Etat de l'art : Réseaux pair à pair, supervision, sécurité et approches collaboratives

Thibault Cholez, Isabelle Chrisment, Festor Olivier, Guillaume Doyen, Rida Khatoun, Juliette Dromard

Groupement d'Intéret Scientifique
Surveillance, Sureté et Sécurité des Grands Systèmes
– GIS 3SGS –

Projet ACDA-P2P :
Approche Collaborative pour la Détection d'Attaques
dans les réseaux Pair à Pair

Délivrable 1

Etat de l'art : Réseaux pair à pair, sécurité et approches collaboratives

UMR STMR 6279 :
Guillaume Doyen
Juliette Dromard
Rida Khatoun

LORIA–INRIA Grand Est :
Thibault Cholez
Isabelle Chrisment
Olivier Festor

22 octobre 2010

Résumé

Le présent délivrable présente le résultat du travail mené durant les trois premiers mois du projet GIS 3SGS Acda-P2P dont l'objectif est de proposer une architecture collaborative pour la détection d'attaques dans les réseaux pair à pair.

Il présente un état de l'art qui couvre les différents aspects scientifiques et techniques du projet. Dans un premier temps, les réseaux pair à pair et leurs architectures sont détaillés. Dans un second temps, les méthodes de supervision pour ce type de réseaux sont présentées et comparées. Dans un troisième temps, les failles de sécurité dans les réseaux pair à pair sont mises en évidence. Enfin, dans un quatrième temps, les approches collaboratives, comme solutions pour la sécurité des réseaux et services, sont exposées.

Table des matières

1 Introduction **4**

2 Architectures pair à pair **6**
2.1 Introduction 6
2.2 Réseaux pair à pair avec serveur 6
2.2.1 Serveur central 7
2.2.2 Serveurs distribués 7
2.2.3 Serveurs différenciés 7
2.3 Réseaux pair à pair sans serveur 8
2.3.1 Réseaux P2P non-structurés 8
2.3.2 Réseaux P2P structurés 10
2.4 Comparaison des architectures P2P 12

3 Supervision dans les réseaux pair à pair **14**
3.1 Introduction 14
3.2 Méthodes de supervision 14
3.2.1 Supervision du trafic P2P 14
3.2.2 Supervision d'un serveur 15
3.2.3 Pots de miel 17
3.2.4 Explorateurs 19
3.2.5 Sondes distribuées 20
3.3 Comparaison des méthodes de supervision 21

4 Sécurité dans les réseaux et services pair à pair **24**
4.1 Introduction 24
4.2 Pollution 24
4.2.1 Stratégies de pollution 24
4.2.2 Pollution du mécanisme d'indexation 25
4.2.3 Diffusion de la pollution 26
4.2.4 Solutions possibles 27
4.3 Attaque Sybil 28
4.3.1 Attaque Sybil 28

5 Approches collaboratives pour la sécurité des services **31**
5.1 Les attaques distribuées 31
5.1.1 Les attaques de déni de service distribuées 32
5.1.2 Historique des attaques majeures 33
5.2 Les IDS collaboratifs 34
5.2.1 Classification des IDS 34
5.2.2 Architectures existantes 36
5.3 Conclusion 41

6 Conclusion et travaux à venir **42**

Chapitre 1

Introduction

Le projet Acda-P2P est un projet académique, financé par le GIS[1] 3SGS[2]. Il regroupe l'équipe ERA[3] de l'UMR 6279 STMR[4] et l'équipe Madynes[5] de l'INRIA[6] Grand Est. D'une durée initiale d'un an, le projet a débuté en mai 2010. Il vise à proposer une solution collaborative pour la détection d'attaques dans les réseaux pair à pair (P2P). Plus spécifiquement, il s'inscrit dans le contexte scientifique suivant.

Les réseaux P2P sont devenus, en quelques années, une application majeure de l'Internet en permettant à des millions d'utilisateurs de partager rapidement et sans coût d'infrastructure de grandes quantités de données. Cependant, les réseaux P2P peuvent également être un support pour des activités malveillantes menaçant la sécurité du réseau P2P lui-même (pollution des données, surveillance des échanges, ...) ou, plus généralement, d'Internet (déni de service, propagation de vers, contrôle de botnet, ...). Étant donné le développement croissant de ces réseaux, pouvoir détecter lorsqu'un réseau P2P devient le support d'activités malveillantes devient primordial pour s'en prémunir.

Les réseaux P2P dits structurés, mettant en œuvre une table de hachage distribuée (DHT), présentent le plus grand intérêt d'étude. Ceux-ci sont en effet largement diffusés, les DHT de KAD ou BitTorrent comptent plusieurs millions d'utilisateurs simultanés, et leur architecture peut être facilement détournée à des fins malveillantes par différents moyens (attaque Sybil, index poisoning). Par exemple, d'après [58], des attaquants ont pu compromettre 300 000 nœuds dans un système pair à pair afin de mener une attaque de type DDoS. Ces réseaux étant entièrement dynamiques et distribués, il est cependant très difficile de collecter des informations afin de détecter les attaques, et encore davantage d'agir sur des comportements malveillants au sein du réseau.

Les approches de supervision classiques dans les réseaux P2P étudient les contenus partagés ou le comportement général des utilisateurs, sans s'intéresser aux motifs traduisant une utilisation anormale de la DHT. Une approche passive permet d'observer sur une petite zone de la DHT le trafic P2P sans injecter de données supplémentaires dans le réseau. Les honeypots permettent également de collecter des informations sur l'activité de quelques contenus spécifiques

[1] Groupement d'Intéret Scientifique
[2] Surveillance, Sureté et Sécurité des Grands Systèmes
[3] Environnement de Réseaux Autonomes
[4] Science et Technologie pour la Maîtrise des Risques
[5] Management of Dynamic Networks and Services
[6] Institut National de Recherche en Informatique et Automatique

au sein du réseau P2P, en attirant les pairs malveillants par l'annonce de faux fichiers. Ces approches offrent une vision précise mais cependant trop localisée et fragmentaire du réseau global. Les méthodes de supervision actives sollicitant directement les pairs sont souvent trop intrusives et détectables. Les explorateurs, peuvent découvrir l'ensemble des pairs d'un réseau mais ne peuvent appréhender les communications (et services) échangés entre pairs. Une solution collaborative pourrait permettre de combiner les qualités des outils de détection localisés tout en étant capable de considérer le réseau dans son intégralité.

Dans ce contexte, nous présentons dans ce premier délivrable un état de l'art relatif au contexte scientifique du projet ACDA-P2P qui s'organise de la manière suivante. Le chapitre 2 introduit les réseaux pair à pair et détaille les différentes architectures sur lesquelles ils peuvent reposer. Celles-ci se différencient principalement par l'assistance ou non de serveurs d'indexation et par leur topologie qui peut être structurée ou non structurée. Le chapitre 3 présente ensuite les différentes méthodes de supervision possibles pour les réseaux P2P. Celles-ci sont : la supervision du trafic, la supervision d'un serveur, la mise en place de pots de miel et enfin le déploiement de sondes distribuées. Une comparaison des différentes méthodes est aussi fournie. Le chapitre 4 recense les failles de sécurité inhérentes aux réseaux pair à pair. Il se concentre plus spécifiquement sur la pollution de contenu et les attaques Sybil qui sont parmi les actions malveillantes les plus répandues sur les réseaux P2P publics. Enfin, le chapitre 5 s'extrait du contexte des réseaux P2P et présente un état de l'art sur l'utilisation d'approches collaboratives dans le cadre de la sécurité des réseaux et services. Il se concentre plus particulièrement sur les systèmes de détection d'intrusion et leur utilisation dans le cadre de l'attaque de déni de service distribuée. Pour terminer, le chapitre 6 indique les travaux en cours et à venir sur le projet ACDA-P2P. Ces derniers consistent principalement à surveiller sur une longue période les contenus populaires partagés sur le réseau P2P Kad afin de mieux comprendre le comportement des pairs malveillants.

Chapitre 2

Architectures pair à pair

2.1 Introduction

Nous présentons dans ce chapitre les différentes architectures P2P existantes. Une architecture P2P est un modèle organisationnel définissant les liens de communication entre les pairs ainsi que le processus de localisation des ressources partagées. Nous regroupons ainsi les réseaux P2P en deux grandes familles : ceux ayant recours à des serveurs et ceux complètement distribués. Nous énonçons les principes de chaque architecture au travers d'exemples de réseaux P2P réels, et comparons les forces faiblesses de chaque approche.

