République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderahmane Mira de Béjaïa

Faculté des Sciences et Sciences de l'Ingénieur

Département d'Informatique

École Doctorale Réseaux et Systèmes Distribués

Mémoire de Magister en informatique

***

Option

Réseaux et Systèmes Distribués

Thème

Détection de l'attaque Eclipse dans le réseau eDonkey

Présenté par : Asma DAOUDI

Directeur du mémoire : Pr. BOUABDALLAH Abdelmadjid

Devant le jury composé de :

Président	DAHMANI Adbennasser	Professeur	Université de Bejaia,Algérie
Rapporteur	BOUABDALLAH Abdelmadjid	Professeur	UTC, Compiègne,France
Examinateur	AHMED-NACER Mohamed	Professeur	USTHB, Alger,Algérie
Examinateur	BADACHE Nadjib	Professeur	USTHB, Alger,Algérie
Invité	TARI Abdelkamel	Chargé de cours	Université de Bejaia,Algérie

Promotion 2006 – 2007

Résumé

Dans ce présent travail, nous avons analysé l'attaque Eclipse dans le cadre du réseau eDonkey. Dans cette attaque, un serveur attaquant a pour but de cacher la partie surveillée du reste du réseau en prenant comme cible la liste des serveurs du réseau eDonkey. Nous avons identifié deux scénarios différents de l'attaque Eclipse : Le premier scénario traite le cas où l'attaque est menée par un seul attaquant et le deuxième scénario traite le cas de plusieurs attaquants. Pour faire face à cette attaque nous avons proposé une solution basée sur la taille des listes des serveurs d'un client détecteur. Les résultats obtenus montrent l'efficacité de la solution proposée à détecter les deux scénarios d'attaques et à isoler les serveurs attaquants du reste des serveurs du réseau. Cependant, des fausses (accusations) alertes peuvent avoir lieu. Pour cela, nous avons proposé d'utiliser le profilage des utilisateurs, des ports TCP, UDP pour minimiser l'impact de ces fausses accusations.

Mots clés : réseax pair-à-pair, eDonkey, l'attaque Eclipse, détection d'attaques dans les réseaux P2P.

Abstract

In this work, we analyzed the Eclipse attack in an eDonkey network. In an Eclipse attack, attacker goal is to keep the supervised part hidden from the rest of the network using the list of servers in eDonkey network as a target. Two different scenarios of Eclipse attack are identified. The first one treats the case where there is only one attacker mount an Eclipse attack on an eDonkey network. The second scenario treats the case of several attackers. To prevent from this attack, we propose solution based on lists of servers size of detector's client. The obtained result shows the efficiency of our solution to detect the two scenarios of Eclipse attack and avoid (isolate) those attackers from the rest of the network. However, false positive can happened. We proposed to use a profiling of the users, TCP and UDP ports to mitigate false positive affect.

Keywords : Peer To Peer Networks, eDonkey Network , Eclipse attaks, attaks detection in P2P networks .

Dédicaces

Je dédie ce travail à ma très chère mère ,
A mon adorable père,
A mes très chères sœurs et frères ,
A toute ma famille,
A tous ceux que je ne cite pas mais aux quels je pense très fort,
vous mes amis(es)

Remerciements

Je remercie Dieu le tout Puissant qui m'a donné la force et la volonté pour réaliser ce modeste travail.

Mes remerciements iront à mes encadreurs Abdelmadjid BOUABDALLAH, Professeur à l'UTC (Compiègne, France) et Hani Ragab Hassen, Docteur à l'UTC (Compiègne, France) pour m'avoir soutenu durant cette année de Magister. Ce travail n'aurait jamais pu aboutir sans eux, qui sont toujours su me consacrer des moments de leurs temps, me guider et me conseiller. Je souhaite leur transmettre l'expression de ma reconnaissance et ma plus profonde gratitude.

Je remercie tout particulièrement les membres de jury, qui ont accepté de juger mon travail.

Je tiens aussi à remercier chaleureusement Kamel TARI, le Chef de département d'informatique et responsable de l'école doctorale, qui nous a toujours encouragé et soutenu, et je tiens aussi à remercier Monsieur Djoudi TOUAZI responsable du centre de calcul pour son aide et sa gentillesse avec tous le monde.

Mes sincères remerciements à tous nos enseignants et tous ceux qui ont contribué à la création et la réussite de l'école doctorale ReSyD de Bejaia.

Je remercie également mes amis et collègues de l'école doctorale de Béjaïa pour avoir créé un environnement d'études convivial durant toute cette formation.

Je ne pourrais clôturer ces remerciements sans me retourner vers ma raison d'être qui ont toujours été à mes cotés et qui m'ont entourée d'affection, de soutien et d'amour, ma très chère mère et mon adorable père. Qu'ils trouvent dans ce modeste travail le fruit de leur soutien.
Je voudrais aussi remercier profondément mes frères et sœurs,ainsi que toute ma famille.

Enfin je remercie tous ceux qui, de près ou de loin, ont contribué à l'aboutissement de ce travail.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

AUJOURD'HUI, les environnements de calcul et de communication sur Internet sont sensiblement plus complexes et divers que les systèmes répartis classiques, manquant de toute organisation centralisée ou commande hiérarchique. Ceci ne constitue qu'une simple face des protocoles et des utilisations du Pair-à-Pair, qui proposent une manière forte et équitable de collaborer en vue d'accroître le potentiel du réseau. Pour ce fait, on s'aperçoit que les architectures centralisées commencent à atteindre leurs limites pour le partage de fichiers de tailles importantes et le Pair-à-Pair a vu le jour.

Nous connaissons actuellement le Pair-à-Pair surtout par des logiciels comme Napster, Gnutella, eDonkey, et plus récemment BitTorrent, pour les échanges de fichiers, de musique ou de film, de manière illicite.

Il existe plusieurs définitions de Pair-à-Pair (("P2P" ou "Peer To Peer"). Milojicic et al. [47]propose la définition : "le calcul pair-à-pair est le partage des ressources et services par échange directe entre les systèmes". Une autre definition est donnée par Schollmeier et al. [70] : "le pair-à-pair se réfère à une classe des systèmes et applications qui utilisent les ressources distribuées pour exécuter une fonction critique d'une façon décentralisée". D'après ces deux définitions, le P2P modélise donc l'échange direct des ressources et services entre ordinateurs, chaque élément peut être client et serveur à la fois.

Le P2P en tant que technologie de partage est apparu bien avant le logiciel Napster et Internet. Le réseau USENET développé en 1979 était déjà basé sur ce type d'architecture qui, plus généralement, a façonné un certain nombre d'architectures et de protocoles préfigurant dans Internet. Mais c'est le succès de Napster apparu en 1999 qui a propulsé cette technologie au niveau des usagers. Les systèmes pair-à-pair ont évolué depuis le simple système de partage de fichiers Napster, jusqu'aux systèmes de gestion de données distribuées comme le système Seti@home.

Un des principaux avantages des réseaux Pair-à-Pair est l'utilisation maximale des ressources du réseau, en éliminant les coûts d'infrastructure et en faisant communiquer directe-

ment les clients entre eux. Cependant, l'aspect sécurité pose beaucoup des problèmes dans les réseaux Pair-à-Pair.

Actuellement, la sécurité des réseaux Pair-à-Pair s'oriente principalement autour de l'anonymat. D'autre travaux traitent la disponibilité et l'authentification dans des réseaux structurés.

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

Des mécanismes réactifs tels que les IDS (Intrusion Détection System) et les méthodes de détections des attaques sont déployées pour sécuriser les réseaux P2P. Leurs objectif est de rendre le système P2P résistant à des personnes mal intentionnées. C'est dans ce contexte que se situe le travail que nous allons présenter dans ce mémoire. Le problème auquel nous avons traité est la détection de l'attaque Eclipse sur le réseau eDonkey.

Organisation du mémoire

Le présent mémoire comporte une introduction générale, six chapitres et une conclusion générale.

Le premier chapitre est consacré à la présentation du modèle P2P d'une manière générale.

Dans le deuxième chapitre, nous présentons le détail du réseau eDonkey et du protocole BitTorrent.

Dans le troisième et le quatrième chapitre , nous donnons un état de l'art sur la sécurité dans les réseaux P2P, et nous présentons quelques attaques et quelques contre-mesures envers ces attaques.

Dans le cinquième chapitre, nous présentons quelques méthodes de détection d'attaques et quelques systèmes de détection d'intrusion existants sur les réseaux P2P.

Le sixième chapitre, est dédié à notre proposition où nous présentons notre algorithme de détection de l'attaque Eclipse sur le réseau eDonkey.

Nous terminons ce mémoire par une conclusion et des perspectives.

Chapitre 1

Les réseaux pair-à-pair

1.1 Introduction

L'expression "réseau peer to peer" (P2P), que l'on traduit généralement par réseau de "poste à poste", "pair à pair" ou encore "d'égal à égal" désigne une architecture de réseau où les postes connectés communiquent directement entre eux et partagent leurs ressources. Tous les postes ont un rôle équivalent, à la fois client et serveur par rapport à ces ressources (espace de stockage, puissance de calcul...etc.), d'où leur appellation de "servent" (contraction de serveur et client). L'objectif de ce chapitre est de présenter de manière générale le Peer-To-Peer. Le premier point situe le modèle Peer To Peer dans les systèmes informatiques, définit leurs objectifs et caractéristiques. Alors que le deuxième point traite des différentes architectures sur lesquelles il se base, enfin nous présentons quelques mécanismes de fonctionnement des réseaux pair-à-pair.

1.2 Modèle P2P

Le modèle Peer-to-Peer est à la base d'Internet dès l'origine. Il a été largement médiatisé ces dernières années, suite aux péripéties judiciaires liées aux partages de fichiers protégés par un copyright. Les applications du modèle P2P vont du partage de films ou de fichiers musicaux pour le grand public au travail collaboratif ou au calcul distribué pour l'entreprise. Quelques applications, classifications, objectifs et autres feront l'objet des sous sections suivantes.

1.2.1 Taxonomie des systèmes informatiques

Les systèmes informatiques peuvent être classés en deux grandes catégories : les systèmes centralisés et les systèmes distribués. Ces derniers peuvent être construits selon deux modèles, le modèle client/serveur (plat ou hiérarchique) et le modèle pair-à-pair qui peut être pur ou hybride, comme le montre la figure 1.1.

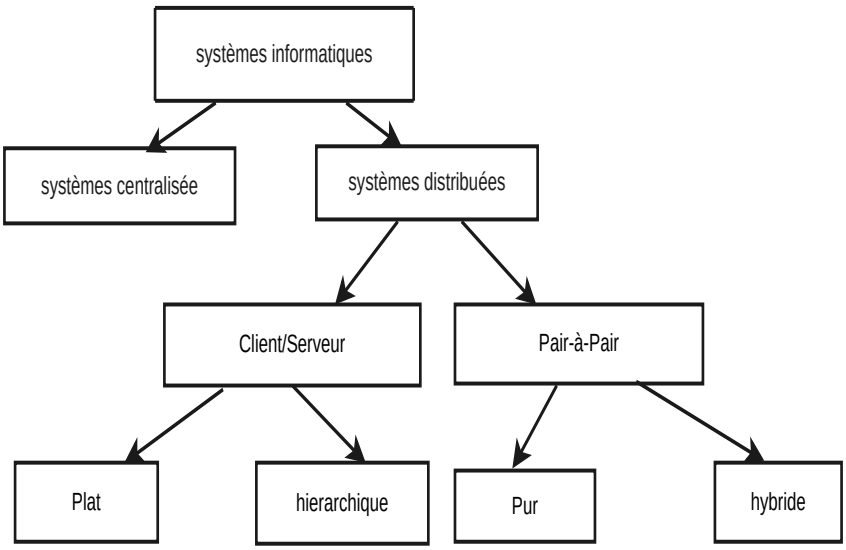

Fig. 1.1 – Classification des systèmes informatiques.

1.2.2 Taxonomie des systèmes Pair-à-Pair

Dans le modèle pur P2P, il n'existe pas de serveur central, c'est le cas des systèmes Gnutella et Freenet [23] par exemple. Et dans le modèle hybride, un serveur est contacté en premier pour obtenir des méta-informations, comme l'identité du nœud (peer) sur lequel les informations sont stockées, puis la communication est réalisée en P2P comme les systèmes : Napster, Groove et Magi. Il existe aussi des modèles intermédiaires avec des SuperPeers, comme le modèle Kazaa. Les SuperPeers contiennent des informations que les autres peers n'ont pas. Ces autres peers consultent les SuperPeers s'ils ne peuvent pas trouver l'information autrement [51].

1.2.3 Quelques caractéristiques d'un réseau Pair-à-Pair

Un réseau Pair à Pair est caractérisé par sa décentralisation, son dynamisme, et l'hétérogénéité entre ses noeuds. Ces facteurs influencent sur les performances et le déploiement des

réseaux P2P.

- **Décentralisation**
  Il n'y a pas de contrôle centralisé, ni de vue globale de tous les pairs dans le réseau. La majorité des systèmes P2P ne sont pas purement décentralisés, mais ils sont hybrides.
- **Dynamisme et Hétérogénéité**
  Les systèmes P2P doivent supporter le dynamisme, c'est-à-dire les ressources comme par exemple les pairs, peuvent joindre ou quitter le système de manière continue, sans qu'ils affectent le fonctionnement de ce dernier. Tous les pairs participants à un réseau P2P sont hétérogènes du point de vu système utilisé, capacité de stockage et finalement la manière dont ils sont connectés aux autres pairs du réseau.
- **Anonymat**
  L'anonymat est défini comme étant le degré pour lequel les systèmes P2P tiennent compte des opérations non identifiées [14].
- **Autres caractéristiques**
  Un système P2P doit être ***extensible*** jusqu'à des centaines de milliers de nœuds avec une augmentation de performances correspondantes. Une défaillance du serveur ou du collecteur de résultats ne doit pas avoir d'incidence sur les machines du système. La perte d'une ou de plusieurs machines participantes ne doit pas affecter la performance du système, donc il doit être ***Tolérable aux fautes***. Comme tout environnement connecté à Internet, un système de calcul pair-à-pair doit assurer la ***sécurité*** du serveur, des machines de calcul et de l'application elle-même contre les attaques que nous allons détailler dans les prochains chapitres du mémoire.

1.2.4 Les applications Pair-à-Pair

Il existe plusieurs types d'applications peer-to-peer, nous pouvons les regrouper en quatre grandes classes comme suit :

– **Les applications Pair-à-Pair parallèles**

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

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

Tout le domaine des grilles de calcul se rapproche du P2P. Des travaux portent actuel-

lement sur le partage de mémoire et d'adressage global à très grande échelle, au dessus d'une infrastructure P2P.

- **La gestion de partage de fichiers et des contenus**

  Ce type d'applications consiste à stocker et retrouver des informations sur différents éléments du réseau. L'application principale concerne l'échange du contenu ; est fortement représenté par Napster [50], Gnutella, Morpheus, Freenet (anonymat des sources et intégrité des documents), Kazaa [44] et BitTorent [4](transfert parallèle de plusieurs sources, possibilité d'arrêt et de reprise d'un transfert). Les bases de données distribuées et les tables de hachage distribuées commencent aussi à utiliser les protocoles P2P comme l'application Mariposa par exemple.

- **La collaboration**

  Elle permet aux usagers de collaborer en temps réel sans utiliser un serveur central. Une application populaire est l'utilisation des messages instantanés comme Yahoo, AOL, Jabber, skype[3]... Le principe d'applications partagées ou de travail coopératif est de permettre aux utilisateurs de travailler de manière commune sur un projet distribué. Ces applications sont de plus en plus nombreuses : Groove, Magi, PowerPoint distribué..etc. Le commerce électronique commence aussi à utiliser des modèles P2P. Un autre aspect de collaboration est : les jeux en réseau (DOOM par exemple), dont l'architecture est exempte de toute autorité centrale.

- **Les plates-formes**

  Les plates-formes proposent une infrastructure générique pour développer des applications Peer To Peer. Elles assurent les mécanismes de fonctionnement de base d'une application P2P comme : la gestion des peers, le nommage, la découverte de ressources, la communication entre peers, sécurité et agrégation des ressources. Les plates-formes les plus connues sont : JXTA, .net, Anthill.

1.2.5 Objectif du modèle P2P

Le modèle P2P étant très général, ses applications sont de nature très différente. En effet, ces objectifs sont variés et nous pouvons citer :

- Le partage et la réduction des coûts entre les différents peers.
- La fiabilité et le passage à l'échelle, en supprimant tout point central de panne permet

d'accroître la fiabilité et d'améliorer le passage à l'échelle en évitant les goulots d'étranglement.
- L'agrégation des ressources et l'interopérabilité, en mettant en commun des ressources individuelles comme celle de la puissance de calcul ou de l'espace de stockage.
- L'accroissement de l'autonomie en l'absence d'une autorité centrale, il est à la responsabilité de chacun de partager ou non des fichiers.
- L'anonymat assuré par certaines applications, en utilisant par exemple des algorithmes de routage qui rendent quasiment impossible le pistage d'une requête.
- La communication ad-hoc est collaborative.

1.2.6 P2P vs Client/Server

Conceptuellement, le modèle P2P est une alternative au modèle client/serveur, qui est constitué d'un serveur ou d'un cluster de serveurs et de beaucoup de clients. Dans le modèle pur du P2P, il n'y a plus de serveur, et tous les participants sont des peers comme le montre la figure 1.2.

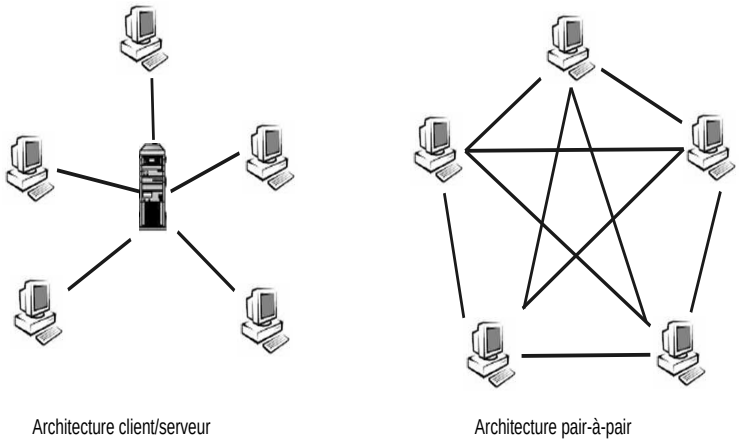

Fig. 1.2 – P2P vs Client/Server.

Et le Tableau 1.1 montre quelques différences entre le modèle client/serveur et le modèle P2P.

Peer To Peer	**Client/Serveur**
Auto-organisé	Gestion centralisée
Evolution dynamique, Ad-hoc	Configuré
Découverte des peers	Consultation de tables
Flux distribué	Flux centralisé
Symétrie du réseau	Asymétrie du réseau
Communication par messages	Orienté RPC
Adressage dynamique au niveau application	Adressage statique @IP
Entités Autonomes	Entités dépendantes
Attaques difficiles (mobilité, anonymat)	Attaque plus simple

TAB. 1.1 – Comparaison entre le modèle client/serveur et le modèle P2P.

1.2.7 Quelques avantages des réseaux P2P

Un des principaux avantages des systèmes pair-à-paire est l'utilisation maximale de la puissance du réseau, en éliminant les coûts d'infrastructure et en faisant communiquer directement les clients entre eux. Nous pouvons noter encore comme avantage :

- L'optimisation de l'utilisation de la bande passante du réseau.
- la réduction du coût du système, dans le modèle Client/Serveur, le serveur doit être assez puissant pour effectuer les tâches demandées par les clients et il doit disposer d'une bande passante suffisante pour transmettre assez rapidement les données à tous les clients.
- Résistance aux pannes : est due au fait que les ressources sont dupliquées alors on a toujours une copie de la ressource en cas de perte ou de panne.
- Extensibilité : Un nouvel utilisateur a juste besoin de se connecter à un nœud du réseau pour devenir à son tour un nœud du même réseau.
- Utilisation des ressources non consommées, telles que la puissance du processeur ou le disque dur (Espace de stockage distribué).

1.2.8 Quelques inconvénients des réseaux P2P

- Apparition / Disparition de ressource à tout moment d'où la difficulté d'administration de ces réseaux. De plus, une ressource sur un tel système a une durée de vie parfois limitée.
- La possibilité d'enfreindre la réglementation sur les droits d'auteurs par échange de fichier, protégé par des copyrights, car il est illégal de télécharger des fichiers de type musicaux, logiciels, vidéos ou autre relevant du droit d'auteur sans s'acquitter de ce

dernier (Loi du World Wide Web).
- Les fichiers mis a disposition des utilisateurs par les peers sur les réseaux P2P, outre leur distribution potentiellement illégale issus de sources invérifiables, peuvent contenir : des virus, spywares et autres ...
- Absence de la confidentialité, un intrus peut facilement trouver l'adresse IP d'un utilisateur.
- Exposition des réseaux P2P aux attaques distribuées que nous allons détailler par la suite dans ce mémoire.

