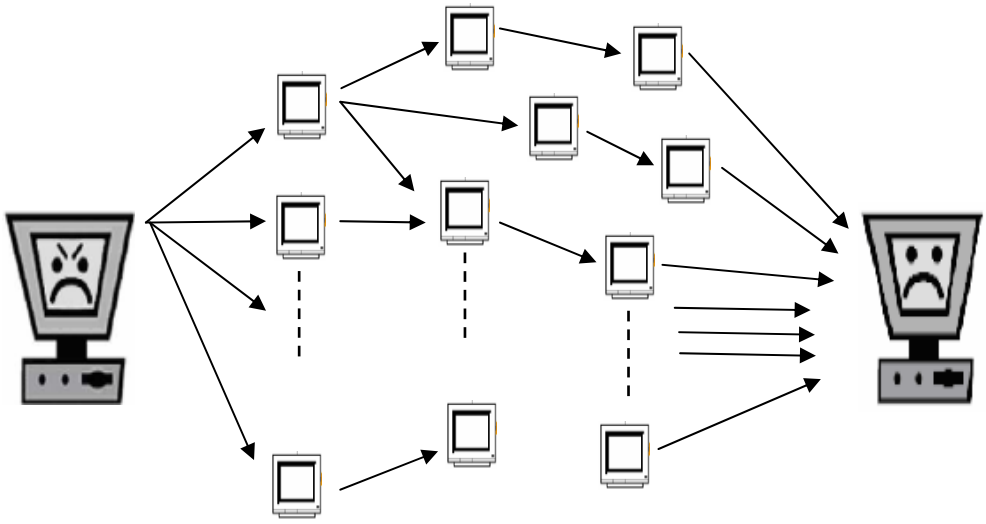


Figure 4.2 : *Une attaque de DDoS*

- **Défenses :** afin de se défendre contre les attaques DoS, il faudrait les détecter. Les signes d'un DoS (ou même DDoS) sont semblables aux signes de haute utilisation de réseau. Un autre facteur important c'est que DDoS est très difficile à bloquer à cause

du grand nombre de nœuds qui peuvent être impliqués. Ce fait est amplifié quand les nœuds attaquants utilisent des nœuds normaux pour relancer leurs attaques (où l'attaque semble venir de nœuds normaux). A cause de ces deux derniers, il est impossible de bloquer toutes les attaques de DoS. Cependant, il existe une technique largement employée pour rendre cette attaque peu fonctionnelle : la « *tarification (*en anglais : *pricing) »* (figure 4.3).

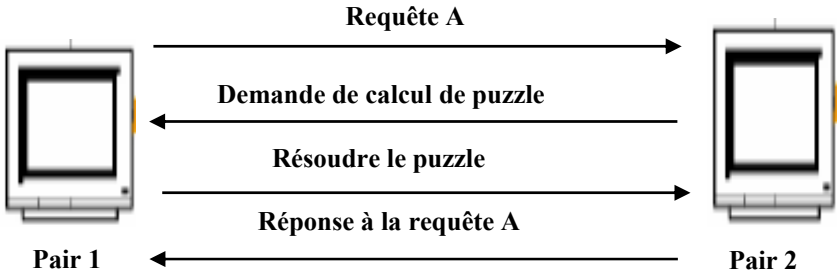


Figure 4.3 : *La tarification est une méthode de limiter la vitesse des demandes*

La tarification est employée pour limiter la vitesse à laquelle les nœuds font des demandes dans le réseau. Quand l'attaquant veut demander quelque chose d'un nœud, ce dernier lui répond par une énigme (puzzle) déterminée par un calcul intensif **[68]**. A ce moment là l'attaquant (ou n'importe quel nœud faisant la demande) doit résoudre cette énigme et fournir une réponse valable avant que la demande ne soit même reconnue.

4.2.2 L'attaque Man-in-the-Middle (*MitM*)

Dans l'attaque MitM **[68] [103]**, l'attaquant se place entre deux autres noeuds dans le réseau pour intercepter toute communication entre eux toute en demeurant soit indétectable et passif soit actif en manipulant la communication. Ceci est réalisé en modifiant les messages ou en insérant de faux messages d'un nœud vers l'autre. Par ce mécanisme, l'attaquant peut prendre l'identité du nœud : soit l'un ou/et l'autre. MitM peut ainsi réaliser une variété de buts. Dans des nombreux cas, c'est la subtilisation de l'identité ou l'envoi de fausse information.

Cette attaque particulière peut être exécutée au niveau de la couche réseau de la communication. Auquel cas, l'attaquant peut voir toutes les communications de niveau

inférieur entre les deux nœuds et comme cette couche est au-dessous de la couche P2P, l'attaquant falsifie n'importe quel type de message P2P sans aucun problème.

Pour attaquer le niveau réseau, avec MitM, l'attaquant doit trouver une façon de se placer dans le chemin entre les deux nœuds, ce qui peut être une tâche complexe. Cependant, les systèmes P2P sont extrêmement vulnérables à cette attaque car ils n'ont aucun contrôle du placement de nœud dans l'espace ID (exemple : Chord, Pastry).

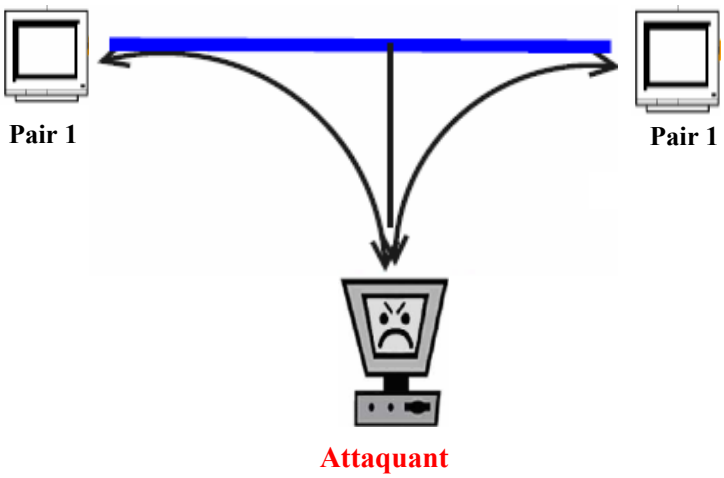


Figure 4.4: *L'attaque Man in the Middle.*

- **Défenses :** la méthode principale pour se défendre contre les attaques MitM est de les rendre aussi inefficaces que possible. Sans quelques types importants de noeud comme : une autorité de certificat (CA), un serveur d'indexation ou un super-nœud, l'attaquant MitM se limite à une compromission de l'intégrité de deux nœuds dans le réseau. Une telle attaque n'est pas une menace à l'ensemble du réseau dans la mesure où ce dernier est tolérant aux pannes et peut manipuler la perte des trois nœuds impliqués (2 pairs et 1 pair attaquant). Néanmoins, pour ces réseaux, l'attaque MitM pourrait être effectuée à une large échelle (comme Eclipse).

  Généralement, la prévention de la falsification d'information la plus admise est l'utilisation de signatures digitales. Ces signatures sont basées sur la cryptographie (clé public/clé privée) qui permet la vérification de l'intégrité d'un document, d'une requête ou d'un message. Une telle technologie a été utilisée dans le courrier électronique pendant plusieurs années car elle fournit l'authentification d'un message. Cette même méthode peut être utilisée pour détecter si un message a été

modifié dans un réseau P2P en ajoutant simplement une signature digitale à la fin du message. D'où le nœud récepteur (au même le nœud au milieu) sera capable de vérifier si le message est inaltéré et est effectivement originaire de la vraie source.

4.2.3 L'attaque du Ver (Worm)

Les vers présentent une des menaces les plus importantes à l'Internet. Actuellement, les vers comme *Code Red* **[76]** ou Nimda **[77]** sont capables d'infecter des centaines de milliers d'hôtes dans un temps record.

Le ver (ou *Worm*) **[78] [79]** est souvent confondu aux virus informatiques. Ils sont tous les deux de types de *malware*[1]. Le ver à la différence de virus, s'étend en envoyant les copies de lui-même à travers le réseau, en exploitant les vulnérabilités dans le programme ou des applications systèmes contrairement aux virus informatiques qui attendent passivement avant d'infecter des fichiers sur des systèmes différents. Toutefois, un ver est un programme informatique auto-dupliqué, autonome et ne doit pas faire partie d'un autre programme pour se propager. Les vers informatiques ont souvent un but secondaire autre que leurs propres diffusions. Cela pourrait être soit d'envoyer les *spam* par des hôtes infectés soit télécharger d'autre code malveillant pour contrôler l'ordinateur d'une façon permanente en utilisant le *backdoor*[2]ou *Torjan torse*[3]. Ils peuvent s'étendre à des réseaux informatiques en utilisant des emails, des réseaux pair-à-pair **[78]** ou même des téléphones portables.

Cette attaque produit des menaces très significatives aux réseaux P2P. Les auteurs de **[68]** ont classifié le ver comme une attaque de niveau moyen. D'après son étude, la plupart des vers se propagent au niveau inférieur. Le fait qu'un nœud utilisé dans un réseau P2P soit infecté cela peut amplifier l'infection et étendre la menace aux autres nœuds du réseau P2P. Beaucoup de réseaux P2P exécutent le même logiciel. Cela signifie que lorsqu'une vulnérabilité est dans ce dernier, tous les nœuds du réseau sont aussi vulnérables (figure 4.5).

[1] Malware : malicious topic software (logiciel malveillant).
[2] Backdoor : façon non documentée d'accéder au système ou aux données informatiques.
[3] Torjan torse : cheval de troie

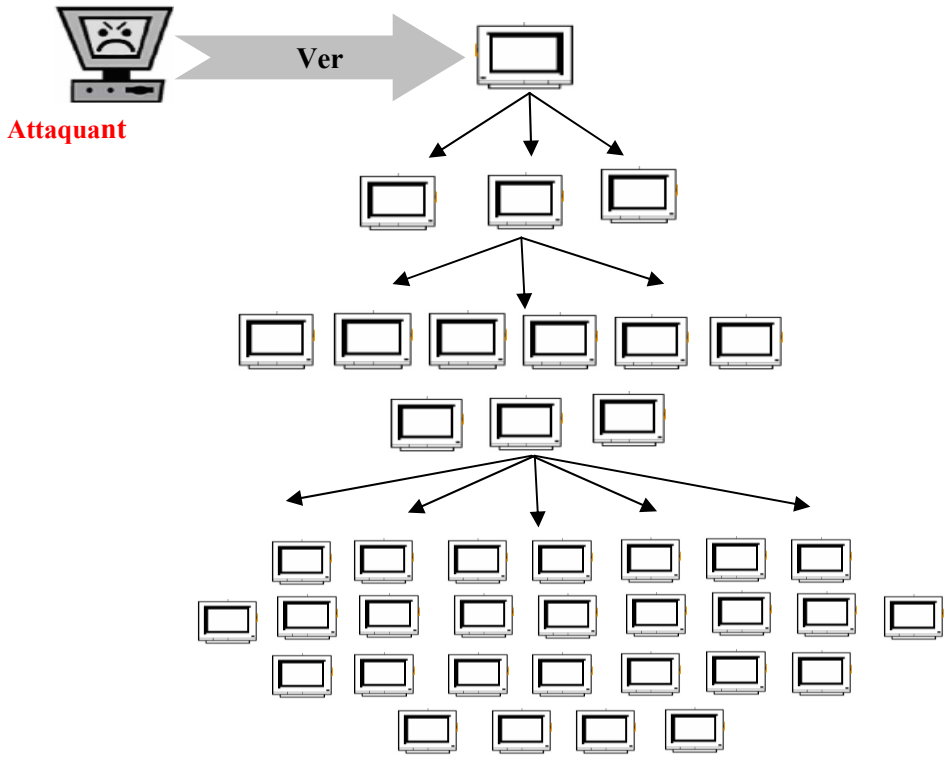


Figure 4.5 : *La propagation du Ver*.