2.2 Réseaux pair à pair avec serveur

Les architectures ayant recours à des serveurs sont les plus simples et furent parmi les premières développées. Nous présentons trois architectures différentes et les exemples associés : architecture avec un serveur central (Napster), architecture avec des serveurs distribués (eDonkey), et architecture avec des serveurs différencié (Bittorrent). L'utilisation de serveurs implique une distribution partielle des services dans le réseau P2P puisqu'une autre partie est centralisée. En général, les serveurs ont la charge de connaître des ressources partagées par les pairs à un moment donné. Ils jouent donc le rôle d'annuaire où chaque pair connecté publie les ressources dont il dispose et recherche celles souhaitées. Une fois les pairs potentiels identifiés grâce au serveur, les échanges de services peuvent être réalisés directement entre pairs. Le serveur a donc un rôle fondamental car il est met en relation les pairs, il est préalable à tout établissement de service. Il doit donc être :

- puissant, pour indexer l'ensemble des ressources et parcourir l'index rapidement afin de répondre aux nombreuses requêtes des pairs ;
- disponible, car sans lui les pairs ne peuvent localiser les ressources et aucun service n'est rendu ;
- de confiance, car il peut divulguer les données indexées ou polluer le service avec de fausses données ;

2.2.1 Serveur central

Napster utilisait ainsi un unique serveur (en réalité un cluster de serveurs dédiés) [29] responsable d'indexer les fichiers partagés par les pairs. Après avoir publié sur le serveur les fichiers partagés, un pair pouvait rechercher les pairs partageant un fichier particulier, pour télécharger le fichier en établissant une connexion directe avec un pair. L'arrêt du serveur en 2001 a donc naturellement entraîné l'arrêt complet du réseau. Le point critique pour Napster fut donc la disponibilité du serveur.

2.2.2 Serveurs distribués

Le réseau eDonkey[1] [22] [63] utilise un ensemble de serveurs distribués et opérés par des entités différentes. Chaque serveur individuellement a un fonctionnement proche de Napster, les pairs y publiant les fichiers partagés et y recherchant les fichiers souhaités. La recherche de fichiers, étant donné un mot clé a lieu par défaut sur le serveu auquel est connecté le pair. Un pair peut cependant demander une recherche globale qui sera transmise aux autres serveurs. Une fois un fichier sélectionné, la recherche des sources pour ce fichier est distribuée sur les serveurs. Les sources peuvent également être partagées entre deux pairs intéressées par un même contenu. Le téléchargement a finalement lieu en pair–pair.

En distribuant les serveurs, la disponibilité du réseau eDonkey semble a priori moins critique que celle de Napster. Pourtant, entre 2006 et 2007 les trois serveurs eDonkey les plus populaires, regroupant à eux seuls la très grande majorité des pairs, à savoir « Razorback2 », « DonkeyServer N°1 » et « DonkeyServer N°2 » ont été fermés pour des raisons légales. A son apogée, Razorback2 [2] atteignait ainsi plus d'un million de pairs connectés. En même temps sont apparus de nombreux serveurs malveillants, enregistrant les requêtes des pairs et polluant les recherches de contenus. Sans vraiment résoudre le problème de la disponibilité, la distribution des serveurs introduit donc en plus un réel problème de confiance envers les serveurs inconnus. Les utilisateurs migrent donc vers le réseau P2P complétement distribué KAD, qui est également accessible par le principal client d'eDonkey : eMule.

2.2.3 Serveurs différenciés

Le réseau P2P Bittorrent a la particularité d'avoir une architecture hybride, reposant sur deux types de serveurs distincts préalables aux changes pair pair. Bittorrent repose tout d'abord sur des serveurs web pour indexer les contenus partagés. Les serveurs web permettent d'obtenir un fichier *torrent* correspondant aux mots-clés souhaités. Ce fichier *torrent* comporte plusieurs informations, d'une part des informations intrinsèques au(x) fichier(s) partagé(s) à travers celui-ci et d'autre part l'adresse d'un serveur appel *tracker*. Le *tracker* est un second type de serveur dont le rôle est, pour un fichier *torrent* donné, de connaître et de maintenir jour la liste des pairs le partageant, ces pairs constituent le *swarm*. Chaque pair partageant le contenu du *torrent* annonce ainsi sa présence au *tracker* et un pair souhaitant le télécharger demande au *tracker* la listes des sources potentielles. Finalement, l'échange de données se fait de pair à pair.

[1] galement appel eDonkey 2000 ou ed2k

[2] www.razorback2.com

Tout comme Napster et eDonkey, les principaux serveurs participant au fonctionnement de Bittorrent sont la cible de procédures judiciaires. Les serveurs web indexant les contenus partagés sont ainsi contraints de retirer les *torrents* indexés (Mininova [19]), ou sont bloqués par les FAI dans certains pays (The Pirate Bay[9]), de même que les serveurs hébergeant des trackers. Les clients Bittorrent évoluent donc en remplaçant les différents composants centraux par des systèmes distribués. Une alternative au *tracker* [8] est d'ors et déjà offerte par deux DHT basées sur Kademlia (Vuze DHT[3] et Mainline DHT). Celles-ci permettent de stocker et de retrouver de manière complètement distribuée la liste des pairs partageant un contenu (cette liste peut également être échangée directement entre les pairs via le protocole PEX[4]) ainsi que le fichier *torrent* associé. La DHT permet donc d'une part de remplacer le serveur *tracker*, et d'autre part de décharger les serveurs web du stockage des fichiers *torrent*. Ces informations sont accessibles dans la DHT partir du hash du fichier *torrent*, appelé également *magnet link* [62]. Si les serveurs web n'ont plus la charge de fournir les fichiers torrent, l'indexation des *magnet links* par rapport aux mots-clés des contenus leur est encore dévolue.

2.3 Réseaux pair à pair sans serveur

La seconde famille d'architectures P2P forme des réseaux complètement distribués. Dans ces réseaux, chaque nœud exécute le même code client. L'indexation des ressources est réalisée par les pairs eux mêmes selon deux principes : soit les pairs indexent leurs propres données et les recherches se propagent pour couvrir l'intégralité du réseau, il s'agit de la méthode employée par les réseaux P2P non-structurés, soit l'indexation d'une donnée est attribuée à un groupe de pairs précis par l'utilisation d'une table de hachage distribuée (DHT), il s'agit des réseaux P2P structurés.

2.3.1 Réseaux P2P non-structurés

Les serveurs ne mettant plus en relation les pairs des réseaux non-structurés, ces derniers établissent des liens aléatoires entre eux. Pour rejoindre le réseau, un pair doit connaître l'adresse d'un autre pair déjà connecté et qui sert alors de nœud d'insertion. Par ce nœud, le pair entrant découvre progressivement d'autres nœuds du réseau et établit des liens avec eux, selon un algorithme propre au réseau, pour ensuite y propager les messages. Les requêtes sont ainsi propagées par inondation, chaque pair propageant la requête à tous ces voisins, ce qui implique un grand nombre de messages, ou en suivant un chemin aléatoire, chaque pair sélectionnant un de ses voisins pour propager la requête. La terminologie des réseaux P2P non-structurés est la suivante :

- le **degré** de connectivité de chaque nœud définit le nombre de pairs connus dans le réseau ;
- le **nombre de sauts**[5] d'une requête définit le nombre de pairs contacts avant d'arrêter sa propagation ;
- la **méthode de propagation** définit la sélection des pairs qui propager la requête ;

[3]http ://wiki.vuze.com/w/Distributed_hash_table

[4]Peer exchange

[5]aussi appel TTL pour Time to Live

Nous présentons trois réseaux P2P non-structurés mettant en oeuvre des organisations et des méthodes de propagation différentes, à savoir Gnutella 0.4, FastTrack et GIA.

Gnutella 0.4 Gnutella est le premier réseau P2P complètement distribué, la première version ayant été créee en 2000. Nous présentons ici la version 0.4 [17] de Gnutella car celle-ci a recours à l'inondation pour propager les messages. Dans Gnutella, chaque pair indexe ses propres fichiers. Cette connaissance n'est jamais partagée ou déléguée à d'autres pairs. Pour rechercher un fichier, un pair transmet la requête à tous ses voisins qui propagent à leur tour la requête par inondation jusqu'à atteindre le nombre de saut maximum prévu par le protocole. L'identifiant unique de chaque requête permet aux pairs de ne pas ré-mettre une requête reçue plusieurs fois et de mémoriser l'association « requête-expéditeur » afin de répondre. Chaque pair ayant reçu la requête l'évalue et répond s'il possède la ressource demandée. Les réponses sont ainsi renvoyées de proche en proche vers l'expéditeur en suivant le chemin initial. A chaque niveau, les réponses sont agrégées avant d'être transmises. Finalement, le pair initial reçoit les réponses et peut alors établir une connexion directe vers les pairs pour obtenir le fichier souhait.