1.3 Les différentes architectures des réseaux peer to peer

Dans [31] plusieurs architectures des réseaux P2P ont été développées.

1.3.1 Architecture centralisée

Le réseau pair-à-pair le plus connu du grand public est sans doute Napster[50]. Son originalité réside dans le fait qu'il utilise une architecture centralisée ce qui a contribué à son succès mais aussi à sa perte.
Dans toute architecture centralisée, un dispositif exclusivement serveur se charge de mettre en relation directe tous les utilisateurs connectés. L'intérêt de cette technique réside dans l'indexation centralisée de tous les répertoires et intitulés de fichiers partagés par les abonnés sur le réseau. En général, la mise à jour de cette base s'effectue en temps réel, dès qu'un nouveau utilisateur se connecte ou quitte le service.
Cela fonctionne avec le client comme avec un moteur de recherche classique : Les requêtes sont lancée en inscrivant un mot clé. L'utilisateur obtiendra une liste d'autres utilisateurs actuellement connectés au service et dont les fichiers partagés correspondent au terme recherché. Dès lors, il suffit de cliquer sur un des intitulés de lien pour se connecter directement à la machine correspondante et entamer le transfert. Dans ces conditions, à aucun moment les fichiers ne se retrouvent stockés sur le serveur central (voir figure 1.2).

1.3.2 Architecture Distribuée (Pur P2P)

Si un utilisateur désire supprimer les serveurs centraux il faut donc trouver le moyen de constituer un annuaire sur chaque client, puis de les faire communiquer. C'est sur ces mécanismes que sont basés les réseaux Peer to Peer décentralisés (figure 1.3).

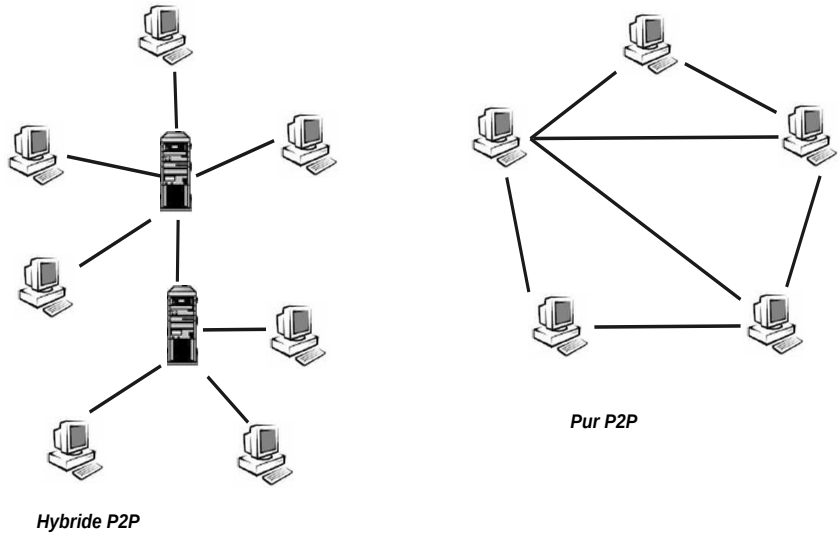

FIG. 1.3 – Modèles Pur et Hybride des réseaux P2P.

1.3.3 Architecture Superpeers (Hybrid P2P)

Le modèle hybride (figure 1.3) a pour but d'utiliser les avantages des 2 types de réseaux (centralisé et décentralisé). En effet sa structure permet de diminuer le nombre de connexions sur chaque serveur, et ainsi d'éviter les problèmes de bandes passantes.
D'autre part le réseau de serveurs utilise un mécanisme issu des réseaux décentralisés pour tenir à jour un annuaire des clients et un index des fichiers à partir des informations provenant des autres serveurs. Un serveur peut donc proposer à n'importe quel client toutes les informations contenues sur le réseau.
Les pairs dans un système P2P sont souvent hétérogènes dans la puissance, la stabilité, et la connectivité de calcul. Les systèmes purement décentralisés ne peuvent pas prendre avantage de cette hétérogénéité tandis que les systèmes hybrides peuvent en tirer profit. Cependant, les nœuds et les *superpeers* de domination doivent être soigneusement choisis pour éviter le seul point d'échec et pour entretenir les goulots d'étranglement. Dans ce cas le réseau n'est plus pollué par les trames de broadcast.
En se basant sur la façon dont les nœuds du réseau P2P sont liés entre eux (la topologie), nous pouvons classifier les réseaux P2P en non structurés ou structurés tel que [42] :

- Dans les architectures non structurées il n'existe aucune règle qui définit où les données sont stockées, et la topologie du réseau est arbitraire. La topologie virtuelle et la topologie physique sont les mêmes. Un réseau P2P non structuré est formé quand les liens entre les pairs sont établis arbitrairement, ce type de réseaux peut être facilement construit,

un nouveau pair qui veut joindre le réseau peut copier des liens existants d'un autre pair et puis former ses propres liens avec le temps. Si un pair veut trouver une donnée dans le réseau, sa requête doit être inondée dans le réseau afin de trouver une réponse à sa demande. Plusieurs pairs peuvent avoir la donnée. Parmi les réseaux P2P populaires non structurés, nous pouvons citer : Freenet, Gnutella, KaZaA, BitTorrent et overnet /eDonkey2000...
Le principal inconvénient de tels réseaux est que les requêtes ne peuvent pas toujours être résolues.

- Les réseaux P2P structurés surmontent les limitations des réseaux non structurés tels que dans un réseau structuré l'architecture du réseau et le placement de données est indiqué avec précision, les voisins d'un nœud sont bien définis, les données sont stockées dans des endroits bien précis, et la topologie du réseau est étroitement contrôlée en utilisant une table de hachage distribuée (DHT :Distributed Hach Table (Figure 1.4)) comme principe de placement des informations d'une manière déterministe et en permettant à chaque pair d'être responsable d'une partie spécifique du contenu dans le réseau.

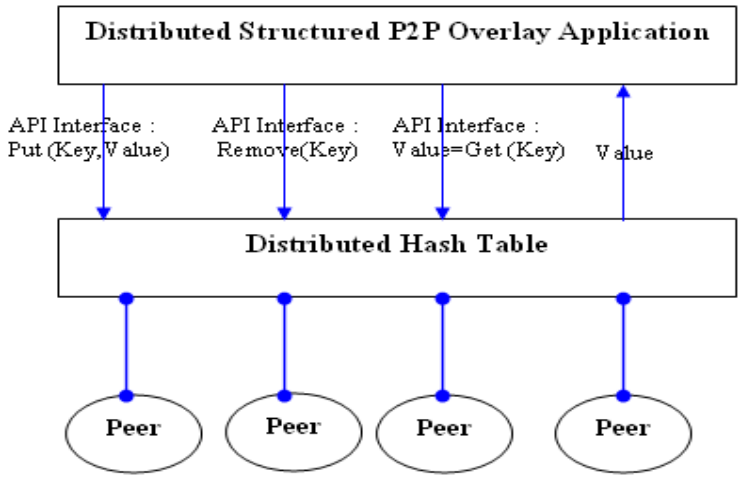

FIG. 1.4 – Interface d'application du DHT sur les réseaux P2P structurés.

Ces réseaux assignent des valeurs à chaque morceau du contenu et de chaque pair dans le réseau et puis suivent un protocole global en déterminant quel pair est responsable de quel contenu. De cette façon, toutes les fois qu'un pair veut rechercher quelques données, il emploie le protocole global pour déterminer les pairs responsables de celui des données et puis dirige la recherche vers les pairs responsables.
Parmi les réseaux structurés bien connus de P2P nous pouvons citer : Chord [24], CAN [40], pastry [2], Tapestry[40], Kademlia [43], et Viceroy [40]...
Par la suite nous n'allons nous intéressé qu'aux réseaux non structurés, en détaillant quelques protocoles de cette classe tels que : eMule /eDonkey2000 et BitTorrent, que nous utiliserons dans la suite de notre travail.

1.4 Etude des mécanismes de base de fonctionnement des réseaux P2P

Une des caractéristiques des réseaux peer-to-peer est que la qualité et la quantité des données disponibles augmentent à mesure que le nombre d'utilisateurs augmente, et la pertinence d'un système peer-to-peer réside dans sa capacité à localiser les ressources efficacement quelle que soit la taille du réseau. Ainsi, ces systèmes doivent reposer sur des méthodes efficaces de découverte des ressources désirées.

1.4.1 Méthodes de recherche dans les réseaux P2P non-structurés

Dans un système P2P non structuré,il n'existe aucune règle qui définit strictement où les données sont stockées et quel nœud est voisin de l'autre, alors différents travaux ont été fait [42] pour résoudre ce type de problème comme l'originale Gnutella qui est basée sur la méthode de recherche **BFS** (*Breadth First Search*), et le Freenet qui est basé sur la méthode de recherche **DFS**(*Depth First Search*) avec une profondeur limite D pour les deux approches et qui sont des méthodes basées sur le principe d'inondation. La profondeur D se rapporte au niveau système au maximum TTL (*Time To Leave*) d'un message en terme du nombre de sauts.
Dans **BFS** le nœud émetteur de la requête envoie sa demande à tous ses voisins. Chaque voisin fait ses traitements sur la requête et renvoie le résultat si les données sont trouvées. Ce voisin expédie alors la demande plus loin à tous ses voisins excepté le nœud émetteur de la requête. Ce procédé continue jusqu'à ce que la profondeur limite D soit atteinte.
Avec la recherche **DFS**, chaque nœud envoi sa requête à un voisin, et attend la réponse définitive de ce voisin avant l'envoi de la requête à un autre voisin (si la requête n'est pas satisfaite), ou renvoi le résultat en arrière au nœud émetteur de la requête (si la requête est satisfaite).
Donc l'inondation essaye de trouver un nombre maximum de résultats dans l'anneau centré par le nœud émetteur de la requête et au rayon D. Cependant, l'inondation produit un grand nombre de messages (parmi eux sont des messages doubles) ce qui n'est pas bon.
Plusieurs schémas alternatifs ont été proposés pour résoudre le problème d'inondation. Parmi ces travaux nous trouvons : *iterative deepening* [86], *k-walker random walk* [41], *modified random BFS* [30], *two-level k-walker random walk* [26], *directed BFS* [86], *intelligent search* [30], *local indices based search* [86], *routing indices based search* [13], *attenuated bloom filter based search* [66], *adaptive probabilistic search* [81], et *dominating set based search* [87].
Dans [42]plusieurs taxonomies de ces méthodes ont été proposées selon les critères :

- Méthode probabiliste / déterministe.

- Basé sur BFS/ Basé sur DFS.
- Regular-grained /Croase-grained tel que dans la première méthode tous les nœuds du réseau participent dans la propagation de la requête, et dans la deuxième seul un sous ensemble de nœuds participent.
- Blind search/ Informed search (Recherche aveugle/Recherche informé) tel que dans la première les nœuds du réseau ne préservent pas des meta-informations à l'inverse de la deuxième qui garde des informations pour faciliter la recherche.

La classification des différentes méthodes de recherche selon le dernier critère indiqué est donnée par :
les méthodes d'aveugle : iterative deepening, k-walker random walk, modified random BFS, et two-level k-walker random walk.
Les recherches informé : les index locaux (local indices), BFS dirigé (directed BFS), recherche intelligente (intelligent search), routage des index (routing indices), filtre atténué de fleur(attenuated bloom filter).

1.4.2 Les réseaux à index

Les réseaux à index correspondent à la vision " classique " du pair-à-pair. Le principe est d'avoir un réseau de pairs tous identiques reposant sur des index. Ces index sont centralisés ou distribués, et permettent l'indexation des ressources proposées par les différents pairs. Une ressource est indexée afin de maintenir l'association entre une ressource et un ensemble de pairs. Lorsqu'un pair recherche une ressource, il demande à l'indexeur de le mettre en relation avec un autre pair (ou un ensemble de pairs) possédant la ressource. Cette approche est largement utilisée (dans par exemple les systèmes : Napster, FastTrack, Kazaa et BitTorent2) car elle permet de mettre à disposition un plus grand nombre de ressources personnelles. Ces réseaux sont fragiles en deux points : la mise à disposition initiale de la ressource ne se fait que sur une seule machine et la fragilité du principe d'indexation qui repose sur un ensemble de noeuds pour identifier l'ensemble des ressources du système.

1.4.3 Les réseaux à " traînée "

Les réseaux à " traînée " cherchent à déposer la ressource initiale en plusieurs endroits afin de la rendre plus stable sur le réseau. Freenet [23]est le premier réseau de ce type. Cette approche est également adoptée dans le réseau à clé de hash. Dans Pastry [2]où la ressource est routée à travers différents nœuds du réseau et où chaque hôte ainsi traversé reçoit une copie de l'association clé/localisation de la ressource. Lors de la recherche, le cheminement se fait

en se rapprochant petit à petit de la source initiale de la ressource. Si le chemin permettant d'atteindre la ressource " croise " le chemin qui a permis de la déposer, la ressource est renvoyée par le nœud se trouvant à la jonction de ces deux chemins.

1.4.4 Réseaux à tables de hashage Distribuées (DHT)

Un protocole de routage a une grande influence sur l'exécution globale du système dans les applications Pair-à-Pair (P2P). Dans les protocoles de routage basés sur le DHT courant, les requêtes sont distribuées à travers tous les pairs du système. Cependant, un saut de cheminement doit pouvoir se produire entre deux pairs largement séparés avec des latences élevées dans le réseau qui augmente considérablement les overheads de cheminement dans le système. Un réseau pair-à-pair fondé sur les clés de *hash* distribuées réalise une correspondance 1-1 entre un identifiant et une valeur de *hash*. Cette correspondance permet de placer directement une ressource dans le réseau de pairs.
La répartition des clés d'identifications est distribuée dans un anneau dans le cas du modèle Chord [24], dans un hypercube dans le modèle Pastry ou dans un espace euclidien virtuel dans le cas des réseaux CAN [40].
Les algorithmes de Chord et de CAN sont relativement simples et faciles à contrôler. Cependant, ils ne considèrent pas trop le caractère topologique du réseau. Par conséquent, ils peuvent réaliser le nombre moyen minimal de sauts de routage mais pas de la vraie latence minimal. En fait, dans la plupart des cas, le chemin choisi par ces algorithmes n'est pas le chemin le plus court mesuré par latence de lien. L'algorithme de *HIERAS* [85] est plus simple et plus facile à contrôler que l'algorithme de Pastry tandis qu'il améliore considérablement l'efficacité du routage pour d'autres algorithmes basés sur DHT tels que Chord.
Ratnasamy et *Shenker* [64]ont précisé que le P2P ou d'autres applications à grande échelle de réseau pourraient potentiellement tirer bénéfice d'un certain niveau de la connaissance au sujet de la proximité relative entre ses nœuds participants. Ils ont suggéré d'assigner les pairs topologiquement adjacents avec les identifiants conformes ensemble, et ils ont appliqué cette idée sur le CAN avec une bonne amélioration d'exécution.
HIERAS a l'objectif semblable et utilise également la propriété topologique des noeuds pour améliorer le routage dans les réseaux pair-à-pair.

1.5 Conclusion

Le P2P a connu et connaît toujours un succès auprès du grand public grâce aux logiciels de partage et de communication, donc le P2P ne doit pas être systématiquement associé aux applications illégales de partage de fichiers car il est actuellement utilisé dans de nombreux autres domaines d'applications tels que : le travail collaboratif ou le calcul distribué...etc.

Si nous considèrons l'évolution actuelle de l'informatique, avec les réseaux ad hoc, les réseaux sans fils, la mobilité, l'accroissement de la puissance des machines et de la bande passante, il nous semble que dans ce contexte, les réseaux P2P prennent leurs places et occupent une partie importante du trafic réseau.

Dans ce chapitre, nous avons fait un tour d'horizon sur les systèmes P2P les plus connus, nous les avons classifiés en deux grandes classes, les systèmes structurés et non structurés et nous n'allons nous intéressé qu'aux réseau non-structurés par la suite.
Le détail des réseaux P2P les plus utilisés fera l'objet du chapitre suivant.

Chapitre 2

Le réseau eMule/eDonkey et le protocole BitTorrent

2.1 Introduction

Le nombre d'application permettant le téléchargement illégal de fichiers avec droits d'auteurs est impressionnant.
Le but de ce chapitre n'est pas de présenter une liste exhaustive, mais plutôt de regarder les applications les plus couramment utilisées, à savoir le réseau eMule/eDonkey et le protocole BitTorrent.

2.2 eMule/eDonkey

eMule 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.

2.2.1 Fonctionnement du réseau

Le client doit se connecter à un serveur pour récupérer des services réseau. La connexion avec le serveur reste ouverte tant que le client est encore dans le système .
Les serveurs sont des serveurs d'index et ne communiquent pas avec d'autres serveurs. Chaque client eMule est pré_configuré avec une liste de serveurs et une liste de fichiers partagés sur son système de fichiers local [88].
Un client utilise une seule connexion TCP pour loger dans le réseau, et obtenir des informations

sur les fichiers désirés et les clients disponibles.
Le client eMule utilise aussi plusieurs centaines de connexions TCP avec les autres clients qui sont utilisées pour télécharger et déposer des fichiers (Figure 2.2). Chaque client eMule

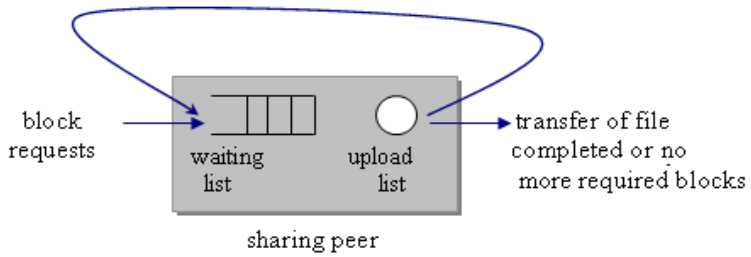

FIG. 2.1 – Un modèle de file de téléchargement.

maintient une file de téléchargement pour chacun de ces fichiers partagés [32][56](Figure 2.1). Les clients demandeurs de fichier joignent la queue de la file et avancent graduellement jusqu'à

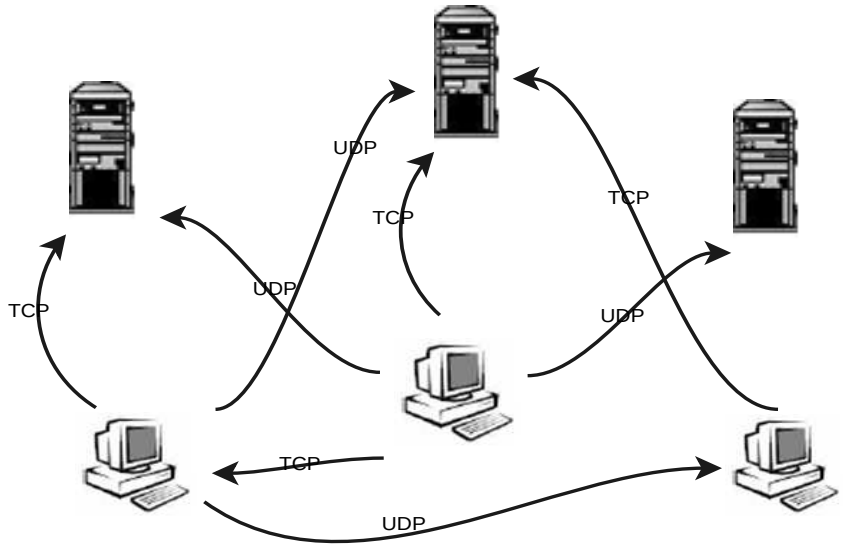

FIG. 2.2 – Schéma simplifier du réseau eMule.

ce qu'ils atteignent la tête de la file puis commencent leurs téléchargements. En général, la structure d'un fichier arbitraire f partagé sur le réseau eDonkey est illustrée sur la figure 2.3. Le client peut télécharger le même fichier à partir de plusieurs autres clients eMule, obtenant des fragments différents à partir de chacun.
Il peut aussi partager un morceau d'un fichier même avant la fin de téléchargement du fichier. Finalement, eMule prolonge les capacités d'eDonkey et permet aux clients d'échanger des informations sur les autres serveurs avec des paquets UDP comme le montre la figure 2.2.