- **Défenses :** différentes solutions ont été présentées pour se défendre contre l'attaque des vers comme :

  (1) maintenir le sécurité des applications (par exemple : la programmation des clients P2P avec un fort langage qui pourrait éviter beaucoup d'anomalies de sécurité «tampon surcharger »),

  (2) éviter les réseaux hybrides. A cause de la meilleure connectivité de leurs super-nœuds, le ver s'étend plus efficacement,

  (3) utiliser des systèmes d'exploitation endurci (comme Open BSD),

  (4) utiliser la nature ouverte du réseau lui-même, en développant des standards et des protocoles libres afin d'encourager les développeurs à créer leur propres clients P2P.

4.2.4 Le facteur humain

Le facteur humain doit être toujours pris en considération quand la sécurité est discutée. L'amélioration des applications P2P est due à son expérimentation et aussi la facilité d'installation et l'utilisation, le bas prix (généralement gratuit) et sa grande

récompense. Les utilisateurs novices ont peu de difficultés à utiliser de telles applications pour télécharger des fichiers que d'autres utilisateurs partagent intentionnellement ou accidentellement sur le réseau P2P. C'est encore un autre problème de sécurité des applications P2P qui se pose. En autorisant un utilisateur (particulièrement un novice) à faire des choix d'accessibilité à leurs fichiers est un risque significatif, comme Kazaa qui peut entraîner un utilisateur à partager inconsciemment le contenu de ses documents et au pire des cas son disque dur entier. Malheureusement, les utilisateurs novices ne connaissent pas les implications de leur passivité en ce qui concerne la sécurité. Des millions de pairs P2P laissés sans surveillance sont vulnérables pendant des grandes périodes de temps. Des utilisateurs malveillants ayant l'aptitude de piratage peuvent profiter de telles situations.

4.3 Les attaques spécifiques aux réseaux P2P

Dans cette section nous présenterons des attaques plus spécifiques particulièrement conçues pour des réseaux P2P avec leurs solutions proposées. Beaucoup d'insistance est concentrée sur les systèmes virtuels structurés, mais très peu d'attention a été investi aux réseaux non structurés qui sont probablement aussi sensibles **[75]**.

4.3.1 L'attaque rationnelle

Pour que les réseaux pair-à-pair puissent être efficaces, les nœuds participant dans le réseau doivent coopérer. Cependant, dans la plupart des scénarios, un nœud interprète un intérêt personnel et on ne peut attendre ni la coopération, ni l'appliquer **[73]**. Cela suppose que la plupart des nœuds exposeront le comportement raisonnable et essayeront de maximiser leur consommation des ressources du système, en réduisant au minimum l'utilisation de leurs propres ressources. Il y a beaucoup de raisons derrière ce comportement par exemple :

- Préserver la largeur de la bande de téléchargement qui est fortement contrôlée par la plupart des fournisseurs des services Internet (ISP).
- Des effets légaux, dans le cas de matériel protégé par le droit d'auteur, le fichier partageant peut avoir des fins négatives. Comme c'est illégal et est tout à fait facile aux autorités de découvrir qui partage des fichiers spécifiques, il peut conduire à une très grande pénalité.

- Le principe de l'individualiste dont certaines personnes ne coopéreront pas pour le simple désir de ne pas aider le réseau.

Ceux-ci ont d'assez bonnes raisons pour motiver des nœuds à devenir des égoïstes et d'avoir des intérêts personnels. Si un grand nombre de nœuds refuse de contribuer, le système pourrait se déstabiliser. Il existe deux classes de base pour cette attaque :

(1). *La restriction de contenu* : les utilisateurs ne partagent pas leurs contenus sur le réseau.

(2). *La restriction de ressource* : les utilisateurs ne contribuent pas leurs ressources sur le réseau.

- **Défenses :** beaucoup de réseaux éprouvent ce problème. Pour que les systèmes P2P réussissent, ils doivent être conçus pour être robustes contre ces classes d'échec. Napster est le premier des réseaux qui a essayé de résoudre les restrictions de contenu et de ressources. Samsara **[80]** est un système P2P de stockage où un nœud peut utiliser autant d'espace que sur un autre nœud tant qu'il obéit au réseau. Parmi tous les réseaux, Bittorrent semble avoir la meilleure défense contre les attaques rationnelles. Il s'adresse à la restriction de ressource. Pour cela, il exécute un système d'échange de gros fragments de données. Plus le nœud partage avec d'autres, plus il reçoit. Ainsi, plus le nœud tolère le partage avec d'autres, plus il obtient un téléchargement plus rapide (algorithme *tit-for-tat*).

4.3.2 L'attaque Sybil

L'une des premières attaques orientée vers les systèmes P2P est l'attaque Sybil **[81]**. C'est une attaque qui fait partie de l'attaque du plan contrôle où une seule entité malveillante représente un nombre différent d'utilisateurs (des identités multiples) pour gagner le contrôle d'une partie du réseau (figure 4.6).

Une fois que cela a été accompli, l'attaquant peut abuser du protocole de n'importe quelle manière possible. Par exemple, il pourrait obtenir la responsabilité de certains fichiers et envisager à les polluer. Si l'attaquant peut placer ses identités d'une façon stratégique dans l'espace ID, les dégâts peuvent être considérables. Il pourrait vouloir

continuer dans une attaque d'Eclipse, ou ralentir le réseau en détournant les requêtes dans une fausse direction.

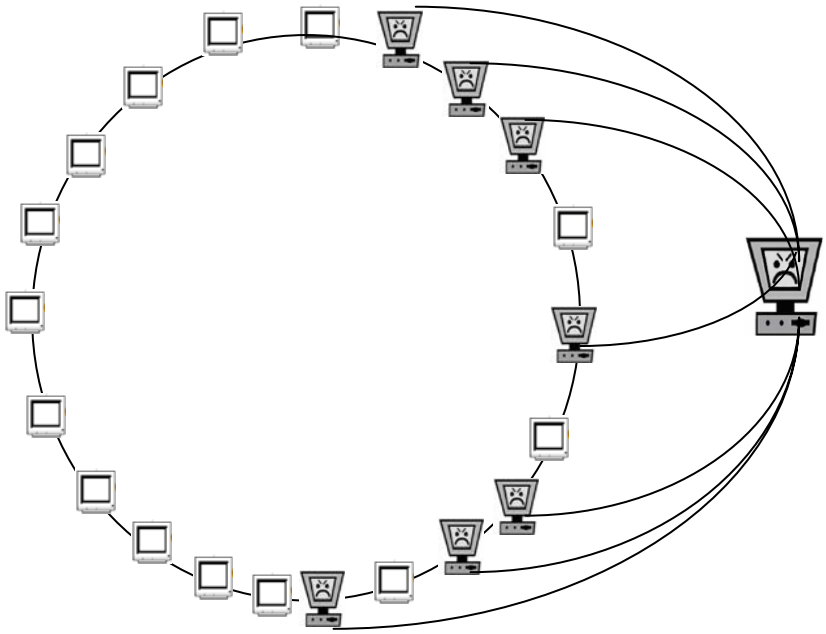


Figure 4 .6 : *Exemple d'une attaque Sybil contre le réseau Pastry.*

- **Défenses :** l'attaque Sybil dans le réseau P2P a été mentionnée par *Douceur* **[81]**. L'auteur a montré que sans une autorité de confiance centrale, il n'est pas possible d'arrêter l'attaque sybil. Les systèmes basés sur la réputation pourraient être capables de ralentir l'attaque, mais ne feront pas beaucoup plus. En effet, une fois que l'attaquant a légalement validé une certaine quantité d'identités, il peut valider le reste. Plusieurs articles **[82] [83]** proposent une autorité centrale pour fournir un niveau minimal de confiance au système, comme une solution aussi bien qu'un protocole basé sur clé public/clé privée **[68]**. Chaque nœud doit signer ses messages et répondre à un challenge par l'autorité. Sans doute, un attaquant simulant beaucoup d'identités aurait besoin d'énormes ressources pour être capable de répondre à tous les challenges périodiquement soumis à chacune de ses identités. Ainsi, l'attaquant prendra beaucoup de temps : cette solution dégrade le modèle P2P en représentant un point centralisé d'échec qui peut facilement être attaqué.

D'autres solutions ont été proposées afin de résoudre ou atténuer ce problème. Elles sont basées sur l'idée de considérer des identités comme des ressources précieuses et forcer des pairs (ou utilisateurs) à payer **[84]**, ou limiter le nombre d'identités publiées **[68]**. Cependant, ces mécanismes ont tendance à être très compliqués et ont un risque non négligeable à être exploités aussi. Ce qui décourage les pairs qui se comportent bien à rejoindre ces systèmes overlay (virtuelle). Deux travaux récents **[85] [86]** semblent encourageants. Le premier **[85]** est basé sur des « *énigmes informatiques* », le deuxième **[86]** est basé sur le « réseau social » entre les identités d'utilisateurs. Ce nouveau système est basé sur le fait qu'une extrémité entre deux identités indique qu'une relation de confiance est établie par l'homme entre les utilisateurs eux-mêmes.

4.3.3 L'attaque Eclipse

L'attaque la plus générale est l'attaque Eclipse **[87]**. Dans cette attaque un ensemble de pairs connectés sont capturés ou contrefaits de telle façon à ce que le cheminement des messages puisse être bloqué ou changé à des fins malveillantes.