La distribution complète du réseau le rend insensible aux problèmes affectant les précédentes architectures reposant sur des serveurs. Cependant, cette architecture a également montré d'importantes limites dues la méthode de propagation employée. L'inondation passe en effet très mal l'échelle, le nombre de messages émis augmentant linéairement avec la taille du réseau. Quand le réseau s'agrandit, une recherche complète nécessite plus de sauts et donc plus de messages, au risque de surcharger les pairs les moins puissants. Si le nombre de sauts maximum n'augmente pas avec la taille du réseau, une partie du réseau n'est alors plus atteignable. Pour pallier cette faiblesse, la version suivante de Gnutella (0.6) [18] délègue la propagation des requêtes de « super-pairs », tout comme l'architecture de FastTrack présentée ci-après.

FastTrack Le réseau Fastrack, créé en 2001, est principalement connu à travers son client Kazaa [36]. Fastrack introduit une différenciation des pairs participant au réseau en distinguant deux groupes, les pairs et les super-pairs. Chaque pair normal est connecté à un super-pair, auquel il délègue l'indexation de ses contenus, les super-pairs étant inter-connectés entre eux. Pour effectuer une recherche, un pair transmet la requête à son super-pair qui la propage alors par inondation aux autres super-pairs. Les super-pairs concentrant la connaissance du réseau, ils doivent seuls être contactés lors d'une recherche et constituent ainsi la colonne vertébrale du réseau par laquelle se propagent les requêtes. Cette organisation réduit fortement le nombre de pairs contacts lors d'une recherche ce qui permet à l'architecture de mieux passer l'échelle. Le problème de cette approche est que l'ensemble de la charge du réseau est supportée par les super-pairs, tant en termes de consommation mémoire et processeur (indexation des contenus, traitement des requêtes) qu'en termes de bande passante. Le choix des super-pairs est donc crucial car ceux-ci doivent être suffisamment puissants et disponibles.

GIA L'architecture proposée par GIA en 2003 [10] améliore encore le passage à l'échelle des réseaux non-structurés. La principale différence réside dans la méthode de propagation qui utilise un chemin aléatoire au lieu de l'inondation. Afin d'améliorer les connaissances parcourues lors de la propagation d'une requête, chaque pair de GIA réplique son index sur ses voisins. Contrairement à FastTrack qui ne propose que deux catégories de pairs, GIA propose un algorithme de sélection des voisins qui adapte précisément la connectivité d'un pair à ses capacités

en définissant pour chaque pair un nombre maximal de voisins. La propagation de la requête suit un chemin aléatoire en favorisant les pairs les plus connectés dont la capacité de traitement des requêtes n'est pas saturée. La propagation est limitée par un nombre de sauts ainsi qu'un nombre de réponses positives au delà duquel la réponse est retournée.

L'architecture de GIA requiert moins de messages lors d'une recherche ce qui la rend compatible avec de très grands réseaux. Elle fournit en contre-partie des résultats moins complets, ce qui est problématique lorsqu'un pair recherche une ressource rare, et une recherche peut complètement échouer si un des pairs sur le chemin de propagation disparait. Ces inconvénients n'affectent pas les réseaux P2P structurés.

2.3.2 Réseaux P2P structurés

Contrairement aux précédentes architectures P2P présentées (à l'exception de GIA), les réseaux P2P structurés sont issus de la recherche académique. Ils ont la particularité d'organiser le réseau P2P en une topologie routable, c'est-à-dire que chaque pair dispose d'un identifiant permettant de le localiser en suivant un chemin déterministe parmi les pairs, et nécessitant un minimum de messages. Chaque ressource partagée au sein du réseau possède également un identifiant, qui est le résultat d'une fonction de hachage, et permettant de la localiser rapidement. Ainsi, les réseaux P2P structurés peuvent être considérés comme une table de hachage distribuée (en anglais « Distributed Hash Table » ou DHT), chaque pair étant responsable des entrées de la table égales ou proches de son identifiant. Pour qu'une donnée soit stockée de manière pérenne dans le réseau malgré des pairs non-fiables, plusieurs pairs satisfaisant aux contraintes de proximité peuvent la stocker, on parle alors de **réplicats**. Les DHT fournissent donc deux primitives essentielles à la réalisation de services : d'une part l'insertion d'une donnée dans le réseau à l'emplacement de son identifiant, et d'autre part la récupération d'une donnée dans le réseau étant donné son identifiant. La principale limite de cette approche est que la donnée doit être parfaitement caractérisée par son identifiant avant d'être retrouvée, les DHT autorisent uniquement des recherches exactes sur les données.

Les DHT permettent une localisation des pairs et des données très performante. Le routage de proche en proche nécessite ainsi au maximum $O(log(N))$ messages[6], N tant le nombre de pairs dans le réseau. Les réseaux P2P structurés passent donc mieux l'échelle que les réseau non-structurés dont le nombre de messages augmente linéairement $O(N)$ avec la taille. De plus, la charge est distribuée de manière totalement homogène sur les pairs et reste faible, tant en termes de messages (efficacité du routage) qu'en termes de traitement, car seuls les pairs terminaux évaluent la requête, les autres la routant simplement.

Nous décrivons dans cette sous partie l'architecture générale de deux réseaux P2P structurés, Chord et Kademlia, afin d'en exhiber les caractéristiques. Chord est une des premières DHT connue, son schéma a ensuite été largement repris et amélioré, notamment par Kademlia qui a ensuite été implanté dans de nombreuses applications réelles. D'autres réseaux P2P structurés existent. Ils partagent les même propriétés générales mais peuvent différer quant à l'espace d'adressage (CAN[49]) et à l'algorithme de routage (Pastry[51], Tapestry) employés.

[6]avec des pairs stables, $O(log^2(N))$ est plus réaliste en considérant l'instabilité des nœuds

Chord Chord [61] organise son espace d'adressage suivant un anneau dont les 2^m adresses (ou identifiants, id) possibles sont ordonnées le long de sa circonférence. Chaque pair ainsi que chaque ressource possède un identifiant obtenu par une fonction de hachage SHA-1 (dans ce cas $m = 160$ bit) qui garantit une distribution homogène des ressources sur l'anneau Chord. Chord se limite une fonction de routage, c'est dire qu'étant donné un identifiant, Chord localise le pair responsable qui est celui possédant le plus petit identifiant supérieur ou égal à celui de la ressource. Quand un pair rejoint le réseau, il prend en charge une partie des identifiants attribués à son successeur direct, et,quand il quitte le réseau, tous ses identifiants sont attribués à son successeur. Chord ne propose donc pas toutes les primitives de stockage / recherche / réplication pour opérer une DHT, et encore moins une application de partage de fichier fonctionnelle.

L'intérêt de Chord est de proposer une structure simple et efficace permettant de router les messages. Pour cela, chaque pair du réseau maintient une table de pointeurs contenant les informations $< id, ip, port >$ pour un certain nombre de pairs. Plus cette liste est grande, plus elle est coûteuse maintenir, mais plus le routage est rapide (le cas extrême étant de connaître tous les pairs du réseau, le routage ne nécessite alors qu'un message). Chord parvient trouver un compromis entre la taille de la table de routage comportant $O(log(N))$ entrées et le routage nécessitant $O(log(N))$ messages. Pour cela, un pair, d'id *currentID*, choisit chaque pointeur de sa table de telle sorte que le pair sélectionne soit le seul représentant de l'intervalle $|(currentID + 2^i) - (currentID + 2^{i+1})|$ avec $i + 1 < m$. Un pair connait donc un successeur représentant une portion de l'anneau qui est deux fois plus grande pour chaque nouvelle entrée. Le routage est ensuite itératif, de proche en proche, chaque pair envoyant la requête au successeur connu le plus proche de l'identifiant recherché.

Les principales limites de Chord sont, d'une part sa topologie en anneau obligeant parcourir l'ensemble de l'espace d'adressage dans le pire cas, et d'autre part, l'absence de primitives pour opérer une DHT. Ces faiblesses sont résolues par l'architecture Kademlia.

Kademlia Dans Kademlia [44], les pairs et les ressources possèdent un identifiant de 160 bits définissant leur place dans l'espace logique du système. Kademlia propose une métrique base sur l'opération XOR pour évaluer les distances sur l'espace d'adressage, là où Chord utilise seulement la notion de « plus proche successeur ». Le premier avantage de cette métrique permet Kademlia, lors de la réception de n'importe quelle requête, de pouvoir retenir le pair émetteur pour peupler sa table de routage, minimisant ainsi le nombre de messages nécessaires à son maintien. Le second avantage est de permettre un routage moins rigide que Chord, l'opération XOR étant symétrique (A XOR B = B XOR A). Chord retient dans sa table de routage le pair précédent exactement un certain intervalle, alors que Kademlia peut contacter n'importe quel nœud à l'intérieur d'un intervalle respectant une certaine distance XOR, les zones ainsi créées permettent de s'affranchir d'une relation d'ordre contraignante entre chaque pair.