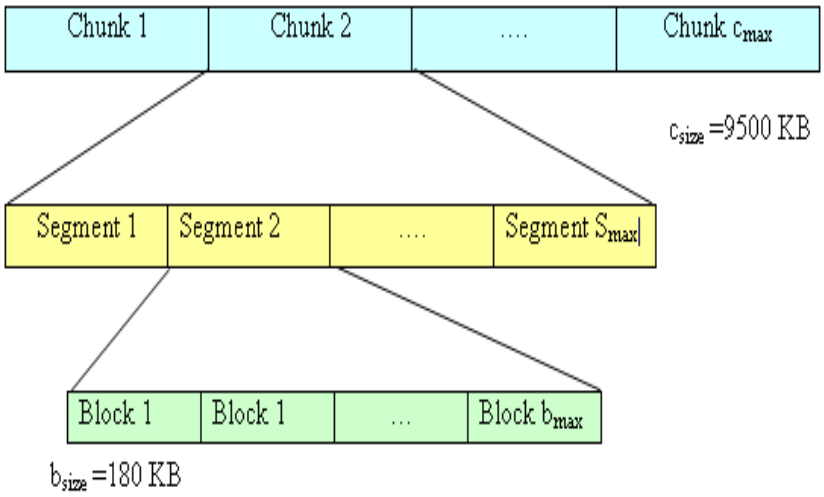

FIG. 2.3 – La structure d'un fichier eDonkey sur la couche application.

2.2.2 Le serveur eMule

Les communications des clients et serveurs sont des connexions basées TCP. Le serveur utilise une base de données interne dans laquelle il sauvegarde des informations sur les clients et les fichiers. Le serveur eMule ne garde aucun fichier. Il joue le rôle d'un index centralisé qui garde des informations sur la localisation des fichiers. Une fonction additionnelle du serveur est de jouer le rôle d'un pont entre deux clients qui se relient par un par-feu et ne peuvent pas accepter les raccordement entrants, cette fonction augmente considérablement la charge du serveur[80][15]. eMule emploie UDP pour augmenter les capacités d'un client.

2.2.3 Connexion Client/ Serveur

Au démarrage le client se connecte à un serveur eMule utilisant une connexion TCP (figure 2.4). Le serveur donne au client un ID [88]qui n'est valide que pour la durée de vie de la connexion du client avec le serveur (notons que quand un client a eu un high ID, il doit recevoir le même ID avec tous les serveurs jusqu'au changement de son adresse IP). Le client dépose ses fichiers chez le serveur. Puis il demande au serveur la liste des serveurs.
Une transaction de recherche - figure 2.5 - commence généralement par une requête Get_Sources (Annex : Tableau 6.9)d'un fichier spécifique qui sera répliqué par une liste de sources (@IP, N°Port) à partir desquels le client demandeur peut télécharger ce fichier.

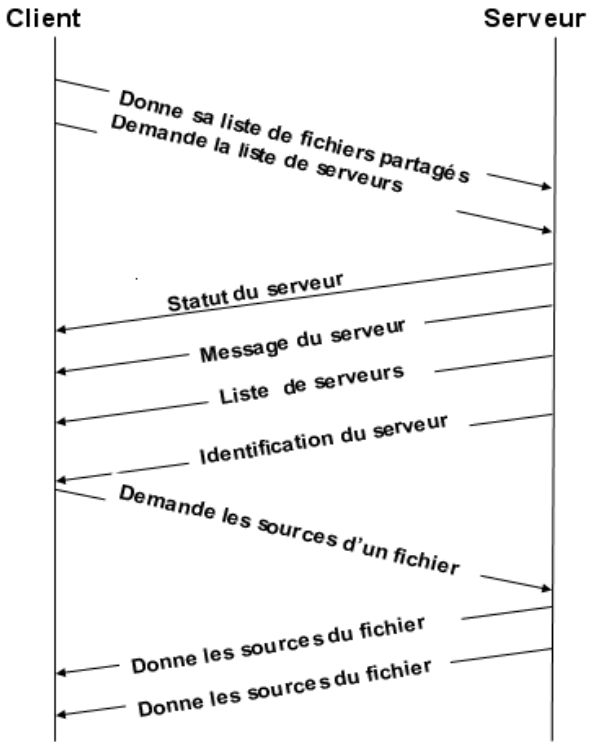

FIG. 2.4 – Connexion client / serveur TCP.

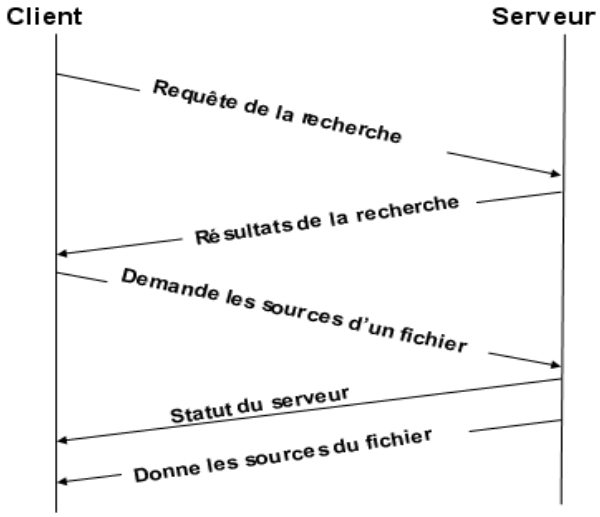

FIG. 2.5 – Dialogue entre le client et le serveur lors d'une requête de recherche .

2.2.4 Connexion Client/ Client

Un client se connecte à un autre pour télécharger un fichier. Chaque client a une file de téléchargement gardant une liste de clients qui sont en attente d'un téléchargement.
Quand la file de téléchargement est vide, la requête du demandeur doit être exécutée directe-

ment et si la file n'est pas vide la requête est ajoutée à la file d'attente.
Un client peut être acquis par un autre en attente avec un rang plus élevé dans la file. Dans les premières 15 minutes de la session de téléchargement, la file hiérarchique des clients est poussées pour prévoir le refus de nouveaux clients[33][88].
Le client qui dispose le fichier initialise une connexion (Figure 2.6) avec le client qui atteint la tête de sa file, pour qu'il puisse lui envoyer la partie du fichier demandé.
Si un client dépose sa demande dans plusieurs files, à la réception de la première réponse, il ne va pas avertir les autres d'enlever sa demande de leurs files d'attente, mais il va simplement rejeter les tentatives de dépôt d'une autre copie.

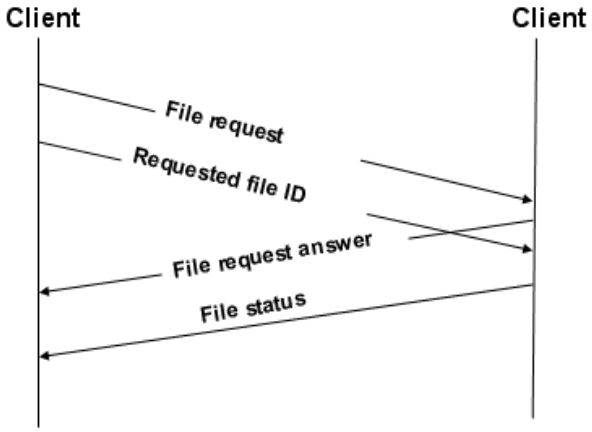

FIG. 2.6 – Connexion TCP client/client .

eMule utilise le credit_system pour encourager le partage de fichiers et prévoir l'usurpation d'identité.
Le credit_system est assuré en utilisant l'algorithme cryptographique à clé publique **RSA**.
La connexion d'un client peut utiliser un ensemble de messages qui ne sont pas définit par le protocole eDonkey, ces messages sont appelés le protocole étendu.
Le protocole étendu est utilisé pour l'implémentation du système de crédit, l'échange des informations en générale (comme la mise à jour des listes de serveurs et sources), et améliorer l'exécution en envoyant et en recevant les fragments compressées d'un fichier.
Le client eMule utilise des connexions UDP limitée de vérification périodique de l'état des clients sur les files de téléchargement tant qu'ils sont en attente d'un slot de téléchargement du fichier.

2.2.5 Liste des serveurs eDonkey

Pour accéder au réseau, la première étape est de se connecter à un serveur. Le système étant semi centralisé, il faut que le client possède une liste de serveurs (adresse IP et numéro 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[88].
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 de 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 leurs listes. Les serveurs testent régulièrement l'activité des autres serveurs. Si l'un d'eux ne répond plus, il sera supprimé de la liste[88].

2.3 BitTorrent

BitTorrent est un système P2P **centralisé** qui utilise un endroit central (tracker, voir Figure 2.7) pour contrôler les téléchargements des utilisateurs. Le terme BitTorrent désigne à la fois le protocole utilisé pour le partage et les logiciels clients qui l'implémentent [36] [61].

2.3.1 Présentation

BitTorrent n'est pas un réseau à proprement parler. Kademlia et eDonkey forment un réseau d'overlay sur le réseau Internet. Ils comportent un plan d'adressage et des protocoles de routage propres. Dans le cas de BitTorrent, il utilise directement le réseau Internet. L'idée de base est d'utiliser un téléchargement multi-sources, où chacun peut télécharger des parties différentes d'un fichier à partir de plusieurs sources. La charge du serveur est ainsi limitée.
Bram Cohen, est l'inventeur de ce protocole. Bien qu'il soit souvent détourné par les pirates pour échanger des fichiers illégaux, ce protocole se développe fortement pour échanger des fichiers légaux.

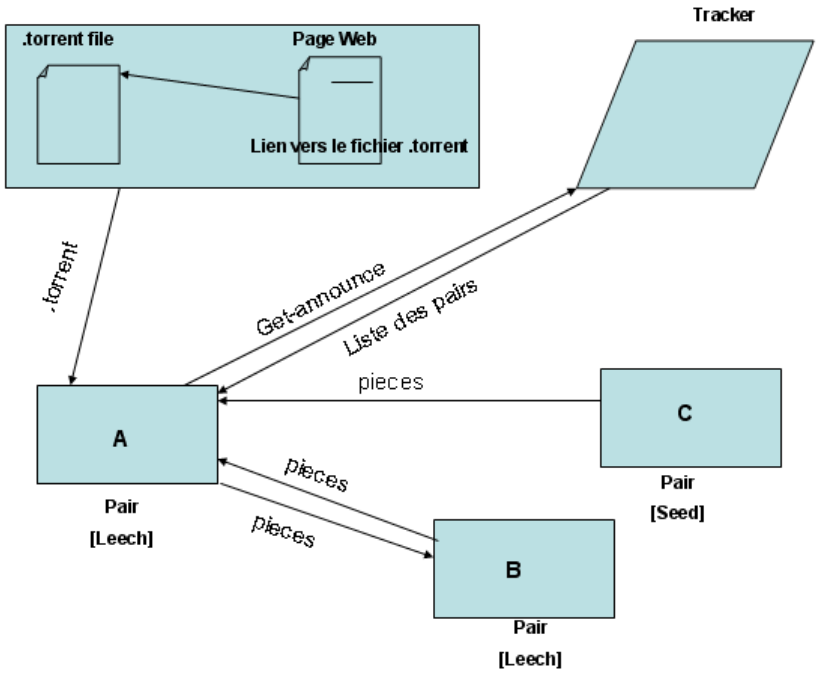

FIG. 2.7 – Fonctionnement de BitTorrent.

2.3.2 Principe de fonctionnement

Le système de téléchargement est basé sur un système de récompense ressemblant à l'algorithme du "**Tit For Tat**".
Pour **partager** un fichier, les clients commencent par la création d'un fichier "**torrent**" (suffix .torrent) qui contient des " métadata " , notamment le HASH File. Il contient aussi l'adresse URL du tracker qu'il faut interroger pour pouvoir télécharger le fichier.

Comme le montre la figure 2.7, pour **télécharger** un fichier, le client BitTorrent (programme client) recherche le .torrent correspondant (sur un serveur web) et le télécharge. Il utilise ensuite l'URL contenu dans le ".torrent" pour contacter le tracker responsable du fichier pour lui demander la liste des pairs partageant actuellement le fichier. Le tracker renvoie alors une liste d'adresses IP à contacter. Dès lors, le client BitTorrent contacte directement ces adresses et commence le téléchargement.
BitTorrent utilise la même technique qu'eDonkey pour gérer l'envoi des fichiers : ceux-ci sont découpés en fragments (de taille 256 Kbytes) qui peuvent être envoyés dans n'importe quel ordre.

Cette technique permet de faire du **multisourcing** : il est possible de télécharger plusieurs morceaux simultanément à partir de plusieurs pairs différents. De même, après avoir obtenu un morceau, le client servira de source pour l'envoyer vers d'autres utilisateurs. Il est bien entendu possible de stopper son téléchargement et de le reprendre là où on l'avait laissé [36]

[61].
Enfin le client BitTorrent informe le tracker, à des intervalles de temps réguliers, de son état dans le téléchargement (ce qu'il a téléchargé, reçu,..etc.) et reçoit du tracker une liste actualisée des pairs.

2.4 Comparaison de BitTorrent avec eDonkey

Une différence majeure entre BitTorrent et les autres protocoles Peer-to-Peer est qu'il ne possède **aucun moteur de recherche intégré**. Il faut donc utiliser un logiciel intermédiaire pour trouver les *.torrent*. Cependant, de plus en plus de navigateurs web peuvent intégrer un client BitTorrent et un mini moteur de recherche de fichiers *.torrent*, rendant le téléchargement très facile.
Contrairement aux autres réseaux populaires tels que FastTrack [58] et eDonkey, BitTorrent ne gère **pas de files d'attente** de téléchargement. Les pairs sont évalués toutes les 10 secondes et le client choisit de fournir des données à ceux qui lui en offrent le plus en retour. Le BitTorrent permet d'accroître la **vitesse de téléchargement**, contrairement à eDonkey où il faut souvent plusieurs jours pour télécharger de gros fichiers.

2.5 Conclusion

Nous avons présenté les protocoles eDonkey et BitTorrent, ainsi nous avons montré qu'ils remplissent parfaitement leurs missions en étant efficaces dans leur domaine. Cependant il reste des possibilités d'amélioration. C'est pourquoi nous pensons qu'à long terme, ils vont probablement laisser la place à d'autres protocoles. Malgré cela, nous pouvons d'ores et déjà, dire qu'ils auront marqués une étape importante dans le monde des protocoles peer to peer.

Un des problèmes majeurs des réseaux P2P est la sécurité. Un état de l'art sur la sécurité dans ces réseaux fera l'objet des deux chapitres suivants.

Chapitre 3

Sécurité dans les réseaux P2P

3.1 Introduction

La sécurité des systèmes pair-à-pair n'est apparue que très récemment (premiers travaux en 2002). La sécurité va de la conservation des propriétés de disponibilité en présence de nœuds malicieux, à la fourniture de services de contrôle aux applications.
À l'heure actuelle, seule la garantie de la disponibilité et l'authentification ont été réellement étudiées.

Dans ce chapitre, nous allons parler de la sécurité des réseaux peer to peer. Nous allons donc citer quelques propriétés de la sécurité informatique qui sont visé par les attaquants, puis nous présenterons quelques fonctions de la sécurité informatique, quelques algorithmes de chiffrements et mécanismes de sécurité.

3.2 Sécurité informatique

La sécurité informatique sur un réseau peer to peer consiste à garantir aux *données* et aux *services* **les trois propriétés** suivantes : **confidentialité, intégrité et disponibilité**. Sur chaque *système*, une **politique de sécurité** définit les **propriétés** attendues et les **moyens** à utiliser pour les mettre en œuvre. Dans cette partie, nous décrivons **les mécanismes habituels de sécurité** , utilisés en environnement centralisé adapté pour les réseaux P2P. L'ensemble de ces mécanismes représente ce qu'on attend d'un système sécurisé. Nous distinguons **deux types** de mécanismes de sécurité : ***la sécurité proactive et la sécurité réactive.***

3.2.1 Sécurité proactive

La sécurité proactive regroupe l'ensemble des mesures telles que la cryptographie [79][73]pour protéger l'information et garantir aux *données* et aux *services* les trois propriétés : confidentialité, intégrité et disponibilité. La ***confidentialité*** consiste à empêcher des utilisateurs non autorisés d'accéder à une information. L'***intégrité*** vise à assurer que les ressources et les informations ne soient pas altérées ou détruites par des sujets non autorisés. Enfin, la ***disponibilité*** est la garantie que l'information soit accessible.
Notons que la **confidentialité et l'intégrité** dépendent principalement des protocoles **cryptographiques** [79][45], et la **disponibilité** implique également **des notions de sûreté de fonctionnement**.
Cette **différence** montre d'une part le fait que ces deux domaines de recherche soient séparés. La *confidentialité et l'intégrité* sont mises en place à travers **cinq services** [59][73] : *l'authentification, l'autorisation, la confidentialité du système, l'intégrité du système et la non-répudiation* (figure 3.1).
L'authentification permet de garantir l'identité de l'utilisateur ; **l'autorisation** est responsable du contrôle de l'accès aux ressources du système par les entités authentifiées ; la **confidentialité** du système garantit la non-divulgation d'information par le système global (serveur et réseau) aux entités non autorisées ; l'**intégrité** du système garantit la non-altération du contenu par des entités non autorisées ; la **non-répudiation** empêche l'utilisateur de 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* comme le montre la figure 3.1, que nous allons détailler par la suite dans ce mémoire.

	chiffrement	Signature	Ctrl d'accès	Ctrl de routage	Tiers de confiance
Authentification	X	X			
Autorisation			X		
Confidentialité	X			X	
Intégrité	X	X			
Non répudiation		X			X

FIG. 3.1 – mécanismes utilisés par les services de sécurité

3.2.2 Sécurité réactive

La sécurité **réactive** consiste à **détecter les comportements malveillants** (*il s'agit principalement de la détection des attaques*).

Les **systèmes de détection d'intrusions** [39][63][53] essayent de détecter les comportements anormaux du système. Pour cela, il existe **deux** possibilités : *détecter un comportement malveillant, ou détecter un comportement non prévu.*

Dans le cas de la détection d'un comportement malveillant [28], il s'agit de modéliser le comportement **d'attaque**. La solution couramment envisagée est basée **sur une base de signatures**, avec les problèmes inhérents à cette méthode. En effet, toute attaque récente (*0-days*) n'est pas détectée, générant des *faux négatifs* (intrusions non détectées), et la mise à jour de cette base est un travail rigoureux.

Dans le cas de la détection de comportement non prévu, il s'agit de modéliser le comportement **normal** du système. La plupart des approches proposant ce modèle reposent sur **une phase d'apprentissage**, qui ne peut être *ni exhaustive* (certains comportements normaux 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, ces attaques sont apprises comme un comportement normal et ne sont pas détectées. La **non exhaustivité** génère donc des **faux positifs** (*fausses alertes*) et la **non garantie** peut provoquer des **faux négatifs** (*intrusions non détectées*).

Remarquons que les faux positifs (fausses alertes) sont aussi dangereux que les faux négatifs, car ils noient les vrais positifs (alertes réelles).
Une étude d'une attaque particulière et sa détection sur le réseau eDonkey fera l'objet de ce mémoire.

3.2.3 Sécurité dans les systèmes actuels

Les réseaux P2P actuels sont tous en danger à travers les attaques répétées des majors de l'audiovisuel qui sont protégées par de puissantes associations comme la **RIAA** (Recording Industry Association of America) et la **MPAA** (Motion Picture Association of America).

Le grand défi des prochaines générations de logiciels P2P sera donc de garantir l'anonymat des utilisateurs et le chiffrage des données qui transitent. Deux logiciels, à l'état embryonnaire, semblent se démarquer : Ants [67] et Mute [68].
Ants est un projet Java développé par l'Italien Roberto Rossi. Il utilise une méthode de chiffrement asymétrique à double clé. Autrement dit, la clé publique sert d'identifiant unique (Unique ID) et est partagée avec les autres nœuds du réseau lorsque l'utilisateur envoie une requête (pour une recherche par exemple). Cette clé sert aussi à crypter le contenu de ce qui est retourné en échange à l'utilisateur et seule sa clé privée peut déchiffrer ses données.

Jason Rohrer, développeur de Mute, estime pour sa part que cette façon de procéder ne peut empêcher les attaques de type Man-In-the-Middle [65]. Un nœud "espion" pourrait intercepter des requêtes, suivre les résultats et les décrypter avec sa propre clé privée.
Mute ne fonctionne pas de la même manière : aucune connexion directe n'est effectuée entre un utilisateur qui cherche un fichier et celui qui le possèdent et les paquets transitent par une suite de nœuds intermédiaires servant de routeurs.