Selon *Singh* et.*al* **[87]** les différents systèmes overlay sont vulnérables à l'attaque d'éclipse. En raison de la structure nettement contrôlée des réseaux structurés, les attaquants peuvent facilement cibler les pairs à attaquer. Cette attaque est d'une certaine manière une attaque MitM à large échelle **[68] [73]**. Une attaque Eclipse peut être aussi une généralisation et une suite à l'attaque Sybil. Tel qu'un attaquant peut employer d'abord cette attaque pour déclencher une attaque Eclipse en produisant de fausses identités de différents pairs afin d'envahir le voisinage des nœuds corrects. Ainsi, ces derniers sont éclipsés depuis le réseau.

Cependant, si un attaquant gère une attaque Eclipse, il peut attaquer le réseau de plusieurs façons :

(1). Il peut attaquer sur le plan contrôle, en réacheminant inefficacement chaque message.

(2). Il peut décider d'abandonner touts les messages qu'il reçoit, ainsi il subdivise le réseau en de sous réseaux différents (figure 4.7).

(3). Il peut attaquer sur le plan donné, en injectant des fichiers pollués ou demandant des fichiers pollués de la part des nœuds sains (ou corrects). Ces fichiers sont cachés ou copiés le long du chemin.

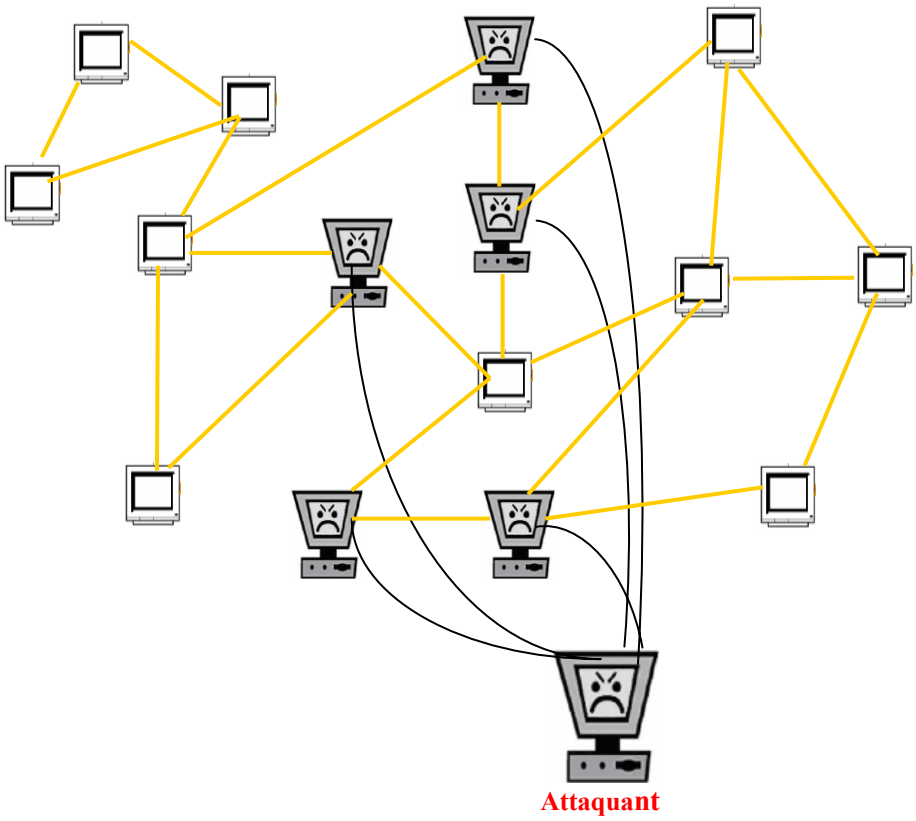


Figure 4.7 : *Exemple de l'attaque Eclipse, qui divise le réseau en deux parties.*

- **Défenses :** comme contre les attaques MitM, des protocoles cryptographiques soigneusement choisis peuvent empêcher une telle attaque (prévenir les faux messages, la modification des messages et la lecture passive des messages). La tarification (*en anglais : pricing*) pourrait aussi aider contre l'attaque Sybil. Le problème avec telles solutions consiste en ce qu'ils ralentissent et nuisent la scalabilité du réseau. Une solution est proposée par *A.Singh et.al* **[87]** basée sur « le *degré de l'audit (en anglais : auditing degree)* » et ils prétendent que leur solution est assez générale pour les systèmes non structurés alors que leur évaluation est focalisée sur les systèmes structurés. Une étude a été faite dans **[88]** qui montre l'impact de l'attaque Eclipse sur les systèmes virtuels structurés et présente les limites des défenses qui existent. Cependant, *Wallach et.al* **[89]** découragent l'utilisation de l'audit car d'une part, il est à peine réalisable d'une façon distribuée et d'autre part,

les attaquants peuvent échapper aux sanctions produites par le processus d'audit (par exemple : l'utilisation de l'attaque Sybil).

4.3.4 L'attaque d'empoisonnement

Les attaques d'empoisonnement peuvent se produire dans les réseaux P2P. Les attaquants utilisent de fausses informations, par exemple : de faux indexes de fichier, de fausses adresses IP ou de fausses tables de routage, afin de violer l'intégrité du système P2P **[90] [91] [92]**.

Un premier type d'attaque consiste à falsifier les *tables de routage* **[93]**. Chaque pair participe au système de routage du réseau. Un nœud malicieux peut donc essayer d'introduire de fausses destinations qui ne mèneront nul part. Les nœuds construisent leurs tables 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, sauf qu'ici, les nœuds ne sont pas forcements éclipsés, mais peuvent posséder de mauvaises informations et conduire à un mauvais fonctionnement de l'ensemble du réseau.

Lors de leur connexion dans le réseau, les nœuds sont aussi vulnérables. Ils doivent en effet se connecter à d'autres pairs afin d'obtenir les informations nécessaires pour participer au réseau. Si un nœud malicieux parvient à devenir 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. Ils pourront ainsi détecter les anomalies.

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

- **Défenses :** les attaques d'empoisonnement des tables de routages 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 les nœuds malicieux peuvent donc être détectées. Un moyen de contrer les attaques sur les données est la réplication : si les données sont présentées sur plusieurs pairs, alors le client pourra interroger les réplications pour savoir si la ressource est vraiment indisponible. L'objectif est d'éviter qu'un seul nœud soit responsable d'une ressource.

4.4 Conclusion

Les P2P actuels 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 « hybrides », s'appuyant donc sur l'utilisation de serveurs centraux qui malgré leurs avantages, constituent une véritable faiblesse.

Ce chapitre présente quelques attaques communes et leurs défenses dans les réseaux P2P actuels. Certaines d'entre elles sont des attaques habituelles dans l'Internet, comme, l'empoisonnement, D/DoS, Vers, …etc. d'autres sont des attaques particulières contre des réseaux P2P virtuels, comme l'attaque Sybil et l'attaque Eclipse. Ces deux dernières attaques sont vraiment deux grandes menaces au réseau P2P. En particulier les réseaux virtuels basés DHT, car les attaquants peuvent contrôler une partie significative du réseau. De plus, il n'y a aucun mécanisme mûr et complet contre elles. On assiste aujourd'hui à l'apparition de protocoles « cryptés » et «anonymes ». Le réseau virtuel anonyme peut cacher les adresses IP des pairs et peut efficacement défendre quelques adresses IP des attaques où diminuent les occasions de telles attaques. Mais un tel réseau P2P est si cher à mettre en œuvre à cause de la cryptographie.

De nos jours, le développement des mécanismes de sécurité dans des réseaux P2P est toujours un sujet important à étudier et à développer.

Chapitre 5

Un modèle de réalisation de l'attaque Eclipse dans un réseau eDonkey

5.1 Introduction

Dans ce chapitre nous allons présenter un modèle de réalisation de l'attaque Eclipse dans un réseau eDonkey. Il faut noter que contrairement à une attaque comme les Vers où l'on peut parler de la propagation, il n'est pas commode de parler de la propagation de l'attaque Eclipse, mais plutôt de la réalisation.

Après une étude sur l'attaque Eclipse, ainsi que l'architecture eDonkey/eMule, on constate que lorsqu'on parle de la réalisation de l'attaque Eclipse sur un réseau eDonkey,

on comprend que la réalisation se fait entre un serveur et ses clients seulement. Ainsi, il est difficile d'envisager une propagation entre les serveurs.

Néanmoins, pour réussir à faire cette propagation, on propose la combinaison entre les deux attaques : Eclipse et Ver car la propagation de l'attaque Ver est plus efficace dans les réseaux hybrides à cause de la meilleure connectivité de leurs super-nœuds. Donc on va se servir de l'attaque Ver pour monter et propager une attaque Eclipse et ainsi parvenir à subdiviser le réseau eDonkey en plusieurs sous-réseaux qui fonctionnent chacun à part.

5.2 Le protocole de communication dans un réseau eDonkey

Le protocole eDonkey est composé de messages envoyés asymétriquement ou bien sur **TCP** ou bien sur **UDP**, selon la fonction en question. Cette section se limitera à la partie du protocole de communication dans un réseau eDonkey. Ce protocole est composé de trois axes de communication : *Client à Serveur*, *Serveur à Client* et *Client à Client*.

Par convention, la communication arrive sur **TCP** à travers le port **4662** et les **pings UDP** et d'autres communications arrivent via le port **4666 [52] [104]**. Nous représentons ci-dessous les messages TCP qui circulent entre le serveur et le client eDonkey **[52] [104]**.

5.2.1 Description générale de message

Chaque message eDonkey est préfixé avec un en-tête du contenu suivant (on peut avoir d'autres protocoles par exemple : 0xC5 pour eMule) :

Protocol	Size	Type
0xE3	**4**	**0x01**

Tableau 5.1 : *L'en-tête de message eDonkey*.

5.2.2 Communication TCP client / serveur

Bien que la plus grande proportion de la largeur de la bande sur le réseau eDonkey soit consommée via la transaction Client / Client, la transaction Client / Serveur (et vice versa) est principalement une communication permettant un accès au réseau.

- Au démarrage le client se connecte à un serveur eMule (ou eDonkey) en utilisant une connexion TCP (Annexe : **tableau A.1**). Le serveur assigne le client un *ID*, cela peut être un *LowID* ou *HighID* (plus d'information sur cette propriété dans **[52]**).
- Le client offre au serveur sa liste de fichiers disponibles à partager (avec d'autres clients) via le message *Offer-Files* (Annexe : **tableau A.2**). Chaque fichier a un format bien défini (Annexe : **tableau A.3**). Le Client fait aussi une demande de la liste d'autres serveurs dans le réseau eDonkey (Annexe : **tableau A.4**). Le serveur lui envoie son statut et sa version (Annexe : **tableau A.5**). Ensuite, ce dernier lui répond par une liste de serveurs via le message *List-of-Servers* (tableau 5.2) qui est enregistré dans un fichier *server.met*.