Kademlia fournit toutes les primitives nécessaires au fonctionnement d'une DHT, à savoir (STORE, FIND_NODE et FIND_VALUE). Une donnée est ainsi stockée (STORE) sur les k pairs trouvés comme étant les plus proches de son identifiant selon la métrique XOR. Ces mêmes pairs peuvent également retourner la donnée stockée lors d'une recherche sur son identifiant (FIND_VALUE). La fonction de routage (FIND_NODE) permet de trouver les k nœuds les plus proches d'une valeur. Pour cela, chaque pair dispose d'une table de routage formant un arbre binaire dont chaque niveau i contient au maximum k contacts (appel $k - bucket$, par exemple, $k = 10$) et représente une zone de l'espace d'adressage située à une distance XOR du

pair courant comprise entre 2^i et 2^{i+1}, avec $0 < i \leq 160$[7]. Le premier niveau de l'arbre permet donc de contacter des pairs situés dans la moitié oppose de l'espace d'adressage (i.e, dont le premier bit diffère) par rapport au pair courant, alors que les niveaux les plus bas représentent le voisinage immédiat du pair.

Pour localiser les k pairs les plus proches d'un identifiant « searchid », un pair réalise des recherches itératives en parallèle. Il envoie ainsi des demandes de contacts concernant *searchid* aux α (par exemple, $\alpha = 3$) contacts les plus proches de sa table de routage. Ceux-ci ne propagent pas directement la requête mais répondent avec les α contacts les plus proches qu'ils connaissent. A l'issue de cette première étape, les α plus proches contacts reçus sont interrogés et ainsi de suite. Quand les pairs retournés ne se rapprochent plus de l'identifiant, la requête de service peut alors être envoyée car les k contacts les plus proches sont atteints. Conduire la recherche de manière itérative et parallèle génère plus de messages mais rend le processus de routage beaucoup plus robuste au départ des nœuds ou à la présence d'attaquants.

Le système P2P proposé par Kademlia a prouvé son efficacité en étant à l'origine de plusieurs réseaux P2P réels largement déployés (comportant plusieurs millions de pairs simultanés) savoir Overnet, KAD et Bittorrent Mainline DHT, qui sont respectivement implantés par les clients eDonkey, eMule et Bittorrent. Bien qu'implantant tous des algorithmes basés sur Kademlia, ces trois réseaux P2P ne sont pas pour autant compatibles, chacun suivant son propre protocole, par ailleurs peu ou non-documentés. Le succès de Kademlia est dû à sa simplicité de mise en œuvre (comparableà Chord) ainsi qu'à sa grande efficacité de routage en $O(log(N))$. Sa structure permet de maintenir naturellement à jour la table de routage, un pair recevant des messages de sources situées à différentes distances de lui dans les mêmes proportions que celles requises par sa table de routage. Kademlia utilise un unique algorithme de routage (contrairement Pastry et Tapestry) et la réplication nécessaire au bon fonctionnement du système pair pair est intrinsèquement présente dans Kademlia.

2.4 Comparaison des architectures P2P

A l'issue de cette présentation des principales architectures P2P, la première conclusion porte sur l'utilisation de serveurs dans une architecture P2P. Ceux-ci posent des problèmes de confiance, de financement et créent des points de défaillance unique rendant le réseau largement vulnérable. Malgré un aspect pratique indéniable, tout composant central doit donc être évité pour respecter le paradigme pair à pair.

Si les premiers réseaux non-structurés montraient des limites de croissance évidentes, les nouvelles architectures sont moins impactées par ce problème. Les solutions proposées ont cependant des désavantages : l'utilisation des super-pairs concentre la charge sur une partie du réseau qui peut être amenée à saturer et l'utilisation d'un chemin aléatoire limite quant à lui l'étendue d'une recherche dans le réseau. Les réseaux non structurés sont en outre vulnérables à des attaques visant les pairs les plus connectés. Si la distribution des liens suit une loi de puissance, [29] a montré qu'attaquer les 4% de pairs les plus connectés partitionne entièrement le réseau.

Les réseaux P2P structurés permettent une répartition homogène de la charge ainsi qu'une localisation des ressources rares. Ils permettent de stocker n'importe quelle donne sur la DHT

[7] les niveaux les plus bas, ne sont généralement pas peuplés par manque de pairs assez proches

et fournissent des mécanismes de routage très efficaces pour les localiser en requérant un faible nombre de messages. La structure peut également être maintenue moindre coût. Les réseaux P2P structurés ont donc prouvé conceptuellement et en pratique leur capacité à passer l'échelle et à supporter des services P2P.

Les avantages et inconvénients des différentes architectures P2P sont synthétisés dans le tableau 2.1.

Architecture P2P	Avantages	Inconvénients
Avec Serveur	– Recherches non-exactes – Charge minimale des pairs – Contrôle des pairs/données possibles	– Pérennité des serveurs – Coût des serveurs – Confiance envers les serveurs inconnus
Réseaux non-structurés	– Distribution complète – Recherches non-exactes	– Passage l'échelle limité (inondation) – Charge non-homogène des pairs (super-pairs) – Localisation de données rares (chemin aléatoire)
Réseaux structurés	– Distribution complète – Passage l'échelle / Charge homogène – Localisation de données rares – Stockage des données	– Recherches exactes uniquement

Tab. 2.1 – Comparaison des architectures P2P

Chapitre 3

Supervision dans les réseaux pair à pair

3.1 Introduction

Les réseaux P2P ont fait l'objet de nombreuses études académiques visant à en comprendre leurs caractéristiques, leur fonctionnement et leurs limites. Pour ce faire, des campagnes de mesures ont été réalisées pour collecter des données issues des quelques réseaux P2P largement déployés. Les données collectées peuvent viser la connaissance de plusieurs objets des réseaux P2P tels que :

- le trafic P2P (e.g. nombre de connexions, dynamique, localisation) ;
- le protocole (e.g. types des messages, proportion des messages, overhead) ;
- la topologie du réseau (e.g. nombre de voisins, tables de routage) ;
- les pairs (e.g. nombre, localisation, comportement) ;
- les contenus (e.g. nombre de fichiers, pollution) ;
- un contenu précis (e.g. nombre de sources, recherches, téléchargements).

Selon les architectures P2P, certaines données sont plus ou moins délicates à obtenir. Les architectures P2P mises en œuvres étant très différentes, de même que les données mesurées, chaque étude a conduit à la réalisation d'une architecture de supervision spécifique. Nous présentons dans ce chapitre les différentes méthodes de supervision existantes, en les regroupant par famille, et en décrivant leur spécificité.

3.2 Méthodes de supervision

3.2.1 Supervision du trafic P2P

Les réseaux P2P reposant sur le réseau IP, une première méthode de supervision consiste à appliquer la supervision du trafic Internet au trafic généré par les applications P2P. Cette approche peut être réalisée pour n'importe quel réseau P2P, indépendamment de son architecture. Elle a également l'avantage d'être passive et de pouvoir aussi bien capturer le trafic de signalisation que celui dû aux échanges de fichiers. Le trafic du réseau P2P étudié est collecté sur un lien significatif du réseau Internet, par exemple le lien principal d'un campus universitaire ou celui d'un fournisseur d'accès à Internet (FAI). Ensuite, les traces collectées sont analysées. Deux approches sont possibles. La première se contente d'analyser les informations

issues des couches réseau et transport du trafic (c'est à dire les entêtes IP et TCP/UDP). La seconde approche analyse également les données applicatives, ce qui nécessite la connaissance du protocole P2P étudié.

Analyse des informations réseau et transport

Dans [54], les auteurs ont analysé le trafic P2P passant par les routeurs de bordure d'un FAI, en s'intéressant aux flux de trois réseaux P2P à savoir Gnutella, FastTrack et DirectConnect. Les informations ainsi collectées permettent d'apprécier la dynamique générale des réseaux P2P (comment sont distribuées les connexions dans le temps, géographiquement, ou entre les noeuds) et leur impact sur le réseau IP. La principale limite de cette étude est de reposer uniquement sur les ports par défaut des applications P2P pour identifier leur trafic. Or, les ports peuvent être facilement changés par les utilisateurs et certains clients utilisent par défaut des ports aléatoires. Cette limite s'applique à d'autres études de trafic basées sur les ports [52][63] .

L'article [25] propose une méthode automatique permettant d'identifier le trafic P2P sans connaissance a priori des ports de communication utilisés, en se basant sur une reconnaissances de motifs propres au trafic P2P au niveau transport. L'algorithme PTP (P2P Traffic Profiling) est ainsi capable de détecter le trafic P2P très efficacement : il reconnait 95% des flux considérés comme étant pair à pair. En revanche, cette approche ne permet pas de distinguer les différents réseaux P2P.