3.3 Quelques propriétés de la sécurité informatique

Nous allons explorer ici des attaques sur les réseaux P2P et nous considérons ces attaques, dans un premier lieu, en regardant leurs objectifs qui visent les trois propriétés de sécurité suivantes : confidentialité, intégrité, et disponibilité.

3.3.1 Confidentialité

La confidentialité, vise à assurer que seuls les sujets (les personnes, les machines ou les logiciels) autorisés aient accès aux ressources et aux informations auxquelles ils ont droit. La confidentialité a pour objectif d'empêcher que des informations secrètes soient divulguées à des sujets non autorisés. L'objectif des attaques sur la confidentialité est d'extorquer des informations [72].
La violation de la confidentialité augmente le coût prévu de distribution de contenu et indique l'information qui peut être employée pour écrire les programmes qui attaquent l'intégrité et la disponibilité du système [18].
La première étape en protégeant la confidentialité du réseau est de chiffrer les données [79] envoyées de sorte que seuls l'émetteur et le récepteur sachent ce qui a été échangé.

3.3.2 Intégrité

L'intégrité vise à assurer que les ressources et les informations ne soient pas corrompues, altérées ou détruites par des sujets non autorisés. L'objectif des attaques sur l'intégrité est de changer, d'ajouter ou de supprimer des informations ou des ressources. L'intégrité des informations dans les réseaux P2P peut être violée par :

- L'introduction de contenus de qualité dégradée (dans le contexte de la musique introduction des enregistrements de bruit).
- Ou dénaturer l'identifiant des contenus (fausser les titres des chansons par exemple).

3.3.3 Disponibilité

la disponibilité vise à assurer que le système soit bien prêt à l'emploi, que les ressources et les informations soient en quelque sorte consommables, que les ressources ne soient pas saturées, que les informations, et les services soient accessibles et que l'accès au système par des sujets non autorisés soit prohibé. L'objectif des attaques sur la disponibilité est de rendre le système inexploitable ou inutilisable [18]. La disponibilité des ressources dans les réseaux P2P peut être violée par :

- La surcharge de la bande passante d'une machine victime (similaire à l'attaque DoS traditionnel de la couche réseau)
- ou le blocage du CPU avec un grand nombre de requêtes complexes sans la consommation de la totalité de la bande passante, ou de la mémoire de stockage [72].

Pour atteindre ces objectifs de sécurité, il est nécessaire de mettre en oeuvre une politique de sécurité, applicable à l'ensemble des entités à l'intérieur d'un domaine géographique ou fonctionnel. Cette politique désigne l'ensemble des lois et des consignes à fins de protéger les ressources et les informations contre tout préjudice à leur confidentialité, leur intégrité ou leur disponibilité, lequel serait dû à un usage inapproprié (incorrect, abusif ou frauduleux). La politique de sécurité utilise un ensemble de fonctions de sécurité dont nous allons détailler dans la section suivante.

3.4 Quelques fonctions de la sécurité informatique

La politique de sécurité utilise un catalogue de fonctions de sécurité, parmi ces fonctions nous pouvons citer :

- **L'identification** des sujets, des objets et des opérations effectuées par ces sujets sur ces objets. Il s'agit de donner un nom à une personne, à une carte graphique, à un document, ou à un paquet IP. Un sujet qui n'a pas de nom est anonyme. Dans ce cas, le sujet ne peut être tenu responsable d'une action fautive. Un sujet peut avoir un pseudonyme (un alias) : il cache alors son vrai nom, mais reste responsable des actions qu'il pourrait exécuter sous son faux nom.

- **L'authentification,** c'est-à-dire la preuve de l'identité de ces entités ou de ces opérations. Il s'agit d'un processus incorruptible pour garantir que le sujet est bien celui qui prétend être, c-à-d on associe à chaque entité et chaque opération une identité unique.

- **Le contrôle d'accès**, c'est-à-dire la restriction d'accès aux ressources et aux informations, aux seuls sujets qui sont autorisés.

- **L'autorisation** des actions par un sujet sur des objets. Les droits spécifiques et les privilèges des sujets sont définis par cette fonction.

- **l'audit du système**, c'est-à-dire 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 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'auditabilité est une fonction qui consiste à pouvoir récupérer des preuves numériques incontestables, en cas de perquisition des données ou d'examen ultérieur des activités.

- **Anonymat** : 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", la première étant les réseaux centralisés (Napster[50]) et la seconde les réseaux décentralisés (Gnutella , eDonkey [33], Kazaa [44]). 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 avoir été proposé est Freenet [9]qui anonymise les communications par des mécanismes de proxy et chiffre les liens pour éviter l'écoute. En revanche, il est possible pour certains nœuds de faire le lien entre un identifiant et une adresse IP d'un nœud.

  Plus récemment, Mute [67]et Ants [68] ont proposé l'anonymisation totale des identifiants comme nous les avions déjà décrits dans la section 3.2.3. Ces réseaux "de troisième génération" apportent donc l'anonymat des pairs, au prix de nouveaux algorithmes de

routage plus complexes.

3.5 Quelques algorithmes de chiffrement et mécanismes de sécurité

Nous avons déjà parlé des algorithmes de chiffrement dans la deuxième section, et de quelques mécanismes de sécurité le long du document, et nous allons détailler dans cette section quelques uns de ces derniers que nous allons utiliser par la suite dans ce mémoire.

3.5.1 Cryptographie symétrique

La cryptographie **symétrique**, avec les algorithmes **DES**, **3DES**, **AES**, n'emploie qu'une seul clé pour chiffrer et déchiffrer un message. Il est donc nécessaire de distribuer cette même clé aux deux protagonistes de la communication. Si une personne s'adresse séparément à plusieurs personnes distinctes, elle aura besoin d'autant de clés distinctes.
Cette famille d'algorithmes sert à chiffrer en temps réel ou en différé, des documents, et des flots d'informations, car ces algorithmes sont puissants et nécessitent assez peu de ressources. En plus, la taille des clés est faible, par exemple 128 bits.

3.5.2 Cryptographie asymétrique

La cryptographie **asymétrique** emploie deux clés différentes : si l'on chiffre avec une clé, il faut déchiffrer avec l'autre. Il existe ainsi deux possibilités d'applications qui servent des objectifs distincts.
Dans le cas du système **RSA**, le système asymétrique le plus utilisé, la taille de clés les plus courantes est 1024 bits. Notons que les clés sont beaucoup plus longues que dans le cas du chiffrement symétrique. Ceci est dû à la nature arithmétique du système RSA dont la sécurité repose sur la difficulté de factoriser des grands entiers. Or les progrès mathématiques et informatiques dans le domaine de la factorisation imposent maintenant cette taille de clé. Ces algorithmes sont beaucoup plus lents (de l'ordre de 10 à 100) que les algorithmes symétriques. On les réserve donc au chiffrement et déchiffrement des messages courts.
Un message court peut justement être une clé d'un algorithme symétrique. Cryptographie asymétrique et cryptographie symétrique sont donc exploitées de manière complémentaire

et successive dans les protocoles cryptographiques pour authentifier l'émetteur, le récepteur, énoncer la non-répudiation des interlocuteurs, et déployer les systèmes de secrets qui vont permettre de communiquer de manière sécurisée.

3.5.3 Fonctions de hachage

Une fonction de hachage permet de calculer un résumé (une empreinte) de taille réduite et fixe (160 bits, par exemple) d'un document volumineux, écrit en binaire à l'aide de 0 et de 1. Les algorithmes les plus courants sont **MD4**, **MD5** (empreinte de 128 bits) et **SHA1** fournit des condensés 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 sur un disque d'ordinateur.

3.5.4 Signature numérique

Une signature numérique est une empreinte (d'un document) chiffrée par la clé privée de l'auteur, cette empreinte chiffrée étant jointe au document originel. La signature permet ainsi de vérifier l'intégrité du document et l'identité de l'expéditeur. On signe un document via des fonctions de hachage.

3.5.5 Infrastructures de confiance

Le problème de la cryptographie asymétrique provient de l'authenticité non établie de la clé publique d'un interlocuteur.
N'importe qui peut, à présent, engendrer un couple de clés privée-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 [65]).

3.5.6 Autorités de confiance

Pour éviter cette méprise (usurpation de clé), la solution consiste à s'en remettre à un tiers, en faisant signer la clé publique par une autorité digne de confiance, qui va ainsi garantir que la clé publique appartient bien au bon interlocuteur. On va donc signer numériquement le couple composé du nom du propriétaire de la clé privée et de la clé publique. La clé privée de l'autorité va donc signer ce couple de manière que la clé publique de cette autorité puisse permettre de vérifier cette signature.

3.6 Conclusion

Actuellement, la sécurité des réseaux Pair-à-Pair s'oriente principalement autour de l'anonymat. Certains travaux ont également été publiés concernant la disponibilité et l'authentification dans des réseaux structurés.

Nous avons parlé dans ce chapitre de la sécurité dans les réseaux peer to peer, dans lequel nous avons cité quelques propriétés de la sécurité informatique à savoir : la confidentialité, l'intégrité et la disponibilité, puis nous avons présenté quelques fonctions de la sécurité informatique, algorithmes de chiffrement et mécanismes de sécurité.

Maintenant, après avoir cité quelques objectifs des attaquants sur les réseaux P2P, nous allons présenter quelques attaques sur ces réseaux qui visent les trois propriétés de sécurité et les contre-mesures utilisées pour se défendre contre elles.

Chapitre 4

Quelques attaques sur les réseaux P2P et leurs contre-mesures

4.1 Introduction

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

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

Nous allons présenter dans ce chapitre quelques attaques sur les réseaux P2P (Sybil, Eclipse, Man in the Middle..etc.), et quelques défenses contre ces dernières.

4.2 l'attaque Sybil

4.2.1 Principe et conséquence

L'idée derrière cette attaque est qu'une simple entité malveillante peut présenter des multiples identités (Figure 4.1), et gagne ainsi le contrôle d'une partie du réseau.
Une fois que ceci a été accompli, l'attaquant peut brusquer le protocole de communication de

n'importe quelle manière. Par exemple il pourrait gagner la responsabilité de certains fichiers et choisir de les polluer.
Si l'attaquant peut placer ses identités d'une manière stratégique, les pertes peuvent être considérables [20][62].

Comme il pourrait choisir de continuer comme une attaque Eclipse (section 4.3), ou ralentir le fonctionnement du réseau en re-cheminant toutes les requêtes dans une autre direction.

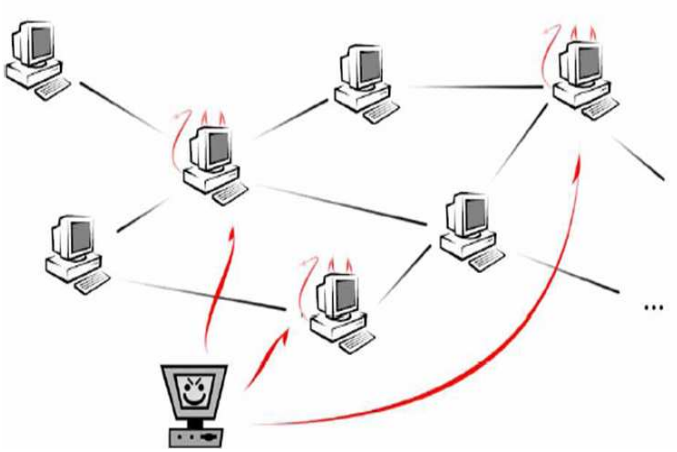

FIG. 4.1 – Exemple d'attaque Sybil

4.2.2 Défense

L'établissement d'une autorité de certificat de confiance pour faire associer à des entités distinctes des identités distinctes pour se défendre contre Sybil.
Donc le nœud génère localement le couple de clés publique / privée qu'il utilise pour l'encryptions et la signature des messages. Le certificat d'identité associe un ID aléatoire à chaque clé publique, il est donc difficile à l'attaquant d'obtenir plusieurs identifiants certifiés ou de choisir plusieurs IDs pour lancer l'attaque.
L'attaque Sybil peut avoir lieu dans tous les overlays qui exigent une entité et une cartographie d'identité (réseaux P2P, Ad hoc, wirless, sensor...) et des contre mesures de l'attaque sur ces réseaux ont été proposées dans ces références [83][84][20].

Aussi dans [89][69][5][19] des solution de défense contre l'attaque Sybil dans les réseaux P2P ont été proposé.

4.3 l'attaque Eclipse

4.3.1 Principe et conséquence

Avant que l'attaquant ne lance une attaque Eclipse, il doit gagner un contrôle sur un ensemble de nœuds tout au long des chemins stratégiques de routage.
Cette attaque peut être lancée en utilisant l'attaque Sybil, c'est à dire en se forgeant de nombreuses identités sur le réseau, afin d'être présent comme voisin sur un maximum de nœud dans le réseau.
L'attaque consiste alors à contrôler plusieurs pairs influant sur le réseau (une grande partie des voisins), pour ne pas rediriger le trafic vers les bons pairs (les pairs " sains "), et donc modifier l'utilisation normale du routage. L'attaquant cache alors les bons pairs sur le réseau, d'où le nom d'" attaque Eclipse ".
Une fois que l'Eclipse est terminée, il peut ensuite partager le réseau en plusieurs sous réseaux comme le montre la figure 4.2.

Ainsi, si un nœud peut communiquer avec un autre nœud qui appartient à un autre sous réseau, son message doit, à certain point, être routé par un des nœuds attaquants.
Eclipse peut attaquer le réseau par plusieurs manières efficaces :

- Niveau contrôle : par le re-routage inefficace des messages qui traversent les nœuds attaquants.
- Niveau donné : par l'injection de fichiers pollués, où l'attaquant peut décider de supprimer tout les messages qu'il reçoit, ainsi, la séparation complète de tous les sous réseaux.

Les conséquences aboutissent à l'extrême, si l'Eclipse permet à l'attaquant de contrôler la totalité du trafic du réseau [76][62][22][21][75].
Même si les attaquant ne contrôle qu'une petite partie du réseau, ils sont capables de lancer une attaque Eclipse.

4.3.2 Défense

La meilleure solution de défense contre Eclipse est l'utilisation d'un pur P2P et encore mieux si on utilise des algorithmes aléatoires de localisation des nœuds (Freenet par exemple). Si les nœuds sont aléatoirement distribués alors, ils n'ont aucune position stratégique et l'attaquant ne peut pas contrôler la position de ses nœuds.

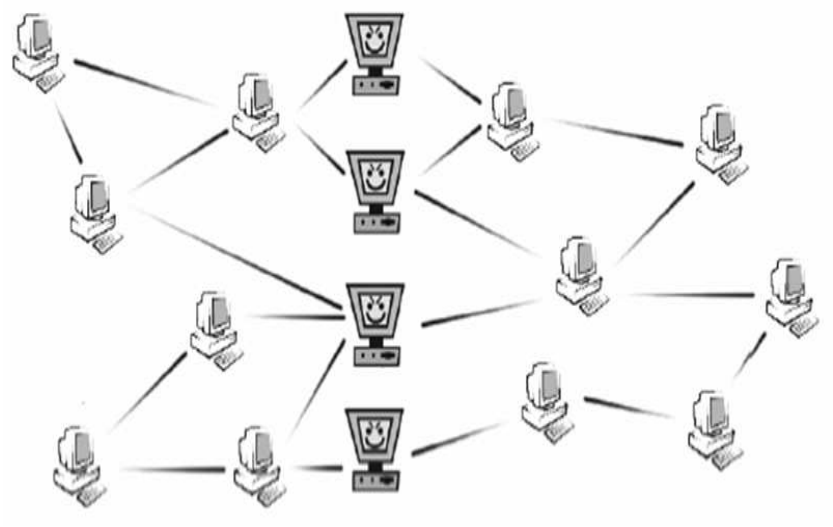

FIG. 4.2 – Exemple d'attaque Eclipse

Donc il est presque impossible de séparer le réseau en sous-réseaux dans ces conditions.

Une autre manière de prévoir l'attaque Eclipse est la limitation des degrés des nœuds. Cela veut dire que les nœuds corrects choisissent leurs voisins parmi le sous-ensemble des nœuds qui ont un in-degré proche d'un seuil configurable.
Mais cette défense introduit une nouvelle attaque puisque l'attaquant peut consommer le in_degré des nœuds corrects et empêche d'autres nœuds corrects de les pointer. Ainsi il est nécessaire de limiter le out-degré c-à-d les nœuds corrects choisissent leurs voisins parmi le sous-ensemble des nœuds qui ont des in-degré et out-degré proches des seuils fixés.
Une autre solution proposée dans [6]est l'utilisation d'une autorité de certification (CA) qui attribue et vérifie les identités du réseau.

4.4 l'attaque déni de service (DoS) et Distributed DoS dans les réseaux P2P

4.4.1 Principe et conséquence

L'attaque DoS est similaire à l'attaque DoS traditionnelle (Figure 4.3) de la couche réseau et DDoS (Distributed DoS) est une attaque DoS large échelle.
Le principe de ces attaques est de rendre un nœud et ces ressources indisponibles(exemple une surcharge de la bande passante d'un nœud).

DoS peut attaquer la disponibilité de toute autre ressource (exemple : bog du CPU en envoyant un grand nombre de requêtes complexes qui bloque le CPU du nœud sans consommation de toute la bande passante. Encore la mémoire de stockage peut être attaquée par un nœud malicieux qui permet de soumettre un document virtuel (qui n'existe pas) pour le stockage.
DoS peut attaquer la Qos des données disponibles (exemple : fabrique un fichier disponible mais après que la requête du fichier soit reçue ; elle sera servie mais avec une faible vitesse moins que celle demandée, donc le demandeur va perdre sa patience et annule le téléchargement avant sa fin, c-à-d décroissance des performances du réseau [49][17].

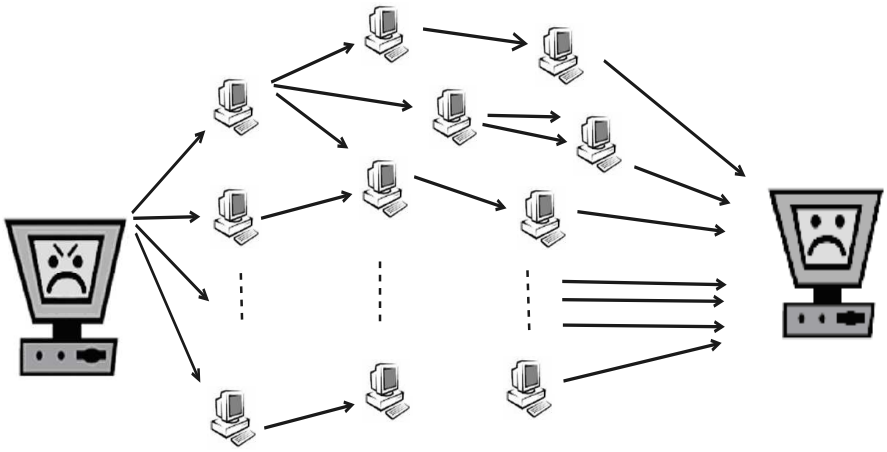

FIG. 4.3 – Exemple d'attaque DoS

4.4.2 Défense

Il est difficile de prévoir l'attaque DoS que ce soit dans les réseaux Internet traditionnels ou dans les réseaux P2P.
La solution de SynCookies (pour l'attaque TCP Syn Flooding) existe mais elle n'est supportée que par les systèmes d'exploitation linux et Solaris.

4.5 l'attaque Man In The Middle

4.5.1 Principe et conséquence

L'attaque Man in the Middle (MitM) se produit quand un attaquant se place entre deux nœuds du réseau comme le montre la figure 4.4, de tel sorte que toutes les communications entre ces deux nœuds passent par l'attaquant.

Cette attaque peut rester indétectable tant que l'attaquant reste passif. Cela permet à l'attaquant d'écouter toutes les communications à tout moment. Après le regroupement de suffisamment d'information, l'attaquant peut décider de devenir plus actif. Dans ce cas, l'attaquant peut modifier les messages qui le traversent, mais il peut aussi envoyer des messages falsifiés aux deux membres de la communication.
En utilisant ce mécanisme l'attaquant peut capter les identités des deux nœuds communiquants.
Cette attaque particulière peut être exécuté au niveau de la couche réseau de communication. Dans tel cas l'attaquant peut voir toutes les communications entre les couches inférieures des deux nœuds.
Tous les systèmes P2P qui n'ont aucun contrôle sur le placement des nœuds dans l'espace des IDs (la majorité des réseaux courant : Pastry, Chord,..etc.), sont extrêmement sensibles à ce type d'attaque. Le danger de cette attaque ne touche pas la totalité du réseau, au pire des cas le réseau va perdre ses trois nœuds (les 2 pairs et l'attaquant) [65][21].