Protocol	Size	Type	Entry Count	Server entries
0Xe3	**4**	**0Xe3**	**00000003**	**Server1 @IP1, num Port1** **Server2 @IP2, num Port2** **Server3 @IP3, num Port3**

Tableau 5.2 : *Un message List-of-Servers*.

- Le client commence sa recherche en envoyant une requête de *Get-source* (Annexe : **tableau A.6**) pour un fichier spécifique qui va être répandu par un message *Found-source* (Annexe : **tableau A.7**) (voir chapitre 2 : figure 2.4) et qui contient la liste de sources (autres clients qui ont ce fichier) : *IP* et *port*.
- La recherche de fichier est initiée par l'utilisateur. L'opération est simple. Une demande de recherche de fichier va être envoyée au serveur (message *Search-request*, Annexe : **tableau A.8**) qui va être répliquée par un message *Search-result* (Annexe : **tableau A.9**).

5.2.3 Communication UDP client / serveur

UDP est employé pour la communication avec des serveurs autre que le serveur dont le client est actuellement connecté. L'objectif des messages UDP est **[52]** :

- d'améliorer la recherche de fichiers.
- d'améliorer la recherche de la source. Lorsque le nombre de sources que se procure le client pour un certain fichier dans sa liste de téléchargement est moins que la limite configurable (100), le client envoie périodiquement des messages *UDP Get-source* aux serveurs de sa liste de serveurs.
- Et finalement, d'effectuer des tests de présence des serveurs (requête de *Keep-alive*) pour s'assurer que tous les serveurs dans la liste de serveurs du client sont valables (*voir la section 2.4.6 de chapitre 2*). Cette vérification est faite en utilisant des requêtes *UDP Server-Status* (tableau 5.3) et *UDP Server-description* (Annexe : **tableau A.10**). Ces dernières, seront répliqués via des messages *Status-response* et *Server-description-response* (Annexe : **tableau A.11** et **tableau A.12**).

Protocol	Type	Challenge
0xE3	**0x96**	**4**

Tableau 5.3 : *Le message Server-status.*

5.3 Propagation de ver sur réseau P2P

Le réseau P2P est un réseau virtuel (overlay) qui est composé des nombreux pairs qui sont distribués sur tout l'Internet. La structure de cela est minutieusement conçue afin de partager les ressources de réseau. Évidemment, le ver peut employer des systèmes P2P comme un véhicule pour la propagation rapide **[105].**

Si un hôte réel appartient à un réseau P2P, il peut attaquer, et ses voisins et les hôtes dans le sous-réseau, comme nous pouvons voir dans la figure 5.1. Il est clair que le réseau P2P accélère la propagation de ver sur tout l'Internet.

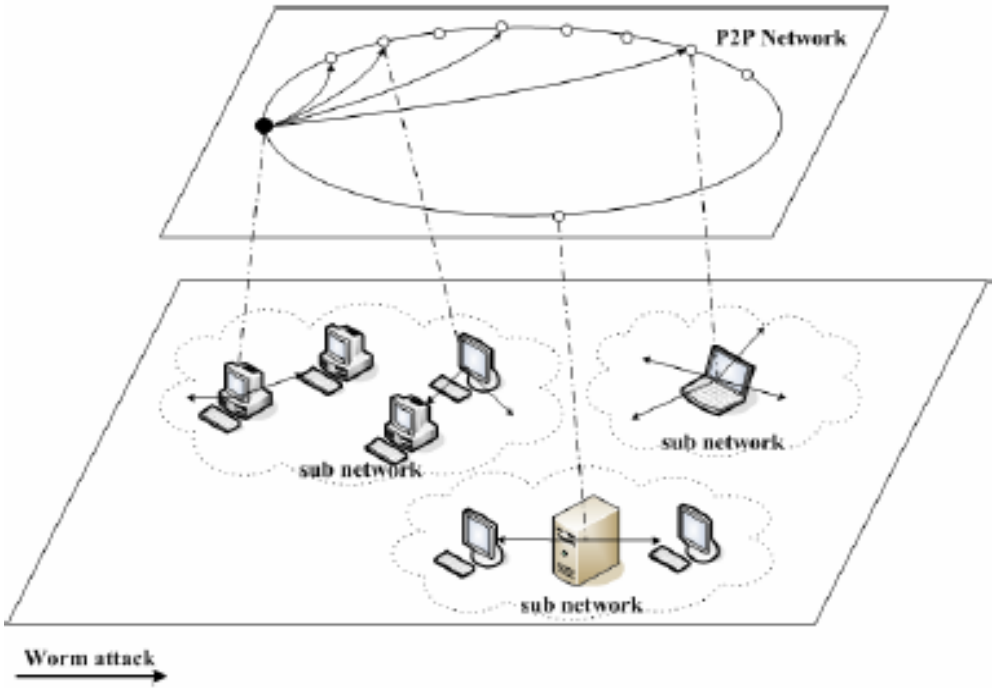


Figure 5.1 : *Propagation de ver sur le réseau P2P*.

Le ver est un composant de logiciel qui, sous ses propres moyens, est capable d'infecter un système d'ordinateur et l'utiliser d'une façon automatisée pour infecter un autre système **[79] [106]**. Ce cycle est alors répété et la population d'hôtes infectés de ver grandit exponentiellement, particulièrement dans un environnement de réseau P2P semi-centralisé (ou hybride).

Dans ces réseaux hybrides, la notion de super-nœud (serveur eMule) a été présentée aux réseaux P2P pour leur meilleure adaptabilité. Des supers-noeuds sont des noeuds avec des ressources suffisantes et des liaisons de haute qualité permettant l'accueil d'un grand nombre de voisins. Ces supers-nœuds (serveurs eMule) jouent sans aucun doute un rôle significatif dans la propagation du ver en raison de leur haute connectivité **[78] [68]**.

Il existe trois stratégies de l'attaque ver. La première est une attaque purement aléatoire (*Random-based Scan*). C'est la stratégie d'attaque fondamentale adoptée par beaucoup de vers. Les deux autres sont des stratégies d'attaques basées sur P2P : l'approche *offline P2P-based Scan* (OPHLS) et l'approche *Online P2P-based Scan* (OPS) **[79]**.

5.4 Proposition d'un modèle de réalisation de l'attaque Eclipse dans un réseau eDonkey

Avant de présenter notre modèle de réalisation de l'attaque Eclipse dans un réseau eDonkey, nous allons présenter ce qu'on entend par une attaque Eclipse. L'attaque Eclipse **[87] [88]** consiste à contrôler plusieurs pairs influents sur le réseau (une grande partie de voisin), pour ne pas acheminer le trafic vers les pairs sains.

L'attaquant éclipse alors les pairs sains sur le réseau. Ainsi, le réseau peut ensuite être partagé en plusieurs sous réseaux. Même si les attaquants ne contrôlent qu'une petite partie du réseau, ils sont capables de lancer une attaque Eclipse.

Par l'analyse de l'architecture eDonkey **[52]** et d'après la définition de l'attaque Eclipse, on constate que la réalisation de l'attaque Eclipse sur un réseau eDonkey se fait entre un serveur et ses clients seulement. Donc, il est difficile d'envisager une propagation entre les serveurs.

Etant donné que la propagation de l'attaque ver est plus efficace dans les réseaux hybrides (voir section 5.3), notre proposition consiste en l'emploi de l'attaque ver pour monter et propager une attaque Eclipse dans un réseau eDonkey.

5.4.1 La description du modèle de réalisation de l'attaque Eclipse dans le réseau eDonkey

Notre modèle exploite les vers dans le réseau eDonkey afin de réussir la propagation de l'attaque Eclipse. Pour cela on a :

5.4.1.1 Hypothèse :

1). Supposant avoir un réseau eDonkey avec la topologie P2P suivante :
   - Un ensemble de serveurs eDonkey ($\mathbf{S_{ed2k}}$) et un ensemble de clients ou peers (**D**). Ils sont respectivement représentés comme suit :
     $\mathbf{S_{ed2k} = \{S_i / i = 1,2,\ldots, n\}}$ et $\mathbf{D = \{C_{i,j} / i = 1,2,\ldots, n}$ et **j varie**$\mathbf{\}}$

Auxquels chaque $\mathbf{S_i} \in \mathbf{S_{ed2k}}$ est responsable et surveille un ensemble de clients $\mathbf{C_{i,j}} \in \mathbf{D}$.

2). Supposant avoir un serveur malicieux nommé « $\mathbf{S_{attaquant}}$ » parmi l'ensemble de serveurs $\mathbf{S_{ed2k}}$ de ce réseau : $\mathbf{S_{attaquant}} \in \mathbf{S_{ed2k}}$ et qui est responsable un ensemble de clients $\mathbf{C_{attaquant,j}} \in \mathbf{D}$ comme le montre la figure 5.2.

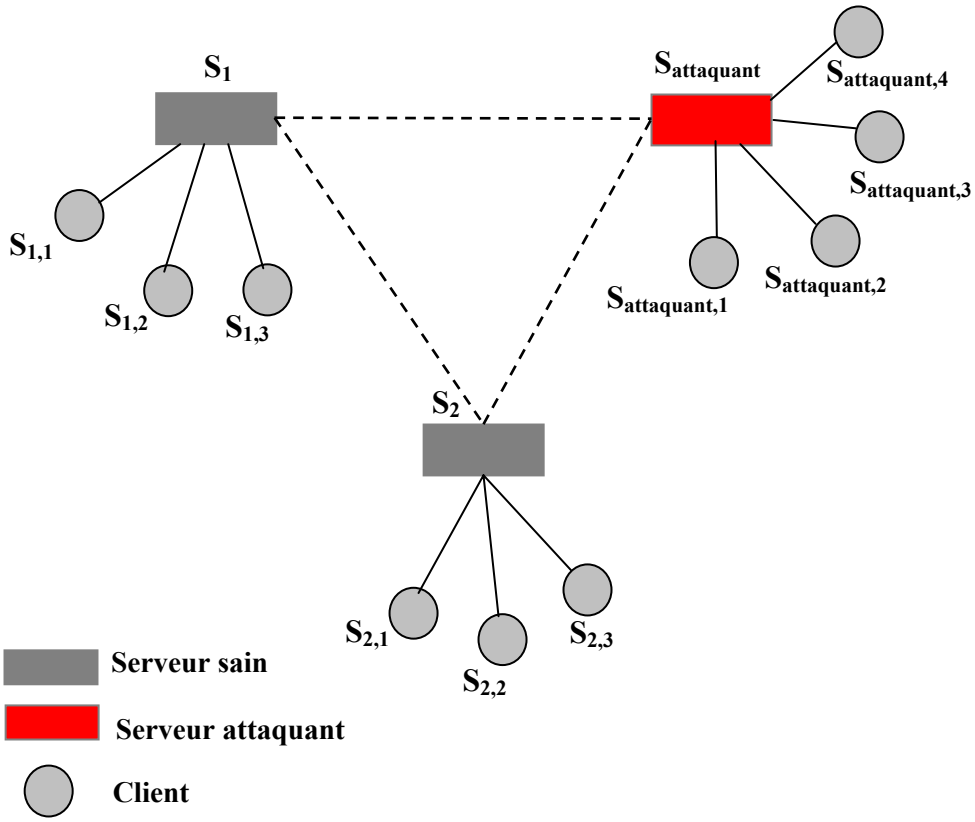


Figure 5.2 : *Un exemple de la topologie de réseau eDonkey avec 3 serveurs où (i= {attaquant, 1,2} et j varie de 1 à 4).*

3). Supposant que le ver peut se propager d'un serveur à un autre serveur du réseau en exploitant la vulnérabilité de l'architecture hérité (sous-jacente).