Analyse des informations applicatives

La distinction entre les différents réseaux P2P peut être faite en analysant la partie applicative des données collectées. Cette analyse permet aux auteurs de [63] de séparer les flux servant à la signalisation de ceux servant au transfert de fichier dans le cadre du réseau eDonkey, ce qui fournit une connaissance supplémentaire. L'étude [35] réalisée sur Kazaa analyse encore davantage les données applicatives et peut ainsi compter le nombre de téléchargements observés sur le lien pour un fichier donné, le nombre de fichiers distincts transférés, ou encore tracer l'activité des utilisateurs par rapport au nom renseigné dans le client P2P. Une meilleure identification du trafic P2P est donc possible par l'analyse en temps réel des données applicatives mais cette méthode pose des problèmes légaux (inspection des paquets) et techniques importants.

En résumé, la seule collecte des flux permet davantage de comprendre les interactions entre le réseau IP et les réseaux P2P que le fonctionnement de ces derniers. L'analyse de données applicatives permet de dépasser cette limite au prix d'un effort de stockage et de traitement important. Cependant, cette approche de supervision ne peut appréhender l'ensemble du réseau P2P.

3.2.2 Supervision d'un serveur

Superviser les réseaux P2P par la collecte de trafic est une méthode difficile à mettre en œuvre et à exploiter. Pour obtenir davantage de données applicatives tout en limitant le nombre de points de collecte et la quantité de données collectées, les serveurs sont des points privilégiés pour superviser un système P2P car ils concentrent les informations du système. Deux approches sont possibles pour superviser les informations d'un serveur : d'une part en opérant et en

instrumentant un serveur dont on souhaite acquérir les données, d'autre part en interrogeant un serveur en émettant les même requêtes qu'un client normal.

Instrumentation d'un serveur

Les données transitant par un serveur peuvent être obtenues soit en accédant directement aux données traitées par le serveur, en créant si besoin, puis en lisant les journaux d'exécution, soit en capturant les paquets sur l'interface réseau du serveur à l'aide d'un outil (par exemple TCPDUMP[1]), pour les traiter ultérieurement. L'instrumentation du serveur ne nécessite l'envoi d'aucun message spécifique.

La solution de supervision la plus directe consiste à obtenir un journal résumant les informations transitant par le serveur. Dans le cadre du réseau Bittorrent [24], le journal du *tracker* permet de connaître l'activité d'un *torrent* (les pairs le partageant à un moment donné), mais une approche complémentaire est nécessaire pour connaître les relations entre les pairs. Pour le réseau eDonkey [31] [20], la même limite est rencontrée car le journal fournit par défaut par le principal programme serveur (lugdunum) n'enregistre qu'une partie des informations à savoir la liste des clients connectés, les recherches textuelles et les recherches de sources émises par ces derniers, mais n'enregistre pas les fichiers publiés. Les données des journaux étant partielles, superviser directement le trafic du serveur permet d'obtenir davantage d'informations.

L'article [2] présente ainsi les résultats de 10 semaines de collecte de trafic du principal serveur eDonkey, avec pour objectif l'analyse du comportement des utilisateurs. L'anonymisation nécessaire à la collecte de trafic est difficile car les données pouvant identifier les utilisateurs (adresse IP) apparaissent à la fois au niveau IP et applicatif, ce qui a nécessité l'élaboration d'un logiciel spécifique de capture pour anonymiser le trafic eDonkey en temps réel avant de l'enregistrer. Une autre difficulté de la capture de trafic vient des paquets perdus lors de la capture et rendant la reconstitution des flux TCP très difficile. Cette méthode de supervision montre donc un réel intérêt par l'ampleur des données collectées (9 milliards de messages de 90 millions d'utilisateurs partageant plus de 275 millions de fichiers différents), ainsi que par leur précision (toutes les publications et recherches émises pour n'importe quel fichier) mais comporte plusieurs désavantages. Tout d'abord, il est très difficile de pouvoir instrumenter de la sorte un serveur populaire (collaboration avec les administrateurs), et, même dans ce cas, la vue du réseau reste partielle car l'activité des autres serveurs et les connexions entre pairs demeurent inconnues. Par ailleurs, la collecte de trafic au niveau IP induit un coût de traitement beaucoup plus important que des données collectées au niveau applicatif, les différents champs du protocole devant être analysés par un outil extérieur.

Requêtes sur serveur

La seconde méthode permettant de superviser un serveur consiste à interroger régulièrement le serveur avec les primitives utilisées par les clients normaux. Contrairement à l'instrumentation d'un serveur cette approche nécessite l'émission de nombreux messages pouvant être détectés ou perturber le réseau. En outre, dans le cadre d'eDonkey, les requêtes doivent spécifier le mot-clé ou le fichier recherché ce qui limite l'étendue des résultats pouvant être obtenus de cette manière. En revanche, pour le réseau Bittorrent, l'activité d'un *torrent* peut être facilement

[1] `http://www.tcpdump.org`

supervisé de la sorte, en émettant régulièrement des requêtes au serveur *tracker* afin d'obtenir la liste des pairs constituant le swarm. La supervision d'un *torrent* étant insuffisante pour être significative, cette supervision est appliquée à plusieurs *torrents* en parallèle et complétée par des mesures directes entre les pairs.

Le tableau 3.1 résume les différentes manières de superviser un réseau P2P avec serveur.

Méthode de supervision	Avantages	Inconvénients
Trafic du serveur	– Connaissance des pairs – Connaissance des contenus – Supervision passive	– Difficulté du traitement des données – Vue partielle du réseau – Difficulté d'instrumenter un serveur significatif
Journaux du serveur	– Traitement facile des données – Supervision passive	– Informations enregistrées limitées – Vue partielle du réseau – Difficulté de mettre en œuvre un serveur significatif
Requêtes sur serveurs	– Plusieurs serveurs contactables – Facilité de mise en œuvre	– Requêtes spécifiques à un contenu – Supervision active

Tab. 3.1 – Comparaison des méthodes de supervision pour les réseaux P2P avec serveur

De telles méthodes de supervision sont par définition inapplicables aux réseaux P2P complètement distribués pour lesquels d'autres approches doivent être considérées. Ces autres méthodes de supervision comme les pots de miel, les explorateurs et enfin les sondes distribuées ne reposent que sur des communications entre clients.

3.2.3 Pots de miel

Les pots de miel (en anglais « honeypots ») viennent du domaine de la sécurité informatique et sont utilisés pour attirer les utilisateur essayant de pénétrer illégalement un système. Spitzner définit ainsi les pots de miel comme étant « une ressource d'un système d'information dont la valeur réside dans son utilisation interdite ou illégale » [2]

Les pots de miel sont majoritairement utilisés pour émuler des failles de sécurité de programmes connectés à Internet afin d'attirer du code malveillant. Dans le cadre des réseaux P2P, les pots de miel consistent à étudier l'activité de contenus spécifiques sur le réseau en

[2] « *A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource* » `http://www.tracking-hackers.com/papers/honeypots.html`

annonçant une instance de chacun d'eux. Les différents fichiers servant d'appâts sont partagés par des clients-honeypots et permettent d'attirer les requêtes des autres pairs, telles que les requêtes de téléchargement, qui peuvent ainsi être supervisées lors de la réception par le pot de miel. Cette méthode a l'avantage d'être applicable à toutes les architectures P2P.

Les articles [5] et [33] proposent des architectures permettant de traquer les pairs recherchant des contenus illégaux diffusés dans les réseaux P2P. L'architecture proposée par [33] déploie de faux serveurs analysant les fichiers partagés par leurs clients et s'apparente davantage à une mesure sur serveur avec un accent mis sur la détection de contenus malveillants qu'une réelle architecture de honeypot. Elle a cependant l'avantage de découvrir les fournisseurs des fichiers. L'architecture de [5] utilise des honeypots distribués et offre une interface de gestion permettant de paramétrer le déploiement des honeypots ainsi que la génération des faux fichiers à annoncer d'après un ensemble de mots-clé malveillants. Une analyse statistique des données collectées pour l'ensemble des mots-clé permet de détecter les pairs suspects ayant émis plusieurs recherches. Les auteurs rappellent que de tels honeypots appréhendent uniquement les pairs cherchant les contenus mais non les fournisseurs dont l'activité est pourtant plus critique. La principale limite de cette architecture est d'utiliser de faux fichiers comme apâts qui n'apparaissent pas comme étant populaires aux autre pairs. De plus, la génération automatique des noms limite encore davatage l'attractivité des apâts.