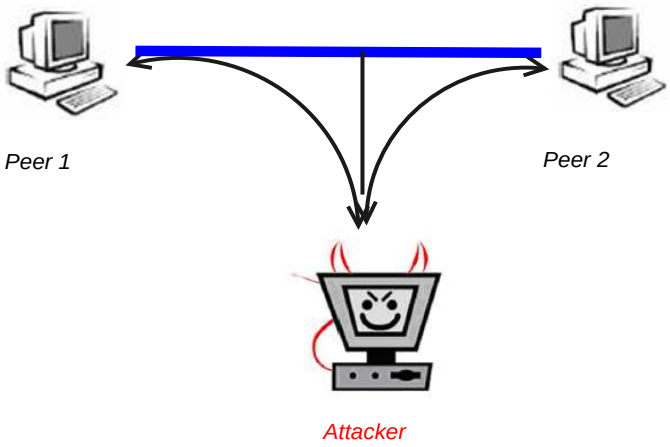

FIG. 4.4 – Exemple d'attaque Man In The Middle

4.5.2 Défense

L'idée principale de se défendre contre l'attaque MitM est le pure peer to peer sans aucune sorte de nœuds importants (comme par exemple : une autorité de certificat, un serveur d'indexe, ou des super nœuds).
Beaucoup de réseaux ont une certaines formes d'autorité centrale,ou de super nœuds comme outil d'empêcher d'autres formes d'attaque. Même pour ces réseaux qui manquent de ces points centraux, l'attaque de MitM pourrait être faite en large échelle (ce qui donne une attaque Eclipse, qui a des conséquences plus graves).

4.6 l'attaque Rational

4.6.1 Principe et conséquence

Le principe de cette attaque, et que les nœuds malveillants cherchent à se comporter d'une manière à réduire leurs propre ressources en partage, tout en maximisant leurs consommation de ressources.
Il y a beaucoup de raisons derrière ce comportement, parmi les quels l'économisation de la largeur de bande de téléchargement qui est fortement réglée par la plupart des fournisseurs du service Internet.
Il y a deux classes de base de cette attaque :

- *Content Restriction* : l'utilisateur ne partage pas ces fichiers sur le réseau.
- *Resource Restriction* : l'utilisateur ne contribue pas ses ressources sur le réseau.

Cela veut dire que les nœuds malveillants ne coopèrent pas avec le reste du réseau [52][62].

4.6.2 Défense

Plusieurs réseaux P2P ont fait l'expérience avec cette attaque, mais quelques uns entre eux ont pu atteindre à déjouer cette attaque.
Napster essaye de résoudre les problèmes de restriction du contenu et de ressources en donnant au participant un 'titre' pour le niveau par lequel ces nœuds partagent leurs ressources (construisant ce nœud plus célèbre parmi les autres utilisateurs).
Samsara [12](un système P2P de backup) exige qu'un nœud A ne puisse utiliser qu'un peu d'espace - sur un autre nœud B - proportionnel à ce que A partage sur le réseau.
Le système de téléchargement de BitTorrent est basé sur un système de récompense ressemblant l'algorithme du "tit-for-tat". Le principe est simple : j'envoie des données aux personnes qui m'en envoient également.
Ainsi, on accordera la priorité aux requêtes des pairs nous apportant le plus, c'est à dire ceux qui participent le plus à notre téléchargement.
Cette technique du "donnant-donnant" permet au BitTorrent de lutter efficacement contre les attaquants Rational.

4.7 Empoisonnement et pollution (Poisoning and pollution attacks)

4.7.1 Principe et conséquence

Ils ont décrit dans [38][9][37][35] que les attaques d'Empoisonnement et de pollution peuvent se produire, comme dans le réseau Internet, dans un réseau P2P par une description différente du contenu (exemple : le remplacement d'un fichier par un autre faux, ou l'ajout des index falsifiés au niveau du serveur d'index, ou par insertion de l' @IP d'une machine victime dans les tables de routage d'un ensemble de nœuds..etc.).
Les fichiers pollués n'ont certainement aucune utilité. Et dans le but d'attaquer par un **Empoisonnement** de fichier, les nœuds malicieux doivent falsifier leurs propres fichiers, et répondent aux requêtes qui leur arrivent par des fichiers corrompus (Figure 4.5).
D'ailleurs tous les messages qui passent par des nœuds malicieux peuvent être empoisonner (comme l'attaque Man in the Middle et l'attaque Eclipse). Cela peut donner aux fichiers empoisonnés une disponibilité élevée, rendant ces fichier plus attrayant au téléchargement que les vrais fichiers.

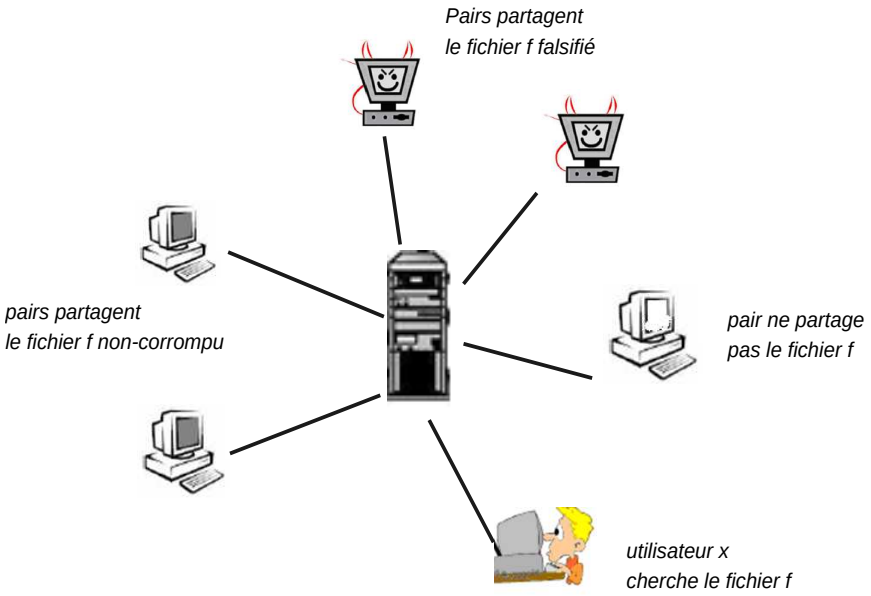

FIG. 4.5 – Exemple d'attaque d'empoisonnement

4.7.2 Défense

La contre-mesure pour l'attaque des serveurs d'index (une sorte d'attaque qui vise à falsifier les index et donc le re_acheminement des requêtes) est difficile car les connexions TCP sont complètes.
Et pour l'attaque des tables de routage, n'est pas fatale puisque plusieurs systèmes P2P mettent à jour leurs tables de routage automatiquement ce qui supprime les fausses adresses IP insérées au niveau de ces tables.

4.8 les Vers dans les réseaux P2P

4.8.1 Principe et conséquence

Un ver est un programme qui peut s'auto-reproduire et se déplacer à travers un réseau en utilisant les mécanismes réseau, ils n'ont pas réellement besoin d'un support physique ou logique (disque dur, programme hôte, fichier ...) pour se propager ; un ver est donc un virus réseau.
Les Vers constituent une des plus grandes menaces sur Internet. Actuellement, les Vers tels que Code Red ou Nimda sont capables d'infecter des centaines de milliers d'hôtes dans quelques heures si ce n'est pas quelques minutes.
Il y a plusieurs facteurs qui rendent les réseaux P2P attrayants aux Vers, parmi lesquels nous citons :

- Le faite que des réseaux P2P se composent des clients qui ont tous le même Software, un ver peut compromettre le réseau en entier dès qu'il trouve un trou exploitable de sécurité d'un seul client.
- Les nœuds P2P ont plusieurs connexions avec les autres nœuds. En effet un ver fonctionnant sur l'application P2P ne va pas faire un balayage précieux pour explorer d'autres victimes. Il devrait simplement chercher la liste des nœuds voisins de la victime et l'écarter dessus...etc.

Une fois que les vers finissent la propagation, leur but est habituellement de lancer les attaques massives comme DDOS (W32/Generic, worm !P2P, W32.SillyP2P..) contre les cibles politiques ou commerciales (whitehouse.gov, microsoft.com..).

4.8.2 Défense

L'idée principale pour se défendre contre les vers, est de maintenir l'application elle-même sécurisée. Sans cette vulnérabilité commune le ver ne peut pas se propager efficacement dans tout le réseau.
Une suggestion, qui a été donnée, est d'écrire les clients P2P avec les langues fortement typées, qui pourraient éviter beaucoup de défaut de sécurité (comme le buffer overflows).

Pour diminuer l'efficacité du ver, nous pouvons éviter les réseaux hybrides (qui contiennent 'des supernœuds'). Ces supernœuds fournissent une augmentation du taux de propagation d'un ver (en raison de leur degré de connectivité élevé).
Une autre possibilité pour réduire le danger des vers, est l'emploi d'un système d'exploitation dur et un réseau de nature ouverte pour développer des standards ouverts [91][90][11][74][8].

4.9 Conclusion

Un autre type d'attaque consiste à fournir des informations lorsqu'on l'interroge sur les données dont il est responsable. Il peut par exemple dire qu'il possède certains fichiers, et refuser les connexions quand d'autres pairs souhaitent les obtenir, etc.
Un moyen de contrer les attaques sur les données est la réplication : si les données sont présentes sur plusieurs pairs, alors le client pourra interroger les répliquas pour savoir si la ressource est vraiment indisponible. Le but est d'éviter qu'un seul noeud soit responsable d'une ressource.

Dans le cas général, un réseau P2P sera fortement perturbé si plus de $\frac{1}{4}$ du réseau sont des pairs malicieux [6].

Comme il existe toujours des failles, des mécanismes réactifs tels que les IDS (Intrusion Détection System) et les méthodes de détections des attaques sont également déployées.
L'objectif est de permettre de rendre un système résistant à des personnes mal intentionnées.
Dans le prochain chapitre nous détaillerons quelques méthodes de détection des attaques et des systèmes de détection d'intrusion dans les réseaux P2P existant.

Chapitre 5

La détection d'attaques dans les réseaux P2P

5.1 Introduction

Les techniques de détection d'intrusion (ou d'attaque) tentent de faire la différence entre une utilisation normale du système et une tentative d'intrusion puis donnent l'alerte.
Typiquement, les données d'audit du système sont parcourues à la recherche de signatures connues d'intrusion, de comportements anormaux ou d'autres choses intéressantes.

Dans ce chapitre, nous allons présenter quelques méthodes de détection des attaques dans les réseaux P2P qui ont été utilisées dans le réseau Internet. Puis, nous allons définir quelques approches de détection d'intrusion. Enfin, nous allons citer quelques systèmes existants de détection d'intrusion dans les réseaux P2P.

5.2 Quelques techniques de détection des attaques sur les réseaux P2P

Parmi les méthodes de détection des attaques dans le réseau Internet en général et dans les réseaux P2P en particulier nous allons citer les techniques suivantes :

5.2.1 La réputation

Les systèmes peer-to-peer actuellement utilisés par le grand public doivent faire face à différentes attaques. Pour ce faire, elles utilisent des systèmes basés sur la réputation [46].
La réputation d'un pair évalue la confiance qu'on peut lui accorder quant à la fiabilité de ses fichiers.
Le point clé de la fiabilité de l'information de réputation est sa localisation : où doit-on stocker la réputation d'un pair qui fournit des contenus ? Cette localisation doit permettre que l'information y soit sécurisée, et non répudiable par le fournisseur.

Trois possibilités sont à envisager : soit on stocke cette réputation chez le pair qui télécharge, soit chez des tiers, soit chez le pair qui fournit ce contenu. Lorsque la réputation est stockée chez le pair qui télécharge, cette réputation ne peut être biaisée aux yeux de ce pair, puisqu'elle est basée sur ses échanges précédentes ; par contre, cette localisation ne va pas permettre une identification rapide des pairs perturbateurs. Lorsque la réputation est stockée chez des tiers, on peut imaginer que cette identification soit au contraire plus rapide ; cependant, un tel stockage est compliqué à mettre en oeuvre : comment choisir alors les tiers et comment rendre la réputation aisément accessible ? De plus un tel stockage n'est pas dans l'esprit " pair à pair ".
La solution adoptée est donc le stockage chez le fournisseur, ainsi la réputation est facilement accessible et non répudiable par le fournisseur.
Dans [77] Sergio et Hector ont proposé une présentation d'une taxonomie des composants des systèmes P2P de réputation, leurs propriétés, et une discussion : comment se comportent les utilisateurs du système et les contraintes techniques qui peuvent être en conflit.

5.2.2 Agents mobiles

Les techniques à base d'agents mobiles peuvent être vues comme un ensemble d'entités autonomes. Chaque agent peut être différent des autres par son code ou par les données sur lesquelles il travaille. Cependant, s'ils ne diffèrent que par leurs données, leurs tests peuvent devenir prévisibles.

Plusieurs domaines de recherche dans les systèmes de détection d'intrusions par des agents mobile et de détection collaborative ont été fait dans [27][16][34][63] et cela dans les buts suivants :

5.2.2.1 Détection multi-point

La détection multi-point est faite en analysant les flux d'audit de plusieurs hôtes pour détecter des attaques distribuées ou autres stratégies d'attaques d'un réseau dans sa globalité. Il est rarement possible de transporter tous les flux d'audit à un IDS central, et même dans ce cas, la détection est difficile.
Les agents mobiles peuvent apporter le calcul distribué, le fait qu'on transporte l'analyseur au flux d'audit et non le flux d'audit à l'analyseur. Les agents pourraient détecter ces attaques, les corréler, et découvrir les stratégies d'attaques distribuées.

5.2.2.2 Architecture résistante aux attaques

Une architecture hiérarchique est souvent utilisée pour des raisons de performances et de centralisation du contrôle. Cette conception a souvent plusieurs lignes de communication non-redondantes. Un intrus peut couper une branche d'un système de détection d'intrusion, voire le désactiver totalement en le décapitant.
Les agents mobiles peuvent apporter quelques solutions à ce problème :

- une architecture complètement distribuée ;
- une architecture hiérarchique standard où un agent peut remplacer chaque nœud et ramener une fonctionnalité perdue ;
- des agents mobiles qui se déplacent quand une activité suspecte est détectée.

5.2.2.3 Agents errants

L'échantillonnage aléatoire est utilisé avec succès depuis de longues années dans le domaine du contrôle de qualité, et les mathématiques sous-jacentes sont bien comprises et les paramètres peuvent être calculés.
Chaque agent effectue un test spécifique et peut errer aléatoirement d'hôte en hôte.
Quand des tests indiquent la possibilité d'une intrusion, des tests plus poussés peuvent être effectués sur l'hôte suspect.

5.2.2.4 Imprévisibilité

Un intrus peut pénétrer dans un système sans être immédiatement détecté par le système de détection d'intrusion (IDS). Cela peut lui laisser le temps d'effacer ses traces ou de neutraliser l'IDS.

Un MAIDS reste vulnérable à cela mais se trouve néanmoins pourvu de quelques avantages. Lorsqu'un agent arrive sur un hôte, il transporte du code non encore altéré, et pourrait par exemple tester l'intégrité de la plateforme IDS locale. L'arrivée des agents et leur manière de fonctionner peuvent être imprévisibles, et il peut être très dur de rester inaperçu.

5.2.2.5 Diversité génétique

Chaque agent peut être différent des autres par son code ou par les données sur lesquelles il travaille.
Si chaque agent d'une même classe avait une façon différente de détecter la même chose, il serait autrement plus difficile de prévoir quoi que ce soit. Une manière de faire les choses serait de décrire ce qu'il faut détecter dans un langage standard et de laisser chaque instance de la classe trouver une manière de le détecter.
Les agents avec un faible taux de détection pourraient essayer de muter en introduisant de légères modifications dans leur façon de détecter l'attaque.

5.2.3 Profilage (profiling)

Une deuxième technique de détection est le profilage [54][55] qui est l'action de représenter en profil un objet. En d'autres termes, le profilage est l'action de représenter les caractéristiques générales d'une personne d'un point de vue professionnel.
Parmi les types de profilage existant nous citons les suivants :

5.2.3.1 Profilage des utilisateurs

On crée et on maintient des profils individuels du travail des usagers, auxquels ils sont censés adhérer ensuite. Au fur et à mesure que l'utilisateur change ses activités, son profil de travail attendu se met à jour. Certains systèmes tentent de concilier l'utilisation de profils à court terme et de profils à long terme.
Il reste cependant difficile de profiler un utilisateur irrégulier ou très dynamique. De plus, un utilisateur peut arriver à habituer lentement le système à un comportement intrusif.

5.2.3.2 Profilage de groupes

Nous plaçons chaque utilisateur dans un groupe de travail qui montre une façon de travail commune au groupe. Un profil de groupe est calculé en fonction de l'historique des activités du groupe entier. Nous vérifions que les individus du groupe travaillent de la manière que le groupe entier a définie par son profil.
Cette méthode réduit le nombre de profils à maintenir. De plus, un utilisateur peut beaucoup plus difficilement élargir le comportement accepté comme dans un profil individuel. Mais il est parfois dur de trouver le groupe le plus approprié à une personne : deux individus ayant le même poste peuvent avoir des habitudes de travail très différentes. Il est de plus parfois nécessaire de créer un groupe pour un seul individu.

5.2.3.3 Profilage d'utilisateurs de ressources

Nous observons l'utilisation de certaines ressources comme les comptes, les applications, les processeurs, les ports de communication sur de longues périodes, et on s'attend à ce qu'une utilisation normale n'induise pas de changement sur cette utilisation par rapport à ce qui a été observé par le passé. On peut aussi observer les changements dans l'utilisation des protocoles réseau, rechercher les ports qui voient leur trafic augmenter anormalement. Ces profils ne dépendant pas aux utilisateurs peuvent permettre la détection de plusieurs intrus qui collaboreraient.

5.2.4 Observation de seuil

On fixe le comportement normal d'un utilisateur par la donnée de seuil à certaines mesures (par exemple, le nombre maximum de mots de passe erronés). On a ainsi une définition claire et simple des comportements non acceptés. Il est cependant difficile de caractériser un comportement intrusif en termes de seuils, et on risque beaucoup de fausses alarmes ou beaucoup d'intrusions non détectées sur une population d'usagers non uniforme.

5.3 Quelques approches pour la détection d'intrusions sur les réseaux P2P

La détection d'intrusions a pour objectif de détecter toute violation de la politique de sécurité en vigueur sur un système informatique. Elle est basée sur l'analyse à la volée ou en temps différé de ce qui se passe sur le système.
Deux approches sont utilisées à cette fin : l'approche par scénario (misuse detection) et l'approche comportementale (anomaly detection). Chacune des deux présente des points forts, mais aussi des faiblesses. Les outils qui implémentent ces approches présentent également des forces et des faiblesses [60][29][1].

5.3.1 Approche par scénario

La détection de malveillances fonctionne essentiellement par la recherche d'activités abusives, par comparaison avec des descriptions abstraites de ce qui est considéré comme malveillant. Cette approche tente de mettre en forme des règles qui décrivent les usages non désirés, en s'appuyant sur des intrusions passées ou des faiblesses théoriques connues.
Les règles peuvent être faites pour reconnaître un seul événement qui est en lui-même dangereux pour le système ou une séquence d'événements représentant un scénario d'intrusion.
L'efficacité de cette détection repose sur la couverture de tous les abus possibles par les règles. Et la difficulté de cette approche se constate lors de la construction de la base de signatures et elle n'assure pas la détection d'attaques non connues.

5.3.2 Approche comportementale

Cette approche se base sur l'hypothèse que l'exploitation d'une faille du système nécessite une utilisation anormale de ce système, et donc un comportement inhabituel de l'utilisateur. Elle cherche donc à répondre à la question " le comportement actuel de l'utilisateur ou du système est-il cohérent avec son comportement passé ? ".

5.3.2.1 Inconvénient

Cette approche pose quelques problèmes, parmi lesquels nous citons :

- Choix des mesures à retenir pour un système cible donné délicat.
- Difficulté à dire si les observations faites pour un utilisateur particulier correspondent à des activités que l'on voudrait prohiber.
- Pour un utilisateur au comportement erratique, toute activité est normale.
- Utilisateur pouvant changer lentement de comportement dans le but d'habituer le système à un comportement intrusif.