5.4.1.2 Fonctionnement de base :

1). Dans ce réseau, les clients $\mathbf{C_{i,j}}$ vont établir un handshake (voir section 5.2.2) avec les serveurs $\mathbf{S_i}$ auxquels ils sont liés directement (connexion TCP).

2). Lors de l'établissement de connexion, les clients $\mathbf{C_{i,\ j}}$ **/ i = {attaquant, 1,2,…, n}** demandent respectivement à leurs serveurs $\mathbf{S_i}$ la liste de serveurs pour effectuer une mise à jour à leurs listes de serveurs.

3). Après le handshake initial, une transaction de recherche s'effectue entre les clients $\mathbf{C_{i,\ j} \in D}$ et leurs serveurs $\mathbf{S_i \in S_{ed2k}}$ auxquels ils sont liés directement. Généralement, cette transaction va être suivie par une requête de recherche de sources d'un fichier spécifique (@IP, N°port) à partir desquels le client demandeur peut télécharger ce fichier.

4). Afin de tester la présence des serveurs $\mathbf{S_i}$ dans la liste de serveurs des clients $\mathbf{C_{i,\ j}}$ (dans le fichier *server.met*), les $\mathbf{C_{i,\ j}}$ vont envoyer des requêtes UDP (*UDP server-status* et *UDP description-server*) à chaque serveur $\mathbf{S_i}$ de la liste de serveurs qu'ils ont pour vérifier l'état des serveurs. A ce moment, les compteurs de tentative pour chaque serveur $\mathbf{S_i}$ commence à s'incrémenter. Si les clients $\mathbf{C_{i,\ j}}$ reçoivent des réponses UDP de la part de ces serveurs $\mathbf{S_i}$ alors les compteurs de tentative vont être réinitialisé, sinon, ces derniers vont atteindre leur limite configurable et par conséquent les serveurs $\mathbf{S_i}$ vont être supprimés de la liste de serveurs des clients $\mathbf{C_{i,\ j}}$.

5.4.1.3 Déroulement de l'attaque et sa propagation

Dans notre modèle de propagation de l'attaque Eclipse, l'attaque est menée par un seul serveur attaquant ($\mathbf{S_{attaquant}}$). Son but est de cacher la partie du réseau qu'il surveille ($\mathbf{C_{attaquant,\ j}}$) du reste du réseau. Nous allons bénéficier de l'attaque ver pour réussir à propager l'attaque Eclipse en cachant (éclipsant) d'autres parties du réseau ($\mathbf{C_{i,\ j}}$ / **i ≠ attaquant**) qui sont surveillées par autres serveurs ($\mathbf{S_i}$ **/ i ≠ attaquant**). Et finalement, subdiviser le réseau eDonkey en différents sous réseaux. Cela peut être réalisé de la manière suivante :

1). Le serveur attaquant dénoté par : $\mathbf{S_{attaquant}}$ accepte les nouveaux clients mais il leur envoie une liste de serveurs qui ne contient que son adresse : $\mathbf{S_{attaquant}}$**@IP**, $\mathbf{S_{attaquant}}$**N°port**, dans la message *List-of-servers* (tableau 5.4).

Protocol	Size	Type	Entry count	Server entries
0Xe3	4	0x32	000000001	$S_{attaquant}$@IP, $S_{attaquant}$N°port

Tableau 5.4 : *Message List-of-servers envoyé par le serveur attaquant.*

2). Le serveur $S_{attaquant}$ va répondre négativement aux requêtes de recherche de ses clients $C_{attaquant, j}$ (tableau 5.5 et 5.6) ou donner de mauvaises réponses comme par exemple : leur fournir des adresses aléatoires de clients (réacheminer la recherche) comme des sources trouvées (figure 5.3).

Protocol	Size	Type	Result count	Result list
0Xe3	4	0x16	0	Vide

Tableau 5.5 : *Exemple de message négatif « Search-result » envoyé par le serveur attaquant.*

Protocol	Size	Type	File hash	Source count	List of sources
0Xe3	4	0x42	NA	000000000	Vide

Tableau 5.6 : *Exemple de message négatif « Found-sources » envoyé par le serveur attaquant.*

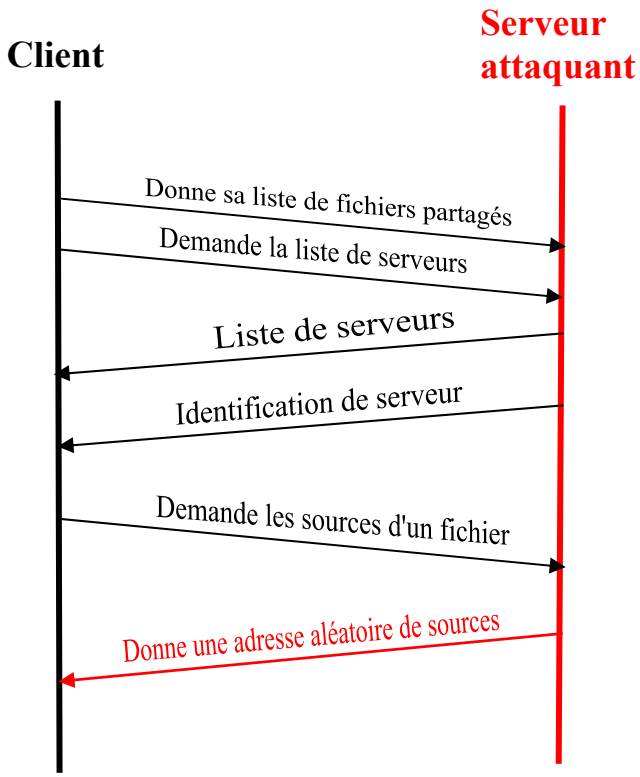


Figure 5.3 : *Le serveur attaquant donne une adresse aléatoire de source (client).*

3). Le serveur $\mathbf{S_{attaquant}}$ va ignorer toutes requêtes extérieures qui ne viennent pas de ses clients $\mathbf{C_{attaquant,\ j}}$ (requêtes UDP). Sauf pour les requêtes : *UDP Server-status* et *UDP Server-description* afin de veiller sur sa notoriété dans tout le réseau (figure 5.4).

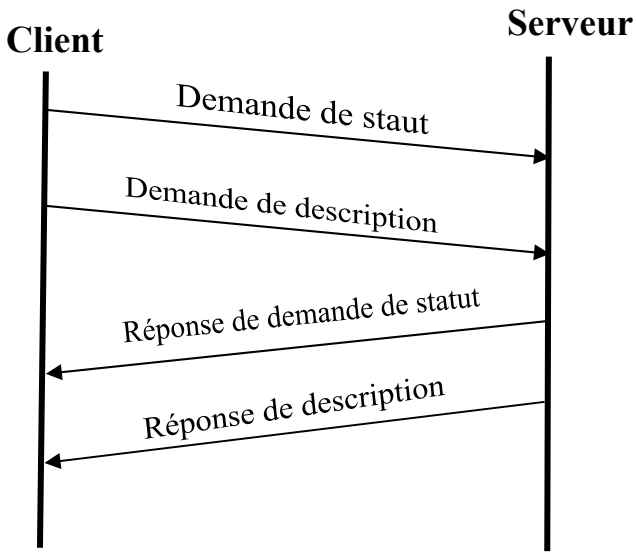


Figure 5.4 : *Séquence UDP de test de présence de serveur attaquant.*

4). Le ver peut opérer à deux niveaux différents :

- *Au niveau clients :*

  Le ver va se propager aux clients $\mathbf{C_{attaquant,\ j}}$ du serveur malicieux $\mathbf{S_{attaquant}}$ à travers la liste des clients que possède le serveur $\mathbf{S_{attaquant}}$. Ensuite, il va fermer le port UDP des clients $\mathbf{C_{attaquant,\ j}}$. Ceci va empêcher la réception des messages UDP depuis les autres serveurs $\mathbf{S_i}$ (de leurs listes de serveur) et particulièrement les messages : *UDP Server-status* et *UDP Server-description*. Cependant, les compteurs de tentative pour chaque serveur $\mathbf{S_i}$ s'incrémentent jusqu'à ce qu'ils atteignent leur limite configurable. Les clients $\mathbf{C_{attaquant,\ j}}$ vont supprimer les serveurs $\mathbf{S_i}$ (distants) de leurs listes de serveurs. Par conséquent, les clients ne vont avoir dans leurs listes de serveurs que l'adresse du serveur $\mathbf{S_{attaquant}}$ auxquels ils sont liés directement (connexion TCP) et ne peuvent donc pas communiquer avec le reste de réseau. Ainsi, la partie dont il surveille le $\mathbf{S_{attaquant}}$ est isolée du reste du réseau eDonkey.

- *Au niveau serveurs :*

  Le ver exploite la vulnérabilité de l'architecture héritée afin de contaminer les autres serveurs (sains) du réseaux eDonkey $\mathbf{S_i \in S / i \neq attaquant}$, ainsi que leurs clients $\mathbf{C_{i,\ j} \in D}$ comme décrit ci-dessus. Le ver supprime les serveurs $\mathbf{s_i}$ sains des listes de serveurs des clients $\mathbf{C_{i,\ j}}$ contaminés.

5). Par conséquent, chaque serveur du réseau eDonkey $\mathbf{S_i \in S / i = \{attaquant, 1, 2, 3 \ldots, n\}}$ ainsi que la partie qu'il surveille $\mathbf{C_{i,\ j} \in D}$ vont être isolés du reste de réseau. Ainsi, on aura des sous réseaux $(\mathbf{S_i, C_{i,\ j}})$ qui ne communiquent pas entre eux, par exemple (voir figure 5.2) : $(\mathbf{S_{attaquant}, C_{attaquant,\ j}})$, $(\mathbf{S_1, C_{1,\ j}})$, $(\mathbf{S_2, C_{2,\ j}})$ et j varie entre {1, 4}.