Dans [3], les auteurs mettent en œuvre une réelle architecture de honeypots distribués sur le réseau eDonkey. Chaque honeypot exécute un client connecté à un serveur ed2k différent, partage des fichiers réels suivant une politique donnée, et enregistre les requêtes reçues correspondantes. L'étude compare ainsi les mesures obtenues par différentes politiques faisant varier le nombre de honeypots, leur comportement (silence ou envoi de donnée aléatoires) et le nombre de fichiers annoncés. Pour obtenir la meilleure supervision possible, les résultats montrent clairement l'intérêt de déployer plusieurs honeypots annonçant un même ensemble important de fichiers, sur une longue période de temps et en répondant aux requêtes de téléchargement (même avec des données aléatoires).

Contrairement aux précédentes architectures de supervision, les honeypots sont faciles à mettre en œuvre car ils ne nécessitent pas de collaboration particulière (FAI, administrateur de serveur). Ils permettent d'étudier facilement l'activité de fichiers[3] ou de mots clés[5] précis au sein du réseau. L'étude des mots-clé est cependant très partielle, une étude complète nécessiterait d'annoncer l'ensemble des fichiers existants s'y rapportant. Les honeypots comportent d'autres limites. Tout d'abord, ils sont sujets aux faux-positifs du fait de la pollution. Les fichiers annoncés par le honeypot peuvent en effet être partagés à travers de faux noms par d'autres pairs, une partie des demandes de téléchargement ne reflète donc pas l'intérêt réel des utilisateurs pour le contenu annoncé. Ensuite, c'est une méthode de supervision active pouvant perturber le réseau. L'attractivité du honeypot dépend directement de la popularité des contenus annoncés, or, les fichiers populaires sont majoritairement soumis au droit d'auteur et leur simple annonce peut être détectée par d'autres entités supervisant le réseau. Enfin, partager un fichier ne permet pas de superviser toute l'activité des pairs pour celui-ci, on peut ainsi connaître quel pair souhaite obtenir le fichier à un moment donné mais non qui le partage déjà, la vue obtenue est donc doublement partielle (certains fichiers, certaines requêtes). Les données inaccessibles aux honeypots peuvent cependant être obtenues par d'autres méthodes de supervision, notamment les explorateurs.

3.2.4 Explorateurs

Les explorateurs (en anglais « crawlers ») sont des outils visant à découvrir l'ensemble des pairs constituant un réseau P2P par l'interrogation individuelle de chacun d'entre eux. Pour connaître la topologie du réseau, le crawler doit communiquer avec l'ensemble des pairs ce qui implique : d'une part, de connaître certaines primitives du protocole P2P étudié, et d'autre part, de définir une méthode automatique d'interrogation des pairs. Les méthodes de parcours diffèrent selon les réseaux P2P, ainsi que les données obtenables en interrogeant les pairs. Nous présentons dans cette section trois types d'explorateurs, chacun conçu pour une architecture P2P différente.

Les explorateurs du réseau Gnutella [29] et [50] furent parmi les premières mesures à large échelle réalisées sur les réseaux P2P. A partir d'une liste initiale de pairs, les explorateurs contactent chacun d'entre eux par une requête d'annonce (PING) et découvrent en retour de nouveaux noeuds (PONG) qui seront à leur tour contactés séquentiellement. Cette méthode est cependant très lente (50 heures pour découvrir 4000 pairs), car l'explorateur doit éventuellement attendre un certain temps la réponse de pairs déconnectés. Pour pouvoir explorer rapidement de grands réseaux, [50] distribue l'exploration sur plusieurs clients (50), synchronisés par un serveur alors que [29] émet des annonces avec un TTL important ce qui surcharge davantage le réseau. Pour chaque pair, l'explorateur identifie son adresse IP, la liste de ses voisins ainsi que le nombre et la taille des fichiers partagés. [29] mesure également la latence et la durée de connexion des pairs. Les informations récoltées globalement permettent de connaître la taille du réseau P2P, sa topologie, sa dynamique, et la propension des pairs à partager des contenus. La principale limite de ces explorateurs est de ne pouvoir appréhender précisément les contenus diffusés au sein du réseau ainsi que les requêtes associées.

Les stratégies d'exploration mises en œuvre sur le réseau eDonkey [32] [21] [68] sont différentes. L'explorateur utilisé dans [32] et [21] consiste en deux étape. La première étape découvre un maximum de pairs en interrogeant l'ensemble des serveurs via des requêtes autorisant à rechercher les clients par leur nom. Les serveurs sont ainsi interrogés avec tous les noms de clients possibles (en générant toutes les chaînes de trois caractères (ie « aaa » ... « zzz ») pouvant entrer dans la composition d'un nom). La seconde étape consiste à contacter directement chaque pair trouvé en demandant la liste des fichiers partagés. Cette exploration permet de dresser la carte des fichiers partagés par les pairs du réseau eDonkey. L'analyse de ces données permet de mettre en évidence des groupements d'intérêts entre les pairs[32] et, plus généralement, de comprendre le comportement des pairs par rapport à l'application de partage de fichiers [21]. Cette méthode d'exploration a cependant plusieurs faiblesses. Tout d'abord les serveurs actuels ne supportent plus les recherches de clients par nom. Cette modification du protocole a amené les auteurs de [68] à obtenir la liste de pairs par un autre moyen, en émettant des recherches de sources pour certains fichiers, eux même obtenus à partir de mots clés (cette approche a également été utilisée pour explorer le réseau Napster [29]). La liste des pairs ainsi obtenue est malheureusement contrainte par la langue des mots-clé initiaux (dans l'étude : l'anglais et le chinois). Ensuite, les pairs trouvés ne sont pas tous contactables directement en raison des pare-feu. Cependant, la principale limitation de cette approche tient au fait que la majorité des pairs ne partage aucun fichier ou n'autorise pas la consultation directe des fichiers partagés, cette fonction étant désactivée par défaut.

Les tables de hachage distribuées peuvent également être parcourues. Les articles [59] et [57]

explorent le réseau KAD à l'aide d'un outil nommé Blizzard. Celui-ci connait quelques pairs à priori et en découvre de nouveaux au fur et à mesure du parcours Pour cela, il émet pour chacun des pairs connus 16 requêtes de routage (FIND_NODE) dont l'adresse cible donnée en paramètre est choisie de telle sorte que les pairs retournés proviennent de parties différentes (k-bucket) de la table de routage du pair interrogé. L'émission et la réception des requêtes est réalisée par deux processus légers asynchrones partageant la liste des pairs. Blizzard est ainsi capable de découvrir le réseau KAD en 8 minutes, ou une zone ($1/256^{eme}$) en quelques secondes. Les informations collectées sur chaque pair (KADID, adresse IP, port) lors d'explorations fréquentes permettent d'obtenir une vue globale du réseau (la caractérisation des sessions des pairs, leur répartition sur la DHT, leur pays d'origine, etc.). L'article [70] utilise une stratégie différente d'exploration pour KAD basée sur des requêtes d'amorçage (Bootstrap). Celles-ci retournent davantage de contacts par réponse mais ceux-ci sont aléatoirement répartis sur la DHT. L'exploration émet moins de messages (1 par pair au lieu de 16) mais prend plus de temps en contre-partie (30 min) et découvre moins de pairs. En résumé, dans le cadre des DHT, les explorateurs sont donc capables de découvrir sa topologie mais sont en revanche incapables de connaître les données indexées en son sein.

Les explorateurs sont un outil de supervision pouvant être adaptés à toutes les architectures P2P. Ils permettent de récolter facilement des informations sur la connectivité des pairs (adresse IP, pairs voisins) et plus généralement sur la topologie et la dynamique du réseau P2P. Cette méthode a cependant deux limitation majeures. D'une part, les explorateurs ne permettent pas d'appréhender (sauf cas exceptionnel) les contenus partagés par les pairs ni les messages échangés entre ces derniers (recherches, publications). D'autre part c'est une mesure intrusive qui peut perturber le réseau car de nombreux messages doivent être émis pour le découvrir. Finalement, la dernière méthode de supervision étudiée ici, l'utilisation de sondes, ne présente pas ces faiblesses.

3.2.5 Sondes distribuées

Cette méthode de supervision consiste à injecter dans le réseau un certain nombre de pairs instrumentés, appelés sondes, dont les échanges de messages sont capturés puis analysés. Cette méthode est peu intrusive car aucun trafic propre à l'activité de supervision n'est généré. Nous présentons ici l'utilisation de sondes pour trois réseaux P2P à savoir Gnutella, Bittorrent et KAD.