5.3.3 Comparaison des deux approches

Chacune de ces approches peut conduire à des faux positifs (détection d'attaque(s) en absence d'attaque) ou à des faux négatifs (absence de détection en présence d'attaque(s)) :

- Un outil basé sur l'approche comportementale génère une alarme dès qu'il détecte un comportement non appris. Or, la déviation du comportement observé peut être due à une évolution naturelle de l'environnement et du système : c'est un faux positif. En outre, l'attaquant (utilisateur interne malicieux) peut modifier lentement son comportement afin de parvenir à un comportement intrusif qui, ayant été progressivement appris, ne sera pas détecté : c'est un faux négatif.
- Le risque de faux positifs est moindre avec l'approche par scénario car l'activité litigieuse est décrite dans la base d'attaque. Cependant, la qualité de la signature est importante : si elle n'est pas assez précise, elle peut également conduire à de nombreux faux positifs.

Enfin, bien évidemment, si la signature de l'attaque n'est pas dans la base (comme c'est le cas pour les nouvelles attaques), l'attaque en question ne sera pas détectée (c'est là un problème similaire à ce que l'on connaît avec les bases de signatures de virus).

5.3.4 Systèmes hybrides

Pour tenter de compenser quelques inconvénients de chacune des techniques, certains systèmes utilisent une combinaison de la détection d'anomalies et de la détection de malveillances. Par exemple, dans [10] Catalin et Dumitrescu ont proposé INTCTD une nouvelle approche de détection d'intrusion dans les réseaux P2P.
INTCTD est un système réparti basé sur les réseaux de neurones pour la détection des anomalies du trafic de réseaux et pour modifier dynamiquement les politiques d'accès aux ressources du réseau.
Ce travail vise principalement la possibilité de détecter des attaques à distance contre les réseaux P2P d'une source extérieure ou de l'intérieure du système.
Le but de cette approche est de traiter les données brutes du trafic de réseau et de détecter

n'importe quel comportement susceptible.
En outre, une fois qu'une nouvelle politique d'accès est apprise, elle est également échangée avec tous les autres participants du réseau pour améliorer la sécurité globale du système.

5.4 Quelques systèmes existants de détection d'intrusions sur les réseaux P2P

La première génération des IDS ont eu seulement deux composants dans leur architecture : composant de collection de données et un analyseur centralisé simple (architecture hiérarchique statique). Des données des audits et les paquets surveillés du trafic réseau ont été rassemblés et envoyés à un analyseur central. Dans un tel IDS, l'analyseur central peut être vu comme un point d'échec.
La plupart des systèmes d'intrusion actuellement utilisés comme Emerald [57], l'AAFID [27], GrIDS [78] etc., suivent une structure hiérarchique.

Dans ce qui suit, nous décrivons brièvement des IDS actuellement en service.

5.4.1 EMERALD

EMERALD [57] a un framework hiérarchique pour la détection distribuée d'intrusion. Il a les moniteurs de service, au niveau le plus bas, qui surveillent le trafic du réseau. L'information recueillie par ces moniteurs est alors communiquée aux moniteurs de client qui la transmettent alors aux moniteurs de domaine, qui la transmettent enfin aux moniteurs d'entreprise.
La communication entre les moniteurs est basée abonnement et les données de chaque moniteur sont rassemblées, analysées, réduites et puis transmises aux couches supérieures.

5.4.2 AAFID

AAFID [27] emploie les agents autonomes à des niveaux plus bas pour rassembler, analyser et filtrer les données. Ces agents sont courus et commandés à chaque hots par des transceivers locaux. Les transceivers font rapport à leur tour aux moniteurs.
Un moniteur simple peut commander plusieurs transceivers situés aux différents hots et peut lui-même faire rapport à d'autres moniteurs au-dessus de lui.
Surveiller de cette manière, donne une vue globale du réseau qui facilite le procédé de décision.

5.4.3 GrIDS

GrIDS [78] construit des graphiques pour analyser et représenter l'activité des hots et du réseau.
Elle présente l'organisation comme une hiérarchie des départements et de chaque département crée son propre graphique pendant l'analyse.
N'importe quel matériel concernant d'autres départements est passé vers le haut de la hiérarchie.

Les systèmes mentionnés ci-dessus de détection d'intrusion sont statiques et l'échec des nœuds à des niveaux plus élevés dans la hiérarchie peut s'avérer très coûteux.

5.4.4 MAIDS

Ramachandra et Delbert présentent un système de détection d'intrusion basé sur les agents mobiles et le concept de neighborhoodwatch [63].
Un voisinage virtuel est créé quand les voisins s'échangent les requêtes entre eux.
Quand une intrusion se produit, les pairs observent cette intrusion et informent les autres au sujet de cette intrusion et agissent collectivement.
Chaque site envoie périodiquement des agents mobiles pour visiter et vérifier ses voisins et revenir avec un rapport d'état.
Quand on observe le comportement contradictoire ou anormal, l'observerneighbor lance un processus de vote pour agir contre le site compromis.

5.4.5 INDRA

Le système INDRA [25] est basé sur un ensemble de démons, mis en application, spécialisés et de propriété industrielle, qui fournissent des services de sécurité tels que la détection des intrusions et le contrôle d'accès aux ressources.

5.4.6 NetBiotic et Trust-Aware

Vlachos et ses collègues présentent NetBiotic [82], un système qui surveille l'activité d'attaque dans chaque hots et emploie des mécanismes P2P pour établir une vue globale des attaques dans le réseau entier.

NetBiotic surveille le changement de la fréquence d'attaque et propage seulement cette information, tandis que le système Trust-Aware [7], envoie des alarmes à d'autres pairs directement, réalisant un temps de réaction plus rapide.
Trust-Aware est un IDS qui transfère la fonction de détecter des alertes à partir des pairs de sonde aux pairs qui pourraient corréler l'information reçue des sources très clairsemées.
Bien que l'architecture de NetBiotic soit semblable à celle de Trust-Aware, l'approche d'envoyer les messages d'avertissement diffère de manière significative.

5.4.7 SNORT

Snort [48] est un IDS (Intrusion Detection System) permet, par exemple, de repérer rapidement certains trafics P2P. Il possède actuellement une base de signatures pour Napster, Gnutella, BitTorent, Fastrack/Kazaa/Morpheus.

5.4.8 MAPIDS

L'approche MAPIDS [34] (*Mobile Agent-based Peer-to-peer Intrusion Detection System*) présente une détection efficace des activités méfiantes. MAPIDS lance un processus de vote pour prendre une décision collective et pour faire plus d'actions.
Contrairement aux autres systèmes de détection d'intrusion basés sur la décision collective, MAPIDS consomme le moins, en bande passante, énergie et préserve le système contre le problème de clairsemé des nœuds.

5.5 Conclusion

La détection peut être faite en temps réel ou en tant qu'analyse post-mortem. Si la détection est en temps réel, le programme peut donner l'alerte, auquel cas le personnel qualifié pourra tenter de remédier à l'intrusion, en coupant le réseau ou en remontant la piste. Mais il peut probablement arriver après la fin de l'intrusion.
Dans tous les cas, les techniques de détection d'intrusion seules n'empêcheront pas l'intrusion.

Dans ce chapitre, nous avons présenté quelques méthodes de détection des attaques dans les réseaux P2P, puis, nous avons définit quelques approches de détection d'intrusion. Enfin, nous avons cité quelques systèmes existants de détection d'intrusion dans les réseaux P2P.

Une proposition d'une méthode de détection de l'attaque Eclipse sur le réseau eDonkey fera l'objet du chapitre suivant.

Chapitre 6

Un algorithme de détection de l'attaque Eclipse sur le réseau eDonkey

6.1 Introduction

Dans le présent chapitre, nous allons analyser deux scénarios différents par lesquelles un attaquant peut mener une attaque Eclipse sur le réseau eDonkey. Cette distinction est basée sur le nombre de serveurs malveillants qui participent dans l'attaque. La première traite le cas où l'attaque est menée par un seul serveur et la deuxième, qui est une généralisation du premiers cas, traite le cas de plusieurs serveurs attaquants. Nous allons aussi proposer une solution de détection pour chaque attaque.

6.2 Protocole de communication dans un réseau eDonkey

Avant de présenter en détail les deux scénarios d'attaque et leur algorithme de détection approprié, nous allons présenter un petit rappel sur la communication entre un serveur eMule et un client(voir aussi chapitre 2). Plus de détails sur le format des messages échangés est présenté dans l'annexe. Quelques étapes du protocole de communication entre un serveur eMule et un client sont :

- Au démarrage le client se connecte à un serveur eMule en utilisant une connexion TCP. Le serveur donne au client un ID qui n'est valide que pour la durée de vie de cette connexion du client avec le serveur.

– Le client dépose ses fichiers chez le serveur via un message Offer_Files (voir annexe : Tableau 6.6). Chaque fichier suit le format décrit dans l'annexe : Tableau 6.7. Puis le client demande au serveur la liste des serveurs. Le serveur lui répond par une liste de serveurs via un message Server_List (le Tableau 6.1 ci-dessous montre un exemple de message Server_List).

– Une transaction de recherche commence généralement par une requête de recherche Get_Sources (annexe : Tableau 6.9) d'un fichier spécifique qui sera répliqué par une liste de sources (@IP, N°Port) à partir desquels le client demandeur peut télécharger ce fichier (message Found_Source annexe : Tableau 6.10).

Il faut noter l'existence d'un champ Entry_count dans le message Server_List qui indique le nombre de serveurs décrit dans ce message et le champ Server_entries - dans le même message - qui est un descripteur des serveurs qui représente chaque serveur par une adresse IP sur 4 octets et un numéro de port sur 2 octets. Le tableau 6.1 montre un exemple d'un message Server_List qui contient deux serveurs.

Name	Protocol	Size	Type	Entry Count	Server entries
Server_List	0xE 3	4	0x32	000000002	Server1 @IP1, num Port1 Server2 @IP2, num Port2

TAB. 6.1 – Exemple d'un message Server_List.

Notons aussi que le champ Sources_Count et List_Of_Sources dans le message Found_Source indique respectivement le nombre des sources et la liste des sources résultant d'une requête de recherche (Tableau 6.2 montre un exemple d'un message Found_Source).

Name	Protocol	Size	Type	File Hash	Sources Count	List Of Sources
Found_Source	0xE 3	4	0x32	NA	000000003	Client 1@IP1, num Port1 Client 2@IP2, num Port2 Client 3@IP3, num Port3

TAB. 6.2 – Exemple d'un message Found_Source.

6.3 Scénarios de l'attaque Eclipse sur le réseau eDonkey

Selon le nombre d'attaquants participant dans l'attaque Eclipse, Nous distinguons entre deux scénarios d'attaques. Le premier scénario traite le cas de l'attaque Eclipse sur le réseau eDonkey menée par un seul serveur attaquant et le deuxième pour le cas de plusieurs serveurs attaquants.

6.3.1 Attaque menée par un seul serveur

Le but d'un serveur attaquant qui utilise l'attaque Eclipse est de cacher la partie du réseau qu'il surveille du reste du réseau . Cela peut être réalisé de la manière suivante :

1. Le serveur attaquant accepte de nouveaux clients mais il leur envoie une liste des serveurs (sous la forme indiquée sur le tableau 6.3) qui ne contient que l'adresse de l'attaquant. La figure 6.1 montre comment se fait l'échange des messages entre un client et un serveur attaquant lors de l'établissement de connexion.

Name	Protocol	Size	Type	Entry Count	Server entries
Server_List	0xE 3	4	0x32	000000001	Attacker @IP , Attacker num Port

TAB. 6.3 – Message Server_List d'un seul serveur attaquant.

2. Le serveur attaquant répond négativement aux requêtes de ses clients par des messages sous la forme indiquée sur le tableau 6.4 (message Not_Found), ou par des messages erronés (par exemple une adresse aléatoire d'un client sera affectée au champ List_Of_Sources du message Found_Sources des réponses au requêtes de recherche) comme le montre la figure6.2 :

Name	Protocol	Size	Type	File Hash	Sources Count	List Of Sources
Found_Source	0xE 3	4	0x32	NA	0	vide

TAB. 6.4 – Exemple d'un message Not_Found.

3. Le serveur attaquant ne répond pas aux requêtes UDP de recherche (c-à-d pas de réponse aux clients distants), mais il doit répondre aux requêtes Server_status_request, et Server_description_request (voir annexe tableau 6.11,6.12) pour garder sa popularité

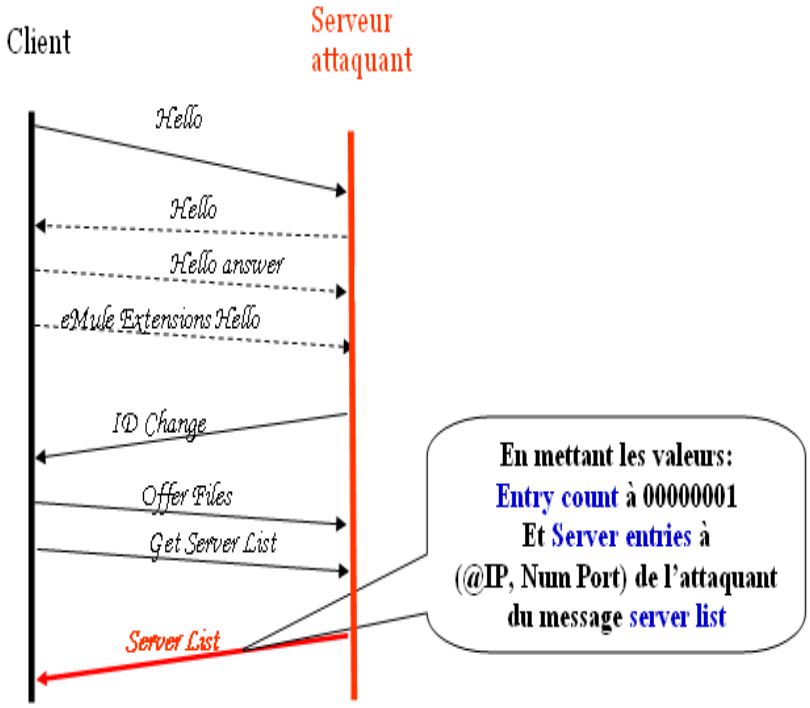

FIG. 6.1 – le serveur attaquant fait croire au client qu'il est le seul serveur dans le réseau

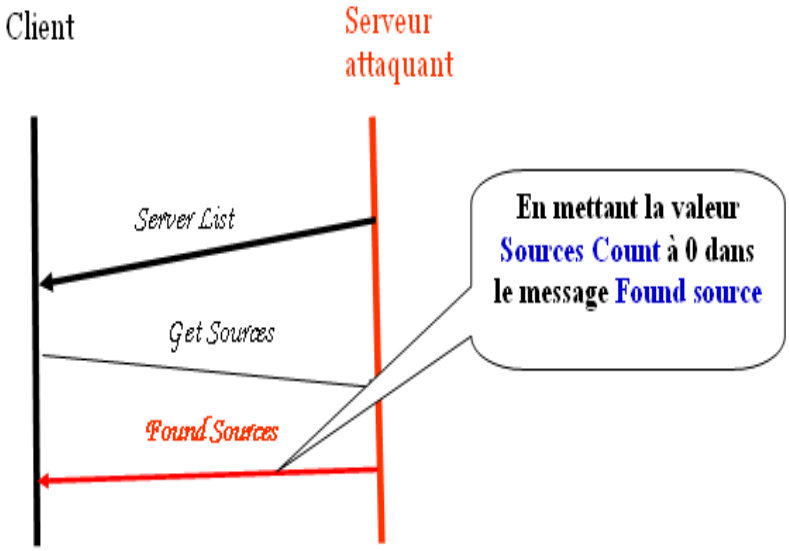

FIG. 6.2 – le serveur attaquant répond négativement à ses clients

dans tout le réseau (sinon les clients distants vont le supprimer de leurs Server_Lists).

4. Mise à jour des listes de serveurs des clients par une liste qui ne contient qu'une seule adresse et qui est l'adresse de l'attaquant pour assurer le non contact avec l'extérieur. Supposons que le serveur attaquant peut forcer l'attempt-counter d'un serveur dans la

liste des serveurs d'un client d'avoir la valeur limite ; alors, quand le client trouve que la valeur du compteur est égale à la valeur limite, il décidera directement de supprimer ce serveur de sa liste des serveurs.
Le serveur attaquant répète la dernière supposition pour tout le reste des serveurs de la liste de serveurs du client autre que lui.
Une autre manière de supprimer un serveur, dans le cas d'un seul serveur attaquant, est de fermer le port UDP du client ; ce qui déclenche la mise à jour du compteur attempt-counter et après un certain moment le client va supprimer tous les serveurs sauf l'attaquant (le serveur sur lequel le client est relié par une connexion TCP).

6.3.2 Attaque menée par plusieurs serveurs

Nous supposons cette fois ci que nous avons plusieurs serveurs attaquants et que ces serveurs se connaissent entre eux (chaque serveur attaquant a une liste qui contient les adresses des autre serveurs attaquants, cette liste est identique pour tous les serveurs attaquants). Le tableau 6.5 montre un exemple de Server_List de deux serveurs attaquant.

Name	Protocol	Size	Type	Entry Count	Server entries
Server_List	0xE 3	4	0x32	000000002	Attacker1 @IP1, num Port1 Attacker2 @IP2, num Port2

TAB. 6.5 – Message Server_List de deux serveurs attaquants.

Ils mènent une attaque par un re-acheminement des requêtes et l'isolation du reste du réseau les parties dont ils surveillent et cela par :

- La mise à jour des listes de serveur des clients reliés aux attaquants par une liste qui ne contient que les adresses des serveurs attaquants.

- Chaque serveur attaquant procède de la même manière que le serveur attaquant décrit dans le premier scénario avec une liste de serveurs contenant les adresses des serveurs attaquants.

6.4 Proposition d'un algorithme de détection de l'attaque Eclipse sur le réseau eDonkey

Notre solution se nome " Eclipse Detection in the eDonkey Network (EDEN) ". Elle est basée sur la solution proposée par Reinier Schoof et Ralph Koning dans [71] pour détecter les botnets dans les réseaux Peer To Peer.

Par analyse de l'attaque Eclipse sur le réseau eDonkey d'après le scénario (ci-dessus), les caractéristiques suivantes sont découvertes, ce qui rend la détection de l'attaque facile.

6.4.1 Ports TCP et UDP ouvert (test 1)

Comme nous avons vu dans notre étude sur l'attaque Eclipse - parmi les attaques spécifiques aux réseaux P2P - (en particulier le réseau eDonkey), des ports spéciaux (port TCP 4661, et UDP 4665), doivent être ouverts pour permettre la connexion et la communication entre les paires (clients et serveurs). Cela facilite la détection des serveurs attaquants, par une surveillance du trafic de données à partir et vers ces ports.
Aussi le calcul de quelques paramètres (par exemple le nombre des paquets UDP) avant et après le lancement de l'attaque va nous conduire vers des résultats qui nous confirme les résultats obtenus par notre algorithme (la troisième caractéristique) en vu de minimiser les faux positifs (présence d'alerte en absence d'attaque).

Cas particulier

Si nous prenons le cas de la mise à jours des listes de serveurs par la fermeture des Ports UDP (des clients du serveur attaquant) - comme nous avons indiqué dans le scénario en cas d'un seul serveur attaquant et seulement dans ce cas - et la liste des serveurs ne contient que l'adresse de l'attaquant, nous pouvons conclure que ce serveur est malicieux.

6.4.2 Des réponses négatives aux requêtes de recherche (massage Not_Found)test 2

Après le lancement de l'attaque Eclipse sur un réseau eDonkey selon le scénario, un taux de messages Not_found accroît rapidement dans le réseau. Alors, nous pouvons utiliser cette caractéristique comme paramètre qui facilite la détection de l'attaque.

6.4.3 Découverte des serveurs et utilisation d'une liste de serveurs attaquants (test 3)

Ce point est basé sur le faite qu'un attaquant qui utilise Eclipse vise principalement la liste des serveurs dans un réseau eDonkey, comme c'est indiqué dans les deux scénarios précédemment décrits.
Telle que dans le cas d'un seul serveur attaquant, sa liste des serveurs ne contient que l'adresse de l'attaquant. Et dans le cas de plusieurs serveurs attaquants, leurs listes des serveurs contiennent seulement leurs adresses (c-à-d il ont tous la même liste).

Avant d'exécuter notre algorithme de détection de l'attaque Eclipse sur le réseau eDonkey, un traitement des listes des serveurs doit être fait.
Commençons par la récupération des listes de serveurs (Figure 6.3) comme suit :

1. Pour chaque serveur de la liste Server_List d'un client détecteur, le client détecteur envoi un message Get_Server_List lui demandant d'envoyer sa liste de serveurs.
2. A la réception de Get_server_list, chaque serveur répond par l'envoi d'un message Server_List qui inclut sa liste de serveurs.
   Exemple : list_S1={S1,S2},
   list_S2={S1,S2},
   list_S3={S1,S2,S3,S5},
   list_S4={S1,S2,S3,S4},
   list_S5={S5,S3,S4},
   Supposons dans notre exemple que S1 et S2 sont des attaquants qui se connaissent entre eux (Figure 6.3).
   Après la récupération des listes de serveurs, des calculs sont nécessaires avant l'exécution de l'algorithme de détection.