5.5 Conclusion

Dans ce chapitre nous avons vu que le ver est facile de se propager dans les réseaux hybrides, comme eDonkey. Du fait que les supers-nœuds jouent sans aucun doute un rôle significatif dans la propagation du ver en raison de leur haute connectivité. Il existe plusieurs manières pour que le ver se propage dans le réseau (*email, message, partage de fichier, vulnérabilité,…*).

Après une étude sur l'attaque Eclipse et l'architecture eDonkey, on réalise que la propagation de cette attaque se fait entre un serveur et ses clients seulement. Nous avons proposé un modèle de réalisation de l'attaque Eclipse dans un réseau eDonkey.

Notre modèle vise la liste de serveurs des clients (ou peers) afin d'isoler ces derniers du reste du réseau eDonkey empêchant tout contact avec l'extérieur et envisager une propagation entre les serveurs. Pour cela, nous avons eu recours à l'attaque ver. Ce dernier, va opérer à deux niveaux différents : A*u niveau clients*, le ver va fermer le port UDP des clients, ce qui déclenche la mise à jour du compteur de tentative des serveurs et après un certain moment le client va supprimer tous ces serveurs. *Au niveau serveurs*, le ver va exploiter la vulnérabilité de l'architecture héritée afin de contaminer les autres serveurs ainsi que leurs clients. Et finalement, parvenir à monter et propager l'attaque Eclipse en divisant le réseau eDonkey à un ensemble de sous réseaux isolés. Même dans le cas où certains serveurs n'ont pas été contaminés, l'ensemble des serveurs contaminés ainsi que leurs clients respectifs constituent des sous réseaux gérés par l'attaquant.

CONCLUSION GÉNÉRALE ET PERSPECTIVES

La sécurité n'a jamais été le point fort de l'Internet. Les systèmes P2P doivent faire face au défi de la sécurité. Sans doute l'action la plus sûre est de ne pas employer le logiciel P2P sachant que son architecture est grande ouverte. Cependant, et étant donné la croissance de l'échelonnement du contenu disponible sur l'Internet et la population énorme des utilisateurs, on ne peut ignorer ce grand marché. Si P2P est employé, ses dangers "doivent être au minimum réduits par l'utilisation rigoureuse de : la sauvegarde, le logiciel anti-viral/Trojan et la configuration prudente du logiciel P2P afin d'assurer qu'il fait exactement ce que l'utilisateur veut qu'il fasse.

Les P2P actuels souffrent de nombreux problèmes tels que les fichiers truqués, virus et autres… A cela, s'ajoute le fait qu'il ne s'agit pas, pour la plupart du temps, de P2P « purs », mais de P2P « hybrides », s'appuyant donc sur l'utilisation de serveurs centraux qui, malgré leurs avantages, constituent une véritable faiblesse, comme le réseau eDonkey.

Nous avons présenté dans les chapitres précédents, ce qu'on entend par un modèle pair-à-pair et les différents protocoles les plus utilisés aujourd'hui, à savoir eDonkey et Bittorent. Et après avoir vu une synthèse des différentes attaques qui peuvent être menées sur ce type de réseau en se focalisant sur l'attaque Eclipse, nous avons constaté, que les réseaux hybrides sont plus vulnérables à ce type d'attaques, en raison de la meilleure connectivité de leurs super-nœuds ainsi la propagation de l'attaque est plus rapide.

Notre travail consistait à proposer un modèle de réalisation de l'attaque Eclipse dans un réseau eDonkey. Il s'agit de réaliser l'attaque Eclipse (entre un serveur et ses clients) et de la propager (entre les serveurs) dans ce réseau hybride. Pour cela, nous avons utilisé l'attaque Ver qui va exploiter la vulnérabilité de l'architecture héritée afin de se propager et contaminer les serveurs ainsi que leurs clients. Par la suite, il va supprimer les serveurs sains de la liste de serveurs des clients. Ainsi donc, interdire la communication entre les clients distants (qui n'on pas le même serveur). Par conséquent, le réseau eDonkey sera subdivisé en sous-réseaux isolés. Même dans le cas où certains serveurs n'ont pas été contaminés, l'ensemble des serveurs contaminés ainsi que leurs clients respectifs constituent des sous-réseaux gérés par l'attaquant.

Comme perspectives, nous envisageons dans le futur, de réaliser un simulateur muni d'une plate forme d'un réseau eDonkey qui va nous permettre de simuler notre modèle et analyser l'impact de la propagation de l'attaque Eclipse dans ce réseau en terme de temps et de coût. De s'étendre à d'autres types d'attaques qui pourront faire l'objet d'un sujet d'étude afin de concevoir une architecture de sécurité plus résistante pour le réseau eDonkey.

Bibliographie

[1] B. Pourebrahimi, K. Bertels, S. Vassiliadis. *"A Survey of Peer-to-Peer Networks"*. Technical Report, Computer Engineering Laboratory, ITS, TU Delft, the Netherlands, 2005.

[2] S. Androutsellis-Theotokis and D. Spinellis. *"A Survey of Peer-to-Peer Content Distribution Technologies"*. ACM Computing Surveys, December 2004.

[3] R. Schollmeier. *"A definition of Peer to Peer Networking for the classification of Peer to Peer Architectures and Applications"*. IEEE Computer Society. 2001.

[4] Y. Liu, L. Xiao, X. Liu, L. M. Ni and X. Zhang. *"Location awareness in unstructured Peer-To-Peer systems"*. IEEE Computer Society, February 2005.

[5] C. Shirky. *"Peer-to-Peer: harnessing the power of disruptive technologies, chapter listening to Napster"*, page 21-37. O'Reilly & Associates, Inc, 2001.

[6] M. Miller. *"Discovering Peer-to-Peer Electronic Book"*. Technical Report, San Francisco.CA. SYBEX, 2001.

[7] F. Picconi, J-Michel Busca and P. Sens. "*Exploiting Network Locality in a Decentralized Read-write Peer-to-peer File System"*. ACM. 2004.

[8] "*The Gnutella Protocol Specification V0.4*". http://dss.clip2.com/GnutellaProtocol04.pdf. 2001.

[9] D. Bickson and D. Malkhi. *"A Study of Privacy in File Sharing Networks"*. In Technical Report TR-2003-67 Leibniz Research Center, the Hebrew University of Jerusalem, Israel, 2003.

[10] D. Doval and D. O'Mahony. "*Overlays networks: A scalable alternative for P2P"*. IEEE Internet Computing, 2003.

[11] F. Depaoli, L. Mariani. "*Dependability in Peer to Peer systems"*. IEEE Computer Society. 2004.

[12] G. Feltin, G. Doyen, O. Festor. ''*Les protocoles Peer-to-Peer, leur utilisation et leur détection*''. Technical Report, JRES2003, LORIA - Campus Scientifique-NANCY. 2003.

[13] K. Nagaraja, J. Pruyne, B. Richard, S. Rollins and Z. Xu. "*Peer-to-Peer Computing"*. Technical Report, HP Laboratories Palo Alto, 2003.

[14] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. "*SETI@home: an experiment in public-resource computing"*. Communications of the ACM 2002.

[15] "*XtremWeb*". http:// www.lri.fr/~fedak/Xtremweb/introduction-php3.

[16] S. Saroiu, P. K. Gummadi, S. D. Gribble. "*Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts"*. IEEE Computer Society, October 2002.

[17] I. Ivkovic. "*Improving Gnutella Protocol: Protocol Analysis And Research Proposals"*. IEEE Computer Society, 2001.

[18] D. Tsoumakos and N. Roussopoulos. "*Analysis and comparison of P2P search methods"*. Technical Report CS-TR-4539, UMIACS-TR-2003-107, University of Maryland – Departement of Compuster Science, 2003.

[19] I. Clarke, O. Sandberg, B. Wiley and T. W. Hong. "*Freenet: A Distributed Anonymous Information Storage and Retrieval System "*. IEEE Computer Society, 2001.

[21] K. Aberer and M. Hauswirth. *"Peer-to-peer information systems: concepts and models, state-of-the-art, and future systems"*. ACM Comunication,2002.

[22] S. Androutsellis-Theotokis and D. Spinellis. *"A Survey of peer to peer file sharing technologies"*. Technical Report, Electornic Trading Research Unit (ELTRUN), Athens University fo Economics and Business, 2002.

[23] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. "*Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications"*. Proc. ACM SIGCOMM Conference, 2001.

[24] A. Rowstron and P. Druschel. "*Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems"*. Proc.Int'l Conf. Distributed Systems Platforms. Journal: Lecture Notes in Computer Science, 2001.

[25] B.Y. Zhao, J.D. Kubiatowicz, and A.D. Joseph. "*Tapestry: An Infrastructure for Fault-Resilient Wide-Area Location and Routing"*. Technical Report UCB//CSD-01-1141, Univ. of California,Berkeley, 2001

[26] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. *"A Scalable Content-Addressable Network"*. Proc. ACM SIGCOMM, 2001.

[27] G. Manku and M. Bawa and P. Raghavan. "*Symphony: Distributed hashing in a small world"*. Proc. of 4th USENIX Symposium on Internet Technology and Systems (USITS'03), Technical Report, 2003.

[28] M. Kelaskar, V. Matossian, P. Mehra, D. Paul, and M. Parashar. *"A Study of Discovery Mechanisms for Peer-to-Peer Applications"*. Proceedings of the 2nd IEEE/ACM, 2002.

[29] X. Li and J. Wu. "*Searching Techniques in Peer-to-Peer Networks"*. Department of Computer Science and Engineering. Florida Atlantic University, 2005.

[30] D Tsoumakos and N Roussopoulos. "*A Comparison of PeertoPeer Search Methods"*. In Proceedings of the Sixth International Workshop on the Web, 2003.

[31] S. Frénot. ‘‘*Gestion du déploiement de composants sur Réseau P2P’’*. In *Décor*, Conférence Grenoble, France, October 2004.

[32] N. Lumineau. ‘‘*Organisation et Localisation de Données Hétrogènes et Répartie sur un Réseau Pair-à-Pair’’*. Thèse de doctorat, Université Pierre et Marie Curie, 7 quai Saint Bernard, Paris 2005.

[33] J. Ritter. *”Why Gnutella Can’t Scale. No Really”*. Technical Report, 2001.

[34] B. Yang, H. Gracia-Molina. *”Improving search in Peer-to-Peer Networks*”. ACM, 2002.

[35] Q. Lv, P. Cao, E. Cohen, K.Li and S. Shenker . *”Search and Replication in Unstructured Peer to Peer Networks”*. Proc. International Conference on Supercomputing, ACM, 2002.

[36] S. Voulgaris, A. Kermarrec, L. Massouli\'e and M. van Steen. ‘‘*Exploiting Semantic Proximity in Peer-to-Peer Content Searching’’*. IEEE Computer Society, 2004.