Les sondes ont d'abord été utilisée dans le réseau Gnutella dont l'architecture se prête particulièrement bien à ce type de supervision. En effet, la propagation des requêtes par inondation, bien que peu efficace, autorise un pair à intercepter une grande partie des messages diffusés sur le réseau. [4] et [43] capturent ainsi les messages (requêtes et réponses) passant par leur(s) sonde(s) pendant une journée. Les informations collectées permettent de connaître le nombre et la proportion de chaque type de requête (ping, querry...), le nombre de pairs répondant sur le chemin d'une requête, leur proximité par rapport à la sonde, mais aussi le nombre de fichiers répondus. Parmi les informations récoltées par les sondes, [56] et [26] étudient en particulier les chaînes de caractères recherchées dans le réseau ce qui permet d'évaluer la popularité des contenus. [26] note également que les fichiers trouvés en réponse pourraient être analysés. Les auteurs de [1] réalisent des mesures similaires mais sur une longue période de temps (3 ans), permettant de constater l'évolution du réseau (topologie, trafic...). Dans le cadre du réseau Gnutella, les

sondes distribuées permettent donc de récolter facilement de nombreuses informations, aussi bien sur la topologie du réseau que sur les contenus partagés et recherchés.

Dans le cadre du réseau bittorrent, deux types de sondes sont utilisées : les unes participent au swarm, les autres à la DHT. Ainsi, [6] essaye d'identifier les pairs participants au partage d'un fichier en injectant des sondes dans le swarn associé. Les sondes communiquent alors avec les autres pairs pour s'assurer qu'ils partagent bien le fichier supervisé. Concernant les DHTs, les auteurs de [13] ont injecté plusieurs sondes dans chacune des deux DHTs utilisées par Bittorrent afin de mesurer et comparer leurs performances. Les articles [16] et [67] étudient quant à eux exclusivement VuzeDHT. L'injection de sondes dans la DHT permet de mesurer les paramètres de fonctionnement tels que le nombre de pairs contactés pour une recherche, la proportion de recherches échouées ou de pairs déconnectés... Au delà des performances, il est également possible de connaître les pairs publiant une entrée donnée (càd le hash d'un torrent), cependant, les informations du torrent associé restent inaccessibles. Afin de faire le lien entre les torrents et les requêtes de publication/recherche capturées sur la DHT, [67] utilise en premier lieu un crawler permettant de récupérer de nombreux torrents afin d'en calculer l'identifiant. La seule injection de sondes ne suffit donc pas pour connaître les données indexées sur la DHT. Le nombre de sondes injectées diffèrent selon les études. Si l'augmentation du nombre de sonde a peu d'intérêt pour évaluer les performances générales, il est en revanche important si l'on souhaite étudier les données indexées. En effet, plus celui-ci est important, plus le nombre de recherches/publications capturées par les sondes sera élevé.

La supervision par l'injection de sondes a été réalisée dans KAD à une plus grande échelle. Les auteurs de [46] [45] ont créé une architecture capable d'insérer dans la DHT autant de sondes qu'il y a de pairs. Afin de ne pas perturber le réseau par l'injection de nombreux pairs, chaque sonde a une visibilité minimale réduite à son voisin immédiat. Les sondes sont capables de capturer les requête de routage (lookup) dont la destination est proche du pair supervisé. 32000 pairs peuvent ainsi être insérés dans une zone de la DHT permettant de capturer une copie de toutes les requêtes de routage à destination de cette zone. Les requêtes de services (recherche/publication) ne peuvent cependant pas être capturées avec cette stratégie peu intrusive car les sondes doivent au préalable répondre à une requête de routage avant de recevoir la requête de service associée. Ces communications avec les autres pairs augmentent la visibilité des sondes sur la DHT. Une coopération entre les sondes permet cependant de réduire leur impact sur le réseau en ne capturant qu'une seule des 10 requêtes de service répliquées. Au final, cette approche souffre de la même limitation que les sondes de Bittorrent, à savoir que la connaissance des contenus, dont seul le hash est visible, reste difficilement accessible aux sondes. Par ailleurs, les requêtes capturées par les sondes peuvent être affectées par la pollution de la DHT et celles-ci ne permettent de constater précisément les transferts de fichiers.

3.3 Comparaison des méthodes de supervision

Parmi les méthodes de supervision présentées, la collecte de trafic est la plus difficile à mettre en œuvre. Son intérêt est évident pour comprendre les interactions entre le réseau Internet et les réseaux P2P mais il devient très limité lorsqu'il s'agit d'obtenir des informations sur le réseau P2P. Le traitement et le stockage des données applicatives pose des problèmes techniques et légaux importants et la vue du réseau P2P reste très limitée (au lien supervisé). Les mesures

sur serveurs permettent de superviser efficacement les réseaux P2P, notamment si un serveur important peut être instrumenté. Cependant, la vue du réseau reste là aussi partielle. En outre, l'évolution des architectures P2P a clairement montré que les serveurs tendent à disparaître au profit d'architectures complètement distribuées, en particulier les DHT, qui nécessitent d'autre moyens de supervision.

Pour les réseaux P2P structurés, trois méthodes de supervision sont possibles : les pots de miel, les explorateurs et les sondes distribuées. Les forces et faiblesses de chacune de ces approches sont présentées dans le tableau 3.2. Si l'on considère plus particulièrement la supervision de contenus partagés dans les réseaux P2P, aucune des solutions de supervision actuelles n'est satisfaisante. Les explorateurs permettent de découvrir la topologie du réseau mais ne renseignent qu'exceptionnellement sur les contenus partagés, en l'occurrence quand les utilisateurs autorisent explicitement l'inspection de l'ensemble de leurs fichiers. En outre, les explorateurs ne permettent pas d'observer les recherches ni les transferts de fichiers. Ils sont donc inadaptés pour superviser l'activité de contenus. Les sondes distribuées ont également un intérêt limité. Elles permettent en effet d'intercepter les requêtes de recherche et de publication mais n'ont pas la connaissance à priori des contenus ainsi supervisés. Augmenter le nombre de sondes pour superviser l'ensemble de la DHT rendrait la méthode très intrusive. Par ailleurs, les transferts de fichiers sont invisibles avec cette méthode, au contraire des pots de miels. Ceux-ci sont en effet capable de constater les transferts de fichiers. Le nombre de contenus malveillants supervisable par un honeypot est cependant limité. L'annonce de faux fichiers limitant énormément l'attractivité et l'intérêt des pots de miel, ces derniers doivent annoncer des contenus réels ce qui pose problème dans le cadre de contenus illégaux. Les pots de miel ont d'autres limitations importantes. Il est difficile d'étudier une thématique (un mot-clé particulier) avec cette approche car un honeypot ne peut découvrir les pairs partageant déjà un fichier. Pour finir, les sondes comme les honeypots sont sujets aux faux positifs du fait de la pollution de la DHT, certains fichiers pouvant être référencés par des mots-clés sans relation.

Méthode de supervision	Avantages	Inconvénients
Explorateurs	– Connaissance des pairs	– Connaissance anecdotique des contenus – Supervision active
Pots de miel	– Connaissance des demandes de fichier – Facilité de mise en œuvre	– Annonce de fichiers illégaux (supervision active) – Faux positifs (pollution) – Pas de connaissance des sources – Etude difficile des mots-clé
Sondes distribuées	– Connaissance des pairs – Supervision passive – Connaissance des contenus possible	– Faux positifs (pollution) – Pas de connaissance des demandes de fichier

TAB. 3.2 – Comparaison des méthodes de supervision appliquées aux contenus des réseaux P2P structurés

Chapitre 4

Sécurité dans les réseaux et services pair à pair

4.1 Introduction

4.2 Pollution

Un des problèmes majeurs des réseaux P2P de partage de fichier est la diffusion de la pollution en leur sein. Un fichier est dit pollué si le contenu fourni par le service P2P ne correspond à la description présentée à l'utilisateur. Plusieurs formes de pollution existent. Le contenu peut ainsi être valide mais sans rapport avec la description, partiellement dégradé, inexploitable un fois téléchargé ou encore, complètement fictif. La pollution dégrade notablement la qualité de service des réseaux P2P d'une part car l'utilisateur doit vérifier le contenu du fichier et rechercher un autre fichier sain et d'autre part, la diffusion de la pollution consomme inutilement les ressources limitées du réseau P2P. La pollution pose également des problèmes de sécurité. Ainsi, un contenu illégal ou malveillant (virus) peut être diffusé par des pairs malicieux avec le nom de fichiers exécutables légitimes (dessins animés). Nous étudions dans cette sous-partie les différentes recherches menées sur la pollution des réseaux P2P, en nous intéressant à la compréhension du phénomène et son impact sur la sécurité.