3. Calculons d'abord la taille de la liste de chacun des serveurs.
   Exemple : taille_list_S1= 2 ,
   taille_list_S2= 2 ,
   taille_list_S3= 4 ,
   taille_list_S4= 4 et taille_list_S5= 3 ,
   A : représente la variable taille_list_S d'un serveur quelconque (A=2,3,4, dans l'exemple).
4. Calculer le nombre **B** des serveurs qui ont la même taille_list_S
   Exemple : Pour A= 2 nous avons S1 et S2 c-à-d B= 2,
   Pour A= 4 nous avons S3 et S4 c-à-d B= 2,
   Pour A= 3 nous avons S5 c-à-d B= 1,

Par ceci le client détecteur a construit l'ensemble des couples (A,B), sur lesquels il va appliquer l'algorithme de détection de l'attaque Eclipse sur le réseau eDonkey par la suite.

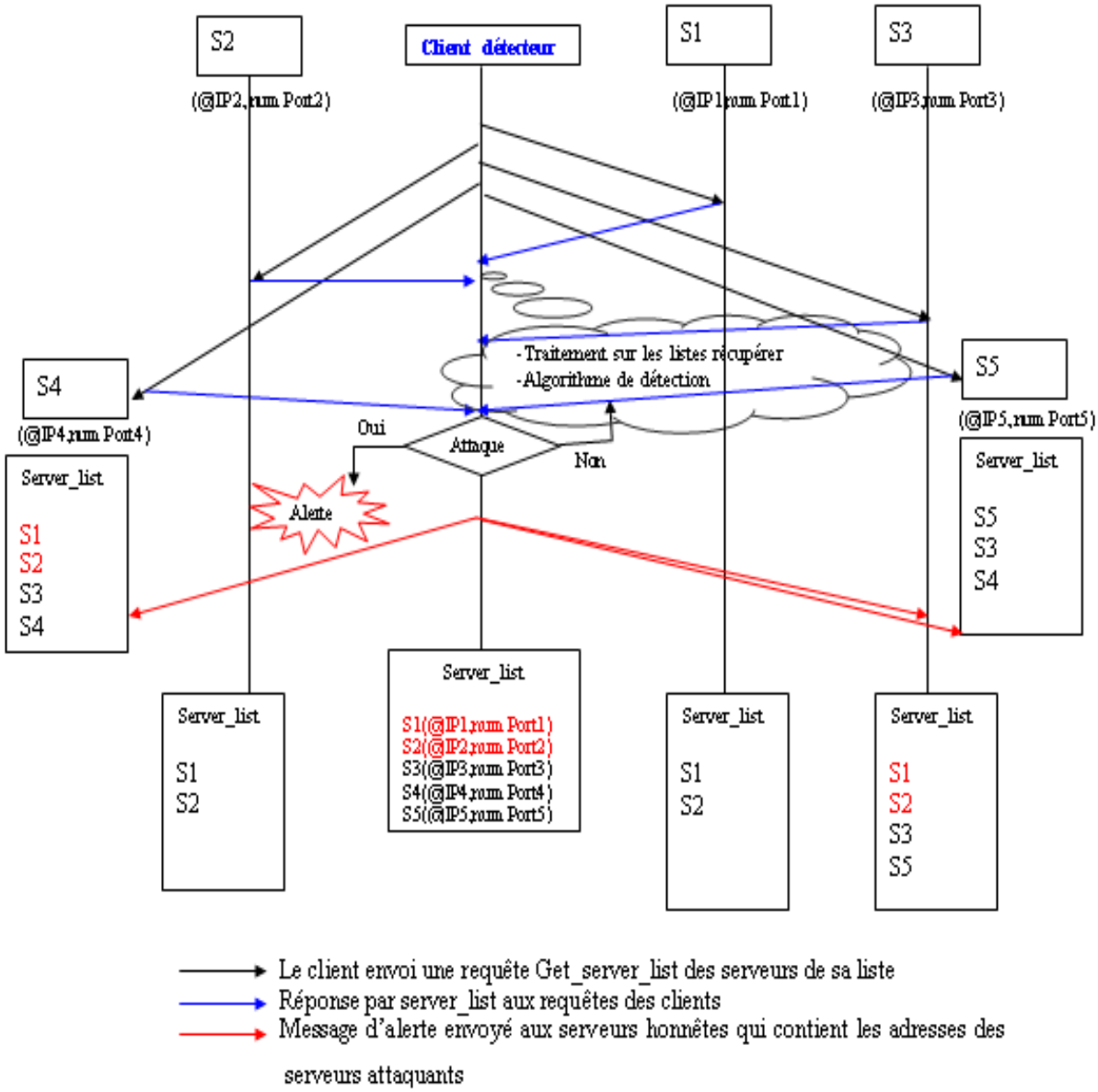

FIG. 6.3 – Exemple illustratif1

5. Enfin, nous allons présenté notre algorithme (algorithme6.1)de détection de l'attaque Eclipse sur le réseau eDonkey comme suit :

   Hypothèse

   Supposons que la liste de serveurs du client détecteur contient les adresses de tous les serveurs du réseau.

Algorithm 6.1 Algorithme de détection de l'attaque Eclipse sur le réseau eDonkey

Var :
N : integer {*Représente le nombre de combinaisons des B serveurs en A serveurs.*}
Iter : Integer {*variable Fin de la boucle Répéter.*}
A : Integer { *la variable taille_list_S de n'importe quel serveur (A= 2,3,4, dans l'exemple)*}
B :Integer {*le nombre des serveurs qui ont la même taille_list_S*}
Keep_attack_servers : List { *sauvegarde la liste des serveurs attaquants*}
Keep_servers : List { *sauvegarde la liste des serveurs honnêtes*}
servers_list : List { *représente la liste des serveurs initiale du client détecteur*}
$list\_S_i$: List {*list_S de l'un des serveurs actuel comme list_S1 = {S1,S2}*}
Init :
Les variables Keep_attack_servers, Keep_servers sont initialisé par l'ensemble vide ;
Iter : 1

si $(A = B)$ **alors**
  Nous allons vérifier que les serveurs qui ont des listes de serveurs de taille égale sont eux même les éléments de ses listes.
  si oui **alors**
    $Keep\_attack\_servers \leftarrow list\_S_i$
    $Keep\_servers \leftarrow server\_list - keep\_attack\_servers$
    Déclencher une alerte { *une fonction à détallé par la suite*}
  fin si
  { *le cas d'un seul serveur attaquant est inclus ici (A=1 et B=1)*}
sinon
  si $(A = 1)$ **alors**
    $Keep\_attack\_servers \leftarrow Union\_des\_B\_list\_S_i$
    $Keep\_servers \leftarrow server\_list - keep\_attack\_servers$
    Déclencher une alerte
  sinon
    si $(B > A)$ **alors**
      Nous allons calculer les combinaisons possible des B serveurs en A serveurs, soit N le nombre de combinaisons, et pour chaque combinaisons :
      répéter
        vérifier que les serveurs qui ont des listes de serveur de taille égale sont eux même les éléments de leurs listes.
        si oui **alors**
          $Keep\_attack\_servers \leftarrow list\_S_i$
          $Keep\_servers \leftarrow server\_list - keep\_attack\_servers$
          Déclencher une alerte
        sinon
          combinaison suivante.
        fin si
        Iter++.
      jusqu'à (Iter>N)
    fin si
  fin si
fin si

6.4.4 Déclenchement d'alerte

Si détection réussie Alors :

Le client détecteur doit avertir le reste du réseau qu'il existe des serveurs malveillants dans le réseau et cela comme suit :

- Le client détecteur envoi un message d'alerte à chacun des serveurs de la liste keep_servers (serveurs honnêtes). Le message d'alerte contient l'ensemble des adresses des attaquants sauvegardé dans Keep_attack_servers (voir Figure 6.3).

- Les serveurs honnêtes vont supprimer les adresses des serveurs attaquants de leurs Serevr-Lists et mettre à jour les listes de leurs clients.

- Après un certains moment, les attaquants vont constater qu'ils ne sont plus accessibles de l'extérieur, donc ils se déconnectent du réseau. Alors, leurs clients vont attendre un time-out des requêtes de Keep-alive (status-request et description-request), et puis ils se déconnectent de leurs serveurs (serveurs attaquants).

6.4.5 Discussion et résultats

Notre algorithme génère une alerte lors de la détection de l'attaque Eclipse sur le réseau eDonkey, et cela dans les trois cas suivants :

6.4.5.1 Cas A=B

(A = B) veut dire que nous avons B serveurs qui ont des listes de serveurs de taille égale. La taille d'une de leurs listes est égal à A.
Alors, nous allons vérifier que les B serveurs qui ont des listes de serveurs de taille égale sont eux même les éléments de leurs listes. Ce qui caractérise une attaque Eclipse sur le réseau eDonkey mené par plusieurs serveurs qui se connaissent entre eux et aussi pour le cas d'un seul serveur attaquant (A=1 et B=1).
Dans ce cas la liste des serveurs attaquants est la liste de un des B serveurs attaquants puisqu'ils ont tous la même liste.

La figure 6.3 montre une exécution de notre algorithme dans le cas d'une attaque Eclipse mené par 2 serveurs S1 et S2. Et la figure 6.4 nous montre l'état du réseau après le déclenchement de l'alerte et la suppression des adresses des serveurs attaquants de toutes les autres listes.

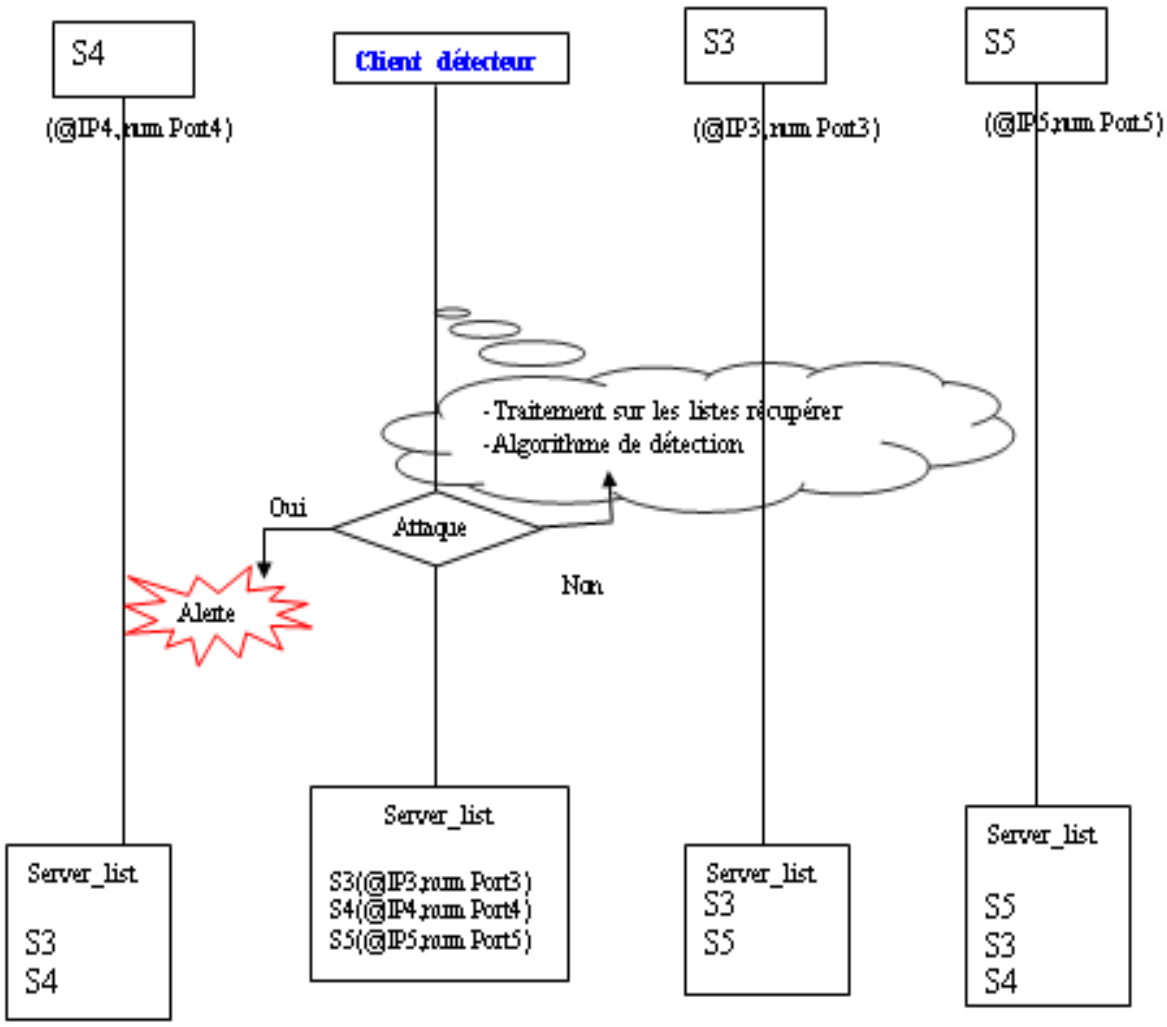

FIG. 6.4 – Exemple illustratif2

6.4.5.2 Cas A = 1 et B > 1

Ce cas veut dire que nous avons B serveurs de taille liste égale à A=1, et B supérieur à un. Donc, nous somme dans le cas de plusieurs serveurs attaquants qui ne se connaissent pas entre eux (voir figure 6.5).
Dans ce cas la liste des serveurs attaquants est calculée par l'union des B listes d'un seul élément pour chacune. La figure 6.4 nous montre l'état du réseau après le déclenchement de l'alerte.

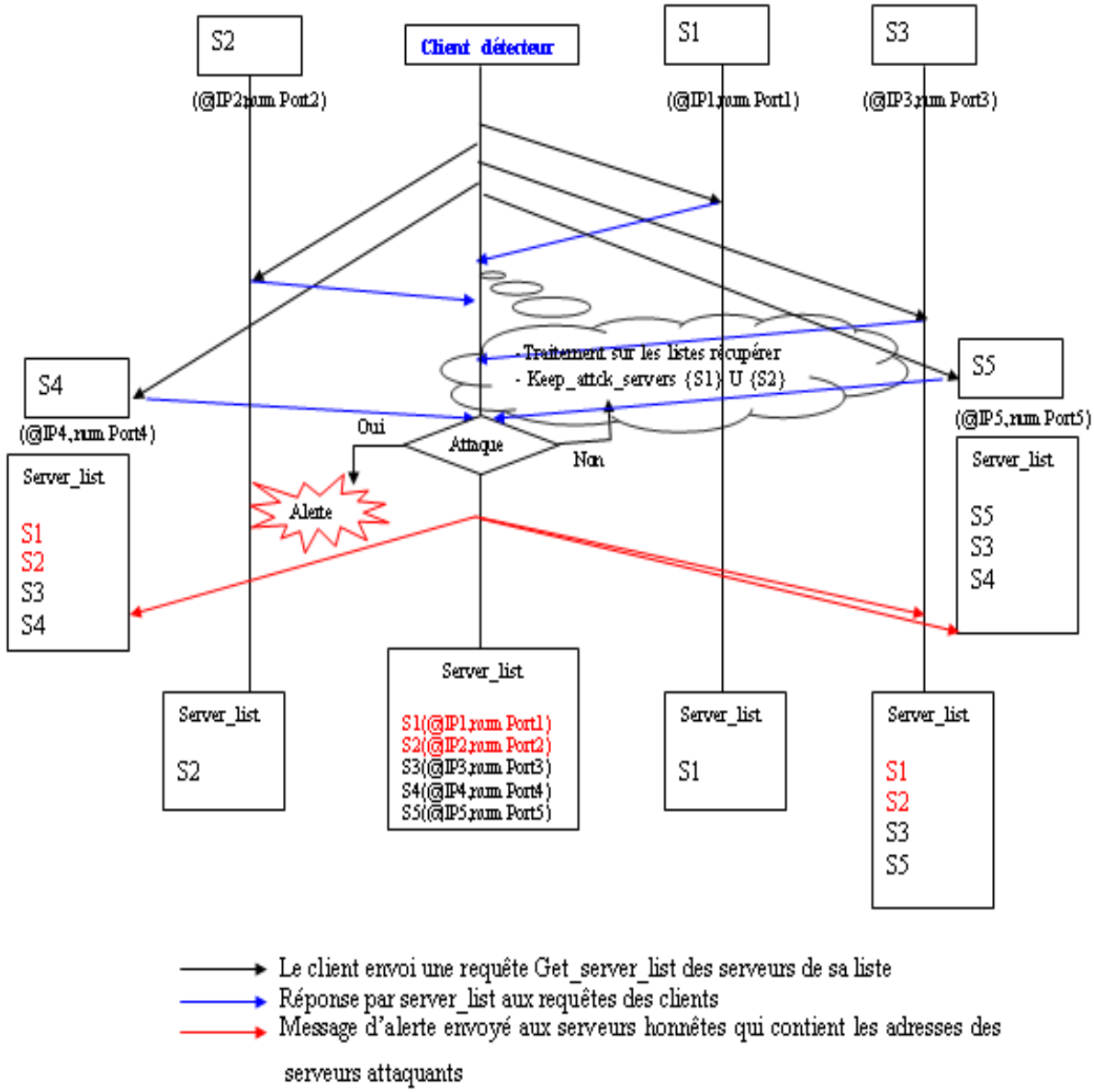

FIG. 6.5 – Exemple illustratif3

6.4.5.3 Cas B > A

Ce cas veut dire que nous avons B serveurs qui ont des listes de serveurs de taille égale et la taille d'une de leurs listes est égal à A et B supérieur à A.

Donc, il y a une forte chance d'avoir une combinaison de A serveurs parmi les B serveurs qui sont des attaquants.
Alors, nous allons calculer toutes les combinaisons possible des B serveurs en A serveurs et pour chaque combinaisons, nous allons appliqué les mêmes étapes de l'algorithme comme dans

le cas de (B = A) car le B de chaque combinaisons est égale à A.
Prenons l'exemple suivant qui explique ce cas de figure :

- Soit les trois serveurs S1,S2,S3 de liste de serveurs {S1,S2}, {S1,S2},{S3,S2}respectivement.

- Remarquons que nous avons trois serveurs de taille liste égale à 2, c-à-d B=3 et A=2 (B>A).

- Calculons les différentes combinaisons possibles des B serveurs en A serveur, comme suit :
  C1 : S1 et S2 (list_S1={S1,S2} et list_S2={S2,S1}) ;
  C2 : S1 et S3 (list_S1={S1,S2} et list_S3={S3,S2}) ;
  C3 : S2 et S3 (list_S2={S2,S1} et list_S3={S3,S2}) ;
  Remarquons qu'après l'exécution des étapes de notre algorithme sur la première combinaison C1 (list_S1={S1,S2} et list_S2={S2,S1}), une alerte va être déclanchée.

6.4.6 Exemple de fausse alerte

Cette méthode nous déclenche des faux positifs, et cela dans le cas où -par exemple- nous avons trois serveurs S1, S2, S3 honnêtes avec la liste des serveurs suivante = {S1, S2, S3}, alors que ce n'est pas une attaque.
Pour remédier à ce problème nous devons faire autres tests comme suit :

Avant le déclenchement d'une alerte nous devons appliquer d'autres tests (test1 et 2) pour confirmer que les serveurs détectés sont réellement des attaquants qui appliquent une attaque Eclipse sur le réseau eDonkey et non pas autre chose.
Aussi, si nous appliquons test1 (surveillance des ports TCP et UDP des peers) ou test 2 (nombre de paquets Not_Found), nous pouvons avoir un nombre énorme de fausses alertes, alors que notre but est de donner des résultats presque exactes (détection d'attaque réussie) dans les plus bref délais et de minimiser au maximum le nombre des faux positives générer.
Donc en augmentant le nombre de tests, on s'éloignant des fausses alertes.

6.5 Conclusion

Dans ce chapitre, nous avons analysé l'attaque Eclipse dans le cadre du réseau eDonkey. Nous avons identifié deux scénarios différents par lesquelles une attaque Eclipse peut être menée.

Le premier scénario traite le cas de l'attaque Eclipse sur le réseau eDonkey menée par un seul serveur attaquant et le deuxième pour le cas de plusieurs serveurs attaquants.
Le but d'un serveur attaquant qui utilise Eclipse est de cacher du reste du réseau la partie dont il surveille. Un attaquant vise principalement la liste des serveurs dans un réseau eDonkey.