[37] C. Gkantsidis, M. Mihail and A. Saberi. *“Random Walks in Peer-to-Peer Networks”*. In Proceedings of IEEE INFOCOM, 2004.

[38] A. Crespo and H. Garcia-Molina. *“Routing Indices For Peer-to-Peer Systems”*. Proc. International Conference on Distributed Computing Systems, ACM, 2002.

[39] K. Sripanidkulchai, B. Maggs and H. Zhang. *“Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems”*. INFOCOM of the IEEE Computer and Communications Societies, 2003.

[40] K. Aberer. *“P-Grid : A Self-Organizing Access Structure for P2P-Information systems*”. Sixth International Conference on Cooperative Information Systems (CoopIS), 2001.

[41] P. Maymounkov and D. Mazieres. *“Kademlia: A Peer-to-peer Information System Based on the XOR Metric”*. In Proceedings of IPTPS02, Cambridge, USA, March, Technical Report, 2002.

[42] S. El-Ansary and S. Haridi. *“An Overview of Structured P2P Overlay Networks”*. SWEDISH Institue of Computer Science SICE publication. Technical Report, July 2004.

[43] D. Platt, Majoras. *“Peer-to-Peer File-Sharing Technology: Consumer Protectionand Competition Issues”*. Technical Report, 2005.

[44] J. Liang and R. Kumar and K. Ross. *“Understanding KaZaA ”*. Technical Reports, 2003.

[45] K. Kutzner and T. Fuhrmann. *“Measuring Large Overlay Networks: The Overnet Example*”. System Architecture Group, Universitat Karlsruhe (TH), Technical Report, 2005.

[46] K. Tutschku. "*A Measurement-based Traffic Profile of the eDonkey File sharing Service*". Technical Report, 2004.

[47] J.A. Pouwelse, P. Garbacki, D.H.J. Epema, H.J. Sips. "*The BitTorrent P2P File-Sharing System: Measurments and Analysis"*. Delft University of Technology, the Netherlands, Technical Report, 2005.

[48] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma and S. Lim. "*A Survey and Comparison of Peer-to-Peer Overlay Network Schemes"*. IEEE communication survey and tutorial, march 2004.

[49] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica. "*Looking up Data in P2P Systems"*. ACM 2003.

[50] M. Bergner. "*Improving performance of modern peer-to-peer services"*. UMEA University, Department of Computer Science, Technical Report, 2003.

[51] J. Kurose, K. Ross. ''*Computer Networking: A Top-Down Approach Featuring the Internet"*. Book 3/E, Addison-Wesley, 2002.

[52] Y. Kulbak and D. Bickson. "*The eMule Protocol Specification"*. Technical report TR-2005-03, the Hebrew University of Jerusalem, 2005.

[53] R. Thommes, M. Coates. "*BitTorrent Fairness: Analysis and Improvements"*.In: Proc. Workshop Internet, Telecom. and Signal Proc., December 2005.

[54] J. A. Johnsen, L. E. Karlsen and S. S. Birkeland. "*Peer-to-peer networking with BitTorrent"*. Technical Reports, Department of Telematics, NTNU - December 2005.

[55] D. Erman, D. Ilie and A. Popescu. "*BitTorrent Session Characteristics and Models*". In Demetres Kouvatsos, editor, Proceedings of the 3rd International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks, 2003.

[56] A. Hindle. "*CSC523: Analysis of the P2P BitTorrent Protocol*". Technical Report. April, 2004.

[57] W. Stallings, "*Cryptography and Network Security: principle and practice"*. 3rd edition: Prentice hall, 2003.

[58] J. Risson and T. Moors. ''*Survey of search towards robust peer-to-peer networks: search methods''*. ACM, 2004.

[59] N. Daswani, H. Garcia-Molina and B. Yang. "*Open Problems in Data-Sharing Peer-to-Peer systems"*, Proceedings of the 9th International Conference on Database Theory, 2003.

[60] Y. Zhang, X. Li and J. Huai. *''Access Control in Peer-to-Peer Collaborative Systems''*. Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'05), 2005.

[61] T. Chothia. *''Analysing the MUTE Anonymous File-Sharing System Using the Pi-Calculus''*. Amsterdam, the Netherlands, Technical Report, 2006.

[62] T. Chothia and K. Chatzikokolakis. *"A Survey of Anonymous Peer-to-Peer File-Sharing"*. Laboratoire d'Informatique, ´Ecole Polytechnique, France. Technical Reports, 2005.

[63] E. Adar and B. Huberman. "*Free Riding on Gnutella"*. IEEE, 2000.

[64] D. S. Wallach. *"A Survey of Peer-to-Peer Security Issues"*. Rice University, Houston, TX 77005, USA, Technical Report, 2002.

[65] A. Mauthe and D. Hutchison. "*Peer-to-Peer Computing: Systems, Concepts and Characteristics"*. Computing Department Lancaster University, Special Issue on Peer-to-Peer , Technical Report, 2003.

[66] M. Parashar, M. Agarwal, S. Arbeeny, V. Bhat and R. Chowdhury. "*Evaluating Security Mechanisms in Peer-to-Peer Applications"*. Department of Electrical and Computer Engineering, Rutgers University, USA, Technical Report, 2001.

[67] P. Kohli and U. Ganugula. "*DDoS Attacks using P2P Networks"*. Centre for Security Theory and Algorithmic Research International Institute of Information Technology Hyderabad, India, Technical Report, 2007.

[68] M. Engle and J. I. Khan. " *Vulnerabilities of P2P Systems and a Critical Look at Their Solutions"*. Internetworking and Media Communications Research Laboratories, Department of Computer Science, Kent State University, Technical Report 2006.

[69] J. V. Vroonhoven. "*Peer to Peer Security"*. Technical Report, University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science, 2006.

[70] E. Sit and R. Morris. "*Security Considerations for Peer-to-Peer Distributed Hash Tables"*. Technical Reports 2002.

[71] X. Lou and K. Hwang**. "***Prevention of Index-Poisoning DDoS Attacks in Peer-to-Peer File-Sharing Networks"*. Manuscript submitted to IEEE Transactions on Multimedia, special issue on Content Storage and Delivery in P2P Networks, *Nov.9,* 2006.

[72] L. Wang. "*Attacks Against Peer-to-peer Networks and Countermeasures"*. Technical Report, Helsinki University of Technology, 2006.

[73] B. Prêtre. "*Attacks on Peer-to-Peer Networks*". Semester Thesis, Dept. of Computer Science Swiss Federal Institute of Technology (ETH) Zurich, Automne 2005.

[74] S.D. Koolen, *"A Review of Social P2P Overlay Networks"*. Thesis, Parallel and Distributed Systems group Faculty of Electrical Engineering, Mathematics, and Computer Science Delft University of Technology, 24th November 2005.

[75] D. Dumitriu, E. Knightly, A. Kuzmanovic, I. Stoica, and W. Zwaenepoel. "*Denial of Service Resilience in PeertoPeer File Sharing Systems"*. In Proc of ACM Sigmetrics Conference, 2005.

[76] C. Zou, W. Gong and D. Towsley. "*Code Red Worm Propagation Modeling and Analysis*". In Proceedings of the 9th ACM Conference on Computer and Communications Security, pages 138--147, November 2002.

[77] V. Vlachos, S. Androutsellis-Theotokis and D. Spinellis. "*Security applications of peer-to-peer networks*". Computer Networks: The International Journal of Computer and Telecommunications Networking, ACM, 2004.

[78] L. Zhou, L. Zhang, F. McSherry, N. Immorlica, M. Costa and S. Chien. *"A First Look at Peer-to-Peer Worms: Threats and Defenses"*. IEEE 2006.

[79] W. Yu, C. Boyer, S. Chellappan and D. Xuan. "*Peer-to-Peer System-based Active Worm Attacks: Modeling and Analysis"*. IEEE 2006.

[80] L. Cox and B. Noble. "*Samsara: Honor Among Thieves in Peer-to-Peer Storage"*. In Proceedings of the ACM Symposium on Operating Systems Principles, October 2003.

[81] J. Douceur. "*The sybil attack*". In Proc. ACM, 2002.

[82] J. Dinger and H. Hartenstein. "*Defending the Sybil Attack in P2P Networks: Taxonomy, Challenges, and a Proposal for Self-Registration"*. IEEE 2005.

[83] B. N. Levine, C. Shields and N. B. Margolin. "*A Survey of Solutions to the Sybil Attack"*. IEEE 2006.

[84] M. Castro, P. Druschel, A. Ganesch, A. Rowstron, D.S. Wallach. "*Secure routing for structured peer-to-peer overlay networks*". ACM SIGOPS Operating Systems Review, 2002.

[85] N. Borisov. "*Computational Puzzles as Sybil Defenses*". Technical Report, Department of Electrical & Computer Engineering University of Illinois at Urbana-Champaign, 2006.

[86] H. Yu, M. Kaminsky, P.B. Gibbons and A. Flaxman. "*SybilGuard: Defending Against Sybil Attacks via Social Networks"*. In Proc. of the ACM *SIGCOMM'06*, September 11–15, 2006.

[87] A. Singh, M. Castro, P. Druschel and A. Row. "*Defending against eclipse attacks on overlay networks"*. In Proc. of the ACM SIGOPS European Workshop September 2004.

[88] Singh, A. Ngan, T.-W. Druschel, P. Wallach, D. S. "*Eclipse Attacks on Overlay Networks: Threats and Defenses"*. IEEE International Conference on Computer Communications, 2006.

[89] S. Nielson and S. Crosby and D. Wallach. "*A Taxonomy of Rational Attacks"*. In Proc.of IPTPS, Ithaca, NY. February 2005.

[90] L. Wang. "*Attacks Against Peer-to-peer Networks and Countermeasures"*. Technical Report, Seminar on Network Security, 2006.

[91] J. Liang, R. Kumar, Y. Xi and K. W. Ross. "*Pollution in P2P File Sharing Systems"*. IEEE 2005.

[92] N. C. Andreas, S. Weigend and J. Chuang. "*Content Availability, Pollution and Poisoning in File Sharing PeertoPeer Networks"*. In the Proceedings of ACM, 2005.

[93] N. Naoumov and K. Ross. "*Exploiting P2P Systems for DDoS Attacks"*. In Proceedings of the ACM conference on Scalable information systems, 2006**.**

[94] S. B. Handurukande, A.-M. Kermarrec, F. Le Fessant, L. Massoulie, and S. Patarin. ''*Peer Sharing Behaviour in the eDonkey Network, and Implications for the Design of Server-less File Sharing Systems''*. In Proceedings of the ACM SIGOPS EuroSys Conference, 2006.