4.2.1 Stratégies de pollution

L'article [11] différencie la pollution normale qui consiste pour un pair à partager par inadvertance un fichier inutilisable, de l'empoisonnement « poisoning » qui consiste à injecter volontairement des leurres (fichiers corrompus) pour un contenu donné afin de réduire son intérêt. Les auteurs évaluent alors la disponibilité réelle de quinze contenus dans quatre réseaux P2P (Gnutella, Fasttrack , eDonkey et Overnet) victimes de différentes stratégies de pollution. L'injection de nombreux faux fichiers uniques, comparable à une pollution involontaire, n'affecte presque pas la disponibilité des contenus réels. Pour nuire à la disponibilité des fichiers réels, il faut que le nombre de faux fichiers injectés soient extrêmement élevé (99 faux fichiers pour 1 réel), de manière à remplir les réponses de recherche sans que des fichiers légitimes soient sélectionnés, ce qui nécessite des dizaines de milliers de fichiers injectés. Cette forme de pol-

lution est donc négligeable. La seconde stratégie de pollution consiste à annoncer peu de faux fichiers différents mais par de nombreux pairs, en les renouvelant périodiquement. Les faux fichiers sont alors mieux classés dans les résultats de recherches, difficilement détectables, et se diffusent facilement. Quelques centaines d'annonces permet ainsi de réduire la disponibilité des fichiers réels au profit des faux.

L'article [64] présente une autre forme de pollution, beaucoup plus élaborée qui est basée sur la collision de la fonction de hachage MD4 utilisée dans le réseau P2P eDonkey (la fonction MD5 présente également cette vulnérabilité). Les auteurs montrent qu'il est possible de générer instantanément un exécutable dont l'identifiant entre en collision avec un autre fichier partagé dans le réseau. Cette attaque est donc un moyen de diffuser des virus à l'insu des utilisateurs, même si ces derniers ont localisé la ressources via une URI. Une contre-mesure possible est la validation des fichiers par l'emprunte AICH également utilisée dans le réseau eD2k pour corriger les erreurs, ou, à long terme, un changement de la fonction de hachage mais ceci briserait la rétro-compatibilité entre clients. Cette attaque est cependant difficile à réaliser, la pollution du mécanisme d'indexation est beaucoup plus abordable et répandue.

4.2.2 Pollution du mécanisme d'indexation

La pollution du mécanisme d'indexation « index poisoning » consiste à diffuser des informations erronées dans l'index (centralisé ou distribué) des réseaux P2P au sujet des fichiers ciblés. Cette forme de pollution est également appelée « Metadata pollution » dans l'article [37]. Le pollueur crée, pour des mots-clés donnés, de fausses références de fichiers dans l'index ou de faux contacts (IP, port). La ressource ainsi polluée apparait comme inaccessible. Contrairement à l'injection de fichiers pollués, cette méthode nécessite très peu de bande passante, seuls des messages de signalisation étant nécessaires. Les auteurs de [39] étudient cette pollution dans deux réseaux P2P : Fastrack et Overnet. Ils recensent ainsi toutes les versions et copies de fichiers publiés pour dix contenus afin d'estimer leur niveau de pollution et les stratégies employées. Une simple analyse du nombre de publications pour même contenu émises par chaque pair permet d'identifier les pollueurs et indirectement les fausses entrées. Cette estimation est confirmée par une analyse automatisée des fichiers téléchargés. Les pollueurs ainsi identifiés représentent 7% des pairs mais 77% des fichiers publiés. Le coût d'une telle pollution est négligeable, en particulier pour Overnet dont les messages d'annonce sont envoyés par UDP.

Les auteurs [39] décrivent également une autre stratégie de pollution plus efficace pouvant être réalisée en insérant des nœuds pollueurs dans le réseau, de manière à les rendre responsables de l'indexation des contenus ciblés et ainsi générer directement des réponses corrompues. Concernant le réseau Overnet, qui est basé sur une DHT, l'injection de pairs malveillants dans la DHT partageant l'identifiant du mot-clé ou fichier à polluer permettrait de réaliser cette pollution. Cette étude omet cependant une autre forme de pollution importante du mécanisme d'indexation. Le mécanisme d'indexation peut en effet être corrompu en mélangeant l'indexation de fichiers existants, plutôt qu'en insérant des références de fichiers inexistants. Cette pollution peut être simplement réalisée en changeant le nom d'un fichier, puis en le partageant sur le réseau. Un objet pollué par cette méthode pourra être téléchargé mais son contenu est sans relation avec le nom du fichier sélectionné par l'utilisateur. Cette pollution est plus intrusive, car un fichier non souhaité est téléchargé par l'utilisateur, et engendre d'importants problèmes de sécurité (contenu offensant, virus). Par ailleurs, cette pollution n'est pas détectable par

les métriques présentées dans [39], car elle ne nécessite pas la création de nombreuses entrées erronées.

Il est à noter l'indexation des contenus P2P peut également se faire par des sites web. C'est notamment le cas du réseau P2P Bittorrent (via l'indexation de tracker), mais également de eDonkey/KAD dont les contenus peuvent être identifiés directement par des URI. Cette forme d'indexation est moins sujette à la pollution car elle ne dépend pas des annonces ou de l'indexation des pairs. Nous avons cependant montré que de tels sites web sont régulièrement victimes de poursuites.

4.2.3 Diffusion de la pollution

Afin de comprendre les facteurs influençant la diffusion de la pollution, l'article [15] modélise et simule deux formes de pollution, l'une ciblée sur un fichier particulier, l'autre affectant toutes les requêtes du réseau indifféremment, et ce, pour différents types d'architecture P2P. Les simulations montrent que la diffusion d'une pollution ciblée de faux fichiers est étroitement liée à la vigilance des utilisateurs et leur propension à rendre disponibles les fichiers téléchargés, et ce, indifféremment des architectures. La pollution globale simulée dans l'article est cependant assez peu réaliste. Elle consiste à insérer aléatoirement des pairs corrompant les recherches de sources (annonçant des pairs lents ou les attaquants eux-même). Les architectures P2P sans hiérarchie sont plus résistantes à l'insertion aléatoires de pairs malveillants car moins de nœuds sont contactés lors de la recherche, et chaque nœud. Cette étude est cependant trop générale dans les stratégies de pollution simulées et dans sa représentation des réseaux P2P pour bien appréhender la pollution, d'où la nécessité de mesures réelles.

L'article [37] établi ainsi un bilan précis de la pollution affectant le réseau P2P Kaazaa. Pour cela, les auteurs utilisent un explorateur émettant des requêtes sur tous les super-nœuds du réseau et découvrant tous les fichiers partagés étant donné un ensemble de mots-clés, puis, appliquent un algorithme de détection de pollution sur les quelques contenus étudiés (analyse du format d'encodage, de la durée...). L'analyse nécessite le téléchargement des fichiers mais est peu sujette aux erreurs d'appréciation. Il apparait que les fichiers populaires sont extrêmement pollués (62% des versions et 73% des copies), malgré le mécanisme de notation des fichiers fourni par les clients, alors que d'autres fichiers ne sont pas du tout affectés.

Les auteurs de [37] identifient les sources originelles de la pollution dans Kaazaa comme étant des sociétés polluant intentionnellement le réseau pour en réduire l'attrait. La pollution est ensuite diffusée par les pairs eux même. L'article [34] propose ainsi un modèle de propagation de la pollution au sein du réseau P2P Kaazaa, basé sur une étude du comportement des utilisateurs. Les 30 utilisateurs participant à l'étude à travers des questionnaires et l'utilisation d'un client instrumenté ont montré une faible réactivité vis-à-vis de la pollution, malgré une bonne connaissance des réseaux P2P et du problème. Ce manque de vigilance se traduit par une vérification tardive des fichiers téléchargés (plusieurs heures après obtention) ou trop superficielle (écoute incomplète) et participe largement à diffuser la pollution. Le modèle de propagation montre qu'une différence de vigilance de 20% peut décupler la pollution initiale et que, globalement, la pollution peut quadrupler la consommation de ressources dans le réseau.

4.2.4 Solutions possibles

Plusieurs solutions peuvent être envisagées pour lutter contre la pollution. L'article [37] classe les solutions potentielles en deux catégories : celles nécessitant le téléchargement du fichier (vérification manuelle ou semi-automatique) et celles pouvant être appliquée a priori (systèmes de confiance ou réputation), nous présentons ici des solutions du deuxième type. Les articles [38] et [39] proposent une méthode capable de détecter les sous-réseaux diffusant la pollution sur Kaazaa afin de filtrer leurs messages. La détection est basée sur l'hypothèse selon laquelle les adresses IP des pairs pollueurs annoncent un très grand nombre de versions différentes pour un même contenu. Cette méthode est efficace mais difficilement applicable à l'échelle d'un client. La collecte des différents fichiers existants nécessite en effet l'utilisation d'un explorateur, coûteux en ressources, et qui ne peux être exécuté par chaque client. Le filtre d'adresse IP pourrait être fourni a priori aux clients mais cela pose le problème de sa mise à jour (nouveaux pollueurs, nouveaux contenus pollués). Par ailleurs, d'autres formes de pollution (mélange d'indexation) ne sont pas détectables par la méthode proposée.

Des mécanismes de réputation ont été appliqués aux pairs ou aux contenus afin de limiter la pollution. L'article [12] présente un