Pour faire face à cette attaque nous avons proposé une solution basée sur la taille des listes de serveurs des serveurs d'une liste d'un client détecteur. Les résultats de cette solution ont montré qu'elle permet de détecter l'attaque Eclipse des qu'elle soit et d'isoler les serveurs attaquants du reste du réseau. Mais cette solution déclenche des faux positifs que nous proposons de les minimiser par un profilage des utilisateurs des ports TCP et UDP et du trafic du réseau.

Conclusion et Perspectives

Les Peer-to-Peer sont aujourd'hui de plus en plus présents dans le monde informatique, que ce soit pour les simples utilisateurs à travers les réseaux d'échanges de fichiers, de messageries instantanés ou de voix sur IP, ou encore au sein des entreprises, pour la mise en place d'applications réparties. D'une manière générale, les P2P actuels souffrent de nombreux problèmes tels que les fichiers truqués, virus et autres...A cela s'ajoute le faite qu'il ne s'agit pas, la plupart du temps, de P2P " purs ", mais de P2P " hybride ", s'appuyant donc sur l'utilisation de serveurs centraux qui, malgré leurs avantages, constituent une véritable faiblesse (comme c'est le cas pour le réseau eDonkey).

Dans ce mémoire, nous avons étudié la détection d'attaques dans les réseaux P2P et bien particulièrement le réseau eDonkey. Nous avons présenté une synthèse des différentes attaques qui peuvent être menées sur ce type de réseau en se focalisant sur l'attaque Eclipse. Pour cette dernière, nous avons identifié deux scénarios possibles sur le réseau par lesquels un Eclipse peut être créé dans le réseau.

Le premier scénario traite le cas de l'attaque Eclipse sur le réseau eDonkey menée par un seul serveur attaquant et le deuxième pour le cas de plusieurs serveurs attaquants.
Le but d'un serveur attaquant qui utilise Eclipse est de cacher du reste du réseau la partie dont il surveille et ceci par la mise à jour des listes de serveurs de ses clients par une liste de serveurs qui ne contient que l'adresse de l'attaquant. Aussi, Le serveur attaquant ne répond pas aux requêtes UDP de recherche pour assurer le non contact de ses clients avec le reste du réseau, mais il doit répondre aux requêtes de Keep_alive pour garder sa popularité dans tout le réseau.

Dans le deuxième cas, chaque serveur attaquant procède de la même manière ce que le serveur attaquant décrit dans le premier scénario avec une liste de serveurs contenant les adresses des attaquants.

Pour faire face à cette attaque nous avons proposé une solution basée sur la taille des listes de serveurs des serveurs d'une liste d'un client détecteur. Les résultats de cette solution ont

montré qu'elle permet de détecter l'attaque Eclipse dès qu'elle y ait et d'isoler les serveurs attaquants du reste du réseau. Mais cette solution déclenche des faux positifs que nous proposons de les minimiser par un profilage d'utilisateurs des ports TCP et UDP et du trafic du réseau.

Comme perspectives nous envisageons de réaliser un simulateur muni d'une plat-forme d'un réseau eDonkey et l'implémentation de mécanismes de sécurité dans ce réseau qui va nous permettre de simuler notre contribution.
Nous envisageons aussi dans un travail futur, d'étendre nos études à d'autres types d'attaques afin d'élaborer une architecture de sécurité pour le réseau eDonkey.

Liste des Abbreviations

P2P : Peer To Peer
RPC : Remote Procedure Call
IP : Internet Protocol
BFS : Breadeth First Search
DFS : Depth First Search
TCP : Transmission Control Protocol
UDP : User Datagram Protocol
RSA : Rivest Shamir Adleman
RIAA : Recording Industry Association of America
MPAA : Media Player Association of America
ID : IDentificateur
MitM : Man In The Middle
DES : Data Encryption Standard
3DES : Le Triple DES
AES : Advanced Encryption Standard
SHA1 :Secure Hash Algorithm1
MD4 : Message Digest 4
MD5 : Message Digest 5
CA : Certificat Authority
IDS : Intrusion Detection System
MAIDS : Multi-Agents Intrusion Detection System
DoS : Denial Of Service
DDoS : Distributed Denial Of Service
URL :Uniform Resource Locator
CPU : Central Process Unit

ANNEXE

Dans l'annexe suivante nous citons quelques formats de messages échangés lors de la communication entre les serveurs et les clients eMule.
pour plus de détaille sur ces messages et d'autres, voir l'article "The eMule Protocol Specification[88]".

Message Offer_Files 6.6

Ce message est utilisé par le client pour décrire les fichiers qui sont à sa disposition dans le but de les rendre accessible au téléchargement.
Dans ce cas le client hash ses fichiers avant de les déposer. Le message Offer_Files est envoyé juste après l'établissement de connexion. Le tableau 6.7 décrit le format d'une entrée de la liste des fichiers.

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.

FIG. 6.6 – Message Offer_Files

Format d'une entrée de la liste des fichiers 6.7
Le tableau suivant décrit une entrée de la liste des fichiers. Plusieurs entrées peuvent co-exister 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

FIG. 6.7 – Format d'une entrée de la liste des fichiers

Liste des serveurs (message Server_List6.8)

Est un message envoyé du serveur au client, ce message contient des informations additionnelles sur les autres serveurs du réseau pour étendre les listes de serveurs des clients. La taille de ce message est variable (dépend au nombre des serveurs transmis).

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	0x32	The value of the OP_SERVERLIST opcode
Entry Count	1	NA	The number of servers described in this message.
Server entries	(Entry Count)*6	NA	Server descriptor entries each entry size is 6 bytes and contains 4 bytes IP address and then 2 byte TCP port.

FIG. 6.8 – Message Server_List

Message Get_Source 6.9

Est un message envoyé du client au serveur pour demander une source (des 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

Fig. 6.9 – Message Get_Source

Message Found_Source 6.10

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 opcode
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

Fig. 6.10 – Message Found_Source

Message Status_request6.11

Le message Satus_request est envoyé périodiquement aux serveurs. Ce message contient un challenge aléatoire sur 4 octets qui doit être echoé par le serveur.

Name	Size in bytes	Default Value	Comment
Protocol	1	0xE3	
Type	1	0x96	The value of the OP_GLOBSERVSTATREQ opcode
Challenge	4	NA	A unsigned integer challenge sent to the server, used for reply verification (the corresponding variable is called 'time' on the client)

FIG. 6.11 – Message Server_Statut_request

Message Server_description_request 6.12

Le message Server_description_request 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

FIG. 6.12 – Message Server_description_request

Bibliographie

[1] A.Abimbola, Q.Shi, and M.Merabti. Using intrusion detection to detect malicious peer-to-peer network traffic. Technical report, Liverpool John Moores University, School of Computing et Mathematical Sciences, 2003.

[2] A.Rowstron and P.Drushel. Pastry : Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of Middleware*, 2001.

[3] S.A. Baset and H.Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. *Department of Computer Science, Columbia University, New York*, 15 September 2004.

[4] Bittorrent. [web]. *http ://bitconjurer.org/BitTorrent/*, 2003.

[5] Nikita Borisov. Computational puzzles as sybil defenses. In *Sixth IEEE International Conference on Peer-to-Peer Computing*, 2006.

[6] M. Castro, P. Druschel, A.Ganesh, A. Rowstron1, and D. Wallach. Secure routing for structured peer-to-peer overlay networks. In *the 5th Usenix Symposium on Operating Systems, Design and Implementation*, December 2002.

[7] C.Duma, M.Karresand, N.Shahmehri, and G.Caronni. A trust-aware, p2p-based overlay for intrusion detection. In *Proceedings of the 17th International Conference on Database and Expert Systems Applications (DEXA'06)*, 2006.

[8] Guanling Chen and Robert S. Gray. Simulating non-scanning worms on peer-to-peer networks. *IEEE*, 2005.

[9] N. Christin, A. Weigend, and J. Chuang :. Content availability, pollution and poisoning in peer-to-peer file sharing networks. *In ACM E-Commerce Conference*, 2005.

[10] C.L.Dumitrescu. Intctd : A peer-to-peer approach for intrusion detection. In *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*, 2006.

[11] Michael Collins, Carrie Gates, and Gaurav Kataria. A model for opportunistic network exploits : The case of p2p worms. *National Science Foundation*, 2006.

[12] Landon P. Cox and Brian D. Noble. Samsara : Honor among thieves in peer-to-peer storage. *LP Cox, BD Noble - ACM SIGOPS Operating Systems - portal.acm.org*, 2003.

[13] A. Crespo and H.Garcia-Molina. Routing indices for peer-to-peer systems. In *the 22nd International Conference on Distributed Computing*, 2002.

[14] C.Siobhan. Dependability in peer-to-peer system. *IEEE internet computing*, 2004.

[15] C.Soldani. Peer to peer behaviour detection by tcp flow analysis. *University of liége*, 2004.

[16] C.V.Zhou, S.Karunasekera, and C.Leckie. A peer-to-peer collaborative intrusion detection system. *National ICT Australia, University of Computer Science and Software Engineering of Melbourne*, 2005.

[17] Neil Daswani and Hector GarciaMolina. Queryflood dos attacks in gnutella. *Journal of the ACM*, 2002.

[18] Neil Daswani, Hector Garcia Molina, and Zevrly Yong. Open problems in data sharing p2p systems. *Stanford university*, 2003.

[19] Jochen Dinger and Hannes Hartenstein. Defending the sybil attack in p2p networks : Taxonomy, challenges, and a proposal for self-registration. In *First International Conference on Availability, Reliability and Security*, 2006.

[20] John R. Douceur. The sybil attack. *SpringerLink ,First InternationalWorkshop, IPTPS Cambridge, MA, USA*, 2002.

[21] Marling Engle and Javed 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*, Novembre 2006.

[22] Lin Wang Helsinki. Attacks against peer-to-peer networks and countermeasures. *Seminar on Network Security*, 2006.

[23] I.Clarke, O.Sandberg, B.Wiley, and T. W.Hong. Freenet : A distributed anonymous information storage and retrieval system. *LNCS 2009*, page 46, 2001.

[24] I.Stoica, R.Morris, D.Karger, M.Frans Kaashoek, and H.Balakrishnan. Chord a scalable peer-to-peer lookup service for internet application. *ACM SIGCOMM, san Diego California*, 2001.

[25] R. Janakiraman, M. Waldvogel, and Q. Zhang. Indra : A peer-to-peer approach to network intrusion detection and prevention. In *Proceedings of 12th IEEE Workshop on Enterprise Security (WETICE)*, pages 226–231, June 2003.

[26] I. Jawhar and J. Wu. A two-level random walk search protocol for peer-to-peer networks. *the 8th World Multi-Conference on Systemics, Cybernetics and Informatics*, 2004.

[27] J.Balasubramaniyan, J. O.Garcia-Fernandez, D.Isacoff, E. H.Spafford, and D.Zamboni. An architecture for intrusion detection using autonomous agents. *Department of Computer Science, Purdue University*, 1998.

[28] Xing Jin, S.-H. Gary Chan, W.-P. Ken Yiu, Yongqiang Xiong, and Qian Zhang. Detection malicious hosts in presence of lying hosts in peer to peer streaming. *IEEE ICME*, 2006.

[29] J.Zimmermann and L.Mé. Les systèmes de détection d'intrusions : principes algorithmiques. Technical report, Supélec, équipe SSIR, 2001.

[30] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-yazti. A local search mechanism for peerto-peer networks. In *the 11th ACM Conference on Information and Knowledge Management*, 2002.

[31] K.Kant, R.Iyer, and V.Tewari. A framework for classifying peer-to-peer technologies. *IEEE/ACM,Symposium on Cluster Computing and the Grid*, 2002.

[32] K.Leibnitz, T.Hoffeld, N.Wakamiya, and M.Murata. On pollution in edonkey-like peer-to-peer file-sharing networks. *School of Information Science and Technology Osaka University,Osaka 565-0871, Japan*, 2005.

[33] K.Tutschku. measurement-based traffic protocole of the edonkey filesharing service. In *5th Passive and Active Measurement Workshop (PAM04)*, Antibes Juan-les-Pins, France, 2004.

[34] K.Xiao, J.Zheng, X.Wang, and X.Xue. A novel peer-to-peer intrusion detection system using mobile agents in manets. In *Proceedings of the Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'05)*, 2005.

[35] Uichin Leey, Min Choiz, Junghoo Choy, M. Y. Sanadidiy, and Mario Gerla. Understanding pollution dynamics in p2p file sharing. *the National Science Foundation under Grant No. 0221528 and the Korean Science and Engineering Foundation under Grant*, 2004.

[36] 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.

[37] J. Liang, R. Kumar, Y. Xi, and K. Ross. Pollution in p2p file sharing systems. *In IEEE INFOCOM*, 2005.

[38] Jian Liang, Naoum Naoumov, and Keith W. Ross. The index poisoning attack in p2p file sharing systems. *IEEE*, 2005.

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

[40] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys Tutorials, IEEE*, 7(2) :72– 93, 2005.

[41] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peerto-peer networks. *Proc. of the 16th ACM International Conference on Supercomputing*, 2002.

[42] L.Xiuqi and W.Jie. Searching techniques in peer-to-peer networks. *Computer Science and Engineering, Florida Atlantic University Boca Raton*, 2004.

[43] P. Maymounkov and D. Mazières. Kademlia : A peer-to-peer information system based on the xor metric. In *Processings of the IPTPS*, pages 53–65, February 2002.

[44] Kazaa media desktop. [web]. *http ://www.kazaa.com/*, 2001.

[45] A.J. Menezes and P.C. van Oorschot. *Handbook of Applied Crypthography*. S.A. Vanstone, 1998.

[46] M.Gupta, P.Judge, and M.Ammar. Reputation system for peer-to-peer networks. *ACM NOSSDAV'03,Monterey, California, USA*, June 2003.

[47] D. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S.Rollins, and Z. Xu. Peer-to-peer computing. Technical report, HP Laboratories Palo Alto, March 8th 2002.

[48] M.Roesch. Snort - lightweight intrusion detection for networks. *http ://www.snort.org*, 1999.

[49] Naoum Naoumov and Keith Ross. Exploiting p2p systems for ddos attacks. *Department of Computer and Information Science*, 2005.

[50] Napster. [web]. *http ://www.napster.com/*, 1999.

[51] W. Nejdl, M.Wolpers, W.Siberski, C.Schmitz, M.Schlosser, I.Brunkhorst, and A.Loser. Superpeerbased routing and clustering strategies for rdfbased peertopeer networks. *ACM*, May 20-24 2003.

[52] Seth James Nielson, Scott A. Crosby, and Dan S. Wallach. A taxonomy of rational attacks. *Proc. 4th IPTPS - cs.rice.edu*, 2005.

[53] Stephen Northcutt. *Networks intrusion détection*. New Riders Publishing, 2003.

[54] N.Stakhanova, S.Basu, J.Wong, and O.Stakhanov, editors. *Trust Framework for P2P Networks using Peer-Profile based Anomaly Technique*. Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'05), 2005.

[55] N.V.Padmanabhan, S.Ramabhadran, and J.Padhye. Netprofiler : Profiling wide-area networks using peer cooperation. *ACM*, 2003.

[56] O.Heckmann, A.Bock, A.Mauthe, and R.Steinmetz. The edonkey file-sharing network. *Technische Université Darmstadt*, 2005.

[57] P.A.Porras and P.G.Neumann. Emerald : Event monitoring enabling responses to anomalous live disturbances. In *National Information Systems Security Conference*, October 1997.

[58] Fasttrack peer-to-peer technology company. [web]. *http ://www.fasttrack.nu/*, 2001.

[59] T.R. Peltier, Justin Peltier, and John Blackley. *Information Security Fundamentals*. Press Company, 2005.

[60] Ph.Biondi. Architecture expérimentale pour la détection d'intrusions dans un système informatique. Technical report, webmotion, 2001.

[61] J.A. Pouwelse, P. Garbacki, D.H.J. Epema, and H.J. Sips. The bittorrent p2p file scharing system : measurments and analysis. *Delft University of Technology, the Netherlands*, 2005.

[62] Baptiste Pretre. Attacks on peer-to-peer networks. *Swiss Federal Institute of Technology (ETH) Zurich*, 2005.

[63] Geetha Ramachandran and Delbert Hart. A p2p intrusion detection system based on mobile agents. *Journal of the ACM*, 2004.

[64] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Proceedings of IEEE INFOCOM'02, NewYork*, Jun 2002.

[65] Thomas Reidemeister, Klemens B¨ohm, Erik Buchmann, and Paul A.S. Ward. Man-in-the-middle attacks in distributed hash-tables. *PAS Ward - ccng.uwaterloo.ca*, 2005.

[66] S. C. Rhea and J. Kubiatowicz. Probabilistic location and routing. In *the 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, 2002.

[67] Jason Rohrer. Mute : Simple, anonymous file sharing. *ZDnet [web]*, 2004.

[68] Roberto Rossi. Ants. *ZDnet [web]*, 2004.

[69] Hosam Rowaihy, William Enck, Patrick McDaniel, and Thomas La Porta. Limiting sybil attacks in structured peer-to-peer networks. *Journal of the ACM*, 2004.

[70] R.Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *International Conference on Peer-to-Peer Computing*, Linko pings Universitet, Sweden, august 2001. IEEE.

[71] R.Schoof and R.Koning. Detecting peer-to-peer botnets. *System and Network Engineering University of Amsterdam*, February 4 2007.

[72] Stuart E. Schechter, Rachel A. Greenstadt, , and Michael D. Smith. Trusted computing, peer-to-peer distribution, and the economics of pirated entertainment. *Harvard University*, 2003.

[73] Bruce Schneier. *Applied Cryptography : Protocols, Algorthms, and Source Code in C.* John Wiley and Sons, 1996.

[74] Srinivas Shakkottai and R. Srikant. Peer to peer networks for defense against internet worms. *arXiv :cs.CR/0605034 v1*, 2006.

[75] Atul Singh, Miguel Castro, Peter Druschel, and Antony Rowstron. Defending against eclipse attacks on overlay networks. *http ://project-iris.net and a gift from Microsoft Research*, 2004.

[76] Atul Singh, Tsuen-Wan, Johnny. Ngan, Peter Druschel, and Dan S. Wallach. Eclipse attacks on overlay networks : Threats and defenses. *Journal of the ACM*, 2006.

[77] S.Marti and H.Garcia-Molina. Taxonomy of trust : Categorizing p2p reputation systems. *Elsevier*, 2006.

[78] S.Staniford-Chen, S.Cheung, and A.al. Grids - a graph based intrusion detection system for large networks. In *Proceedings of the 19th National Information Computer Security Conference (Baltimore, MD)*, October 1996.

[79] William Stallings. *Cryptography and Network Security Principles and Practices*. Prentice Hall, November 2005.

[80] T.Hoffeld, 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.

[81] D. Tsoumakos and N. Roussopoulos. Adaptive probabilistic search in peer-to-peer networks. *the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03),*, 2003.

[82] V. Vlachos, S. Androutsellis-Theotokis, and D. Spinellis. Security applications of peer-to-peer networks. *Computer Networks*, pages 195–205, 2004.

[83] Weichao Wang and Aidong Lu. Visualization assisted detection of sybil attacks in wireless networks. *Journal of the ACM*, 2006.

[84] Bin Xiao, Bo Yu, and Chuanshan Gao. Detection and localization of sybil nodes in vanets. *Journal of the ACM*, 2006.

[85] Z. Xu, R. Min, and Y. Hu. Hieras : a dht based hierarchical p2p routing algorithm. *the 32nd International Conference on Parallel Processing (ICPP'03)*, 2003.

[86] B. Yang and H. Garcia-Molina. Improving search in peer-to-peer networks. In *the 22nd IEEE International Conference on Distributed Computing*, 2002.

[87] C. Yang and J. Wu. A dominating-set-based routing in peer-to-peer networks. In *the 2nd International Workshop on Grid and Cooperative Computing Workshop*, 2003.

[88] Y.Kulbak and D.Bickson. The emule protocol specification. *The Hebrew University of Jerusalem*, 2005.

[89] Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham Flaxman. Sybilguard : Defending against sybil attacks via social networks. *SIGCOMM ACM*, September 2006.

[90] Wei Yu. Analyze the worm-based attack in large scale p2p networks. *Eighth IEEE International Symposium on High Assurance Systems Engineering*, 2004.

[91] Lidong Zhou, Lintao Zhang, Frank McSherry, Nicole Immorlica, Manuel Costa, and Steve Chien. A first look at peer-to-peer worms : Threats and defenses. *IEEE*, 2006.