[95] C.Soldani. "*Peer-to-Peer Behaviour Detection by TCP Flows Analysis"*. Technical Report, University of Liége, 2004.

[96] T. Hobfeld, K. Leibnitz, R. Pries, K. Tutschku, P. Tran-Gia, and K. Pawlikowski. "*Information diffusion in edonkey-like p2p networks"*. In Australian Telecommunication Networks and Applications Conference (ATNAC), Bondi Beach, Australia, 2004.

[97] J. Kim, A. Biryukov, B. Preneel and S. Lee. "*On the Security of Encryption Modes of MD4, MD5 and HAVAL*". Technical Report, Korea University, 2006.

[98] *Emule* : http://www.emule-project.net/.

[99] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang. "*A performance study of bittorrent like peer-to-peer systems"*. IEEE Journal on selected areas in communication, 25(1), JANUARY 2007.

[100] M. Locasto, J. Parekh, A. Keromytis and S. Stolfo. "*Towards collaborative security and p2p intrusion detection"*. IEEE Information Assurance Workshop, 2005.

[101] W. Diffie and M. Hellman. "New Directions in Cryptography". IEEE Trans. Info. Theory 22(6), pp644–654 (1976)

[102] R. Rivest, A. Shamir, and L.M. Adelman. "*A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*". Communication ACM, 1978.

[103] T. Reidemeister, K. Bohm, E. Buchmann and P. A.S. Ward. "*Man-in-the-middle attacks in distributed hash-tables"*. Technical Reports, 2005.

[104] A. O'Connor, C. Brady, P. Byrne and A. Olivré. "*Characterising the eDonkey Peer-to-Peer File Sharing Network"*. Technical Reports, Networks and Distributed Systems, Computer Science Department, Trinity College Dublin, Ireland, 2005.

[105] Y. Zhou, Z. Wu, F. Li, H. Wang and Z. Zhu. "*Analyzing Impacts of Software Diversity on Worm Propagation in Peer-to-Peer Networks"*. International Journal of Computer Science and Network Security. February 2006

[106] M. Collins, C. Gates and G. Kataria. "*A model for opportunistic network exploits : The case of p2p worms"*. Technical reports, National Science Foundation, 2006.

[107] C. Wang and B. Li. "*Peer-to-Peer Overlay Networks: A Survey*". Technical reports, 2003

ANNEXE A

Dans cette annexe, nous allons présenter les différents format des messages transmis entre les pairs (client / serveur) du réseau eDonkey cités dans ce mémoire. Pour plus de détaille, voir l'article « The eMule Protocol Specification **[55]** ».

A.1 Message Login

Le message d'établissement de la connexion est le premier message envoyé par le client au serveur après l'établissement d'une connexion TCP. La longueur de message varie et il dépend de la configuration d'utilisateur comme le surnom d'utilisateur.

Name	Size in bytes	Default Value	Comment
Protocol	1	0xE3	
Size	4		The size of the message in bytes not including the header and size fields
Type	1	0x01	The value of the OP_LOGINREQUEST op-code
User Hash	16		Details about user hash can be found in section 1.4
Client ID	4	0	The client ID that is sent on first connection is usually zero. Details about client ID can be found in section 1.3
TCP Port	2	4662	The TCP port used by the client, configurable
Tag Count	4	4	The number of tags following in the message
Name Tag	varies	NA	The user's nickname (configurable in the software). The tag is a string tag and the tag name is an integer of value 0x1
Version Tag	8	0x3C	The eDonkey version supported by the client. The tag is an integer tag and the tag name is an integer of value 0x11
Port Tag	8	4662	The TCP port used by the client. The tag is an integer tag and the tag name is an integer of value 0x0F
Flags Tag	8	0x01	The tag is an integer tag and the tag name is an integer of value 0x20

Tableau A.1 : *Message Login.*

A.2 Message Offer-Files

Ce message est utilisé par le client afin de décrire les fichiers qui sont disponibles localement dans le but de les rendre accessible pour le téléchargement. Le message est envoyé juste après l'établissement de connexion. Il a une autre utilisation, tester la présence du client en envoyant périodiquement.

Name	Size in bytes	Default Value	Comment
Protocol	1	0xE3	
Size	4		The size of the message in bytes not including the header and size fields
Type	1	0x15	The value of the OP_OFFERFILES opcode
File Count	4	NA	The number of files described within. in any case no more than 200. The Server can also set a lower limit to this number
Files	varies	NA	An optional list of files, the format of a single entry is described below.

Tableau A.2 : *Message Offer-Files.*

A.3 Format d'une entrée de la liste de fichier

Le tableau suivant décrit une entrée de la liste des fichiers. Plusieurs entrées peuvent être annoncé sous forme d'une série dans le même message Offer-Files.

Name	Size in bytes	Default Value	Comment
Hash Code	16	NA	The result of a hash (specification TBD) performed on the file contents. The hash is used to uniquely identify files, ignoring name differences between clients
Client ID	4	NA	The client ID in case the client has high ID, or zero otherwise
Client Port	2	0x15	The Client's TCP port or zero in case the client has low ID
Tag Count	4	NA	The number of tags following this field
File Name Tag	varies	NA	The filename (Mandatory). The tag is a string tag the tag name is an integer of value 0x1
File Size Tag	8	NA	The file size in bytes (Mandatory). The tag is an integer tag and the tag name is an integer of value 0x2

Tableau A.3 : *Format d'une entrée de la liste de fichier.*

A.4 Message Get-list-of-servers

Ce message est envoyé du client au serveur, immédiatement après le handshake initial afin de mettre à jour sa liste de serveurs.

Name	Size in bytes	Default Value	Comment
Protocol	1	0xE3	
Size	4		The size of the message in bytes not including the header and size fields
Type	1	0x14	The value of the OP_SERVERSTATUS opcode

Tableau A.4: *Message Get-list-of-servers.*

A.5 Message Server-Status

Ce message est envoyé du serveur au client. Le message contient l'information sur le nombre actuel d'utilisateurs et des fichiers sur le serveur.

Name	Size in bytes	Default Value	Comment
Protocol	1	0xE3	
Size	4		The size of the message in bytes not including the header and size fields
Type	1	0x34	The value of the OP_SERVERSTATUS opcode
User Count	4	NA	The number of users currently logged in to the server.
File Count	4	NA	The number of files that this server is informed about

Tableau A.5 : *Message Server-Status.*

A.6 Message Get-source

Ce message est envoyé du client au serveur pour demander une source (autres clients) pour un fichier. La taille de ce message est de 22 octets.

Name	Size in bytes	Default Value	Comment
Protocol	1	0xE3	
Size	4		The size of the message in bytes not including the header and size fields
Type	1	0x19	The value of the OP_GETSOURCES opcode
File hash	16	NA	The requested file hash

Tableau A.6 : *Message Get-source.*

A.7 Message Found-source

C'est un message envoyé du serveur au client avec les sources (autres clients) qui répondent à la requête de recherche d'un fichier. La taille du message est variable.

Name	Size in bytes	Default Value	Comment
Protocol	1	0xE3	
Size	4		The size of the message in bytes not including the header and size fields
Type	1	0x42	The value of the OP_FOUNDSOURCES op-code
File Hash	16	NA	The requested file hash
Sources Count	1	NA	The number of sources in this message
List of sources	Varies	NA	A list of sources

Tableau A.7 : *Message Found-source.*

A.8 Message Search-request

Ce message est envoyé du client au serveur pour la recherche de fichiers. La taille du message est variable.

Name	Size in bytes	Default Value	Comment
Protocol	1	0xE3	
Size	4		The size of the message in bytes not including the header and size fields
Type	1	0x16	The value of the OP_SEARCHREQUEST op-code
Parsed search string	varies	NA	The parsed search string format is described below
File Type Constraint	varies	NA	Optional. A string constraint. The string values are one of ("Audio", "Video", "Pro" or "Image" The type field is 3 bytes: 0x1 0x0 0x3
Min Size Constraint	Varies	NA	Optional. An integer constraint. The file size is provided in mega bytes. The type field is 4 bytes: 0x1 0x1 0x0 0x2
Max Size constraint	Varies	NA	Optional. An integer constraint. The file size is provided in mega bytes. The type field is 4 bytes: 0x2 0x1 0x0 0x2
Availability Constraint	Varies	NA	Optional. An integer constraint. Sets a lower limit on the number of client that poses the searched file The type field is 4 bytes: 0x1 0x1 0x0 0x15
Filename Extension constrain	Varies	NA	Optional. A string constraint. The type field is 3 bytes: 0x1 0x0 0x3

Tableau A.8 : *Message Search-request.*

A.9 Message Search-result

C'est un message envoyé du serveur au client comme une réplique au message ci-dessus. La taille du message est variable.

Name	Size in bytes	Default Value	Comment
Protocol	1	0xE3	
Size	4		The size of the message in bytes not including the header and size fields
Type	1	0x16	The value of the OP_SEARCHRESULT opcode
Result Count	4	NA	The number of search results in this message
Result list	Varies	NA	A list of search results

Tableau A.9 : *Message Search-result.*

A.10 Message Server-description-request

Ce message est une part de communication UDP entre un client et serveur. Il est envoyé périodiquement aux serveurs.

Name	Size in bytes	Default Value	Comment
Protocol	1	0xE3	
Type	1	0xA2	The value of the OP_SERVER_DESC_REQ opcode

Tableau A.10 : *Message server-description-request.*

A.11 Message Status-response

C'est une réplique envoyer par le serveur au client via une connexion UDP.

Name	Size in bytes	Default Value	Comment
Protocol	1	0xE3	
Type	1	0x97	The value of the OP_GLOBSERVSTATRES opcode
Challenge	4	NA	Unsigned integer. Could be an echo of the challenge sent by the client or a different number. In the latter case packet processing stops
User Count	4	NA	Optional. The number of users currently logged in to the server
Files count	4	NA	Optional. The number of files in the server's database
Soft files limit	4	NA	Optional. Unsigned integer server soft file limit
Hard files limit	4	NA	Optional. Unsigned integer server hard file limit
UDP flags	4	NA	Optional. The server's UDP flags. Two distinctive flags are defined: 0x01 indicates the the server supports the get sources message. 0x02 indicates the the server support the extended get files message

Tableau A.11 : *Message Status-response.*

A.12 Message Server-description-response

Ce message est envoyé du serveur au client comme une réponse à la demande de description de serveur du client. La taille du message est variable.

Name	Size in bytes	Default Value	Comment
Protocol	1	0xE3	
Type	1	0xA3	The value of the OP_SERVER_DESC_RES opcode
Name	NA	NA	The server's name, a string encoded in a 2-byte length, char array value format
Description	NA	NA	The server's description, a string encoded in a 2-byte length, char array value format

Tableau A.12 : *Message Server-description-response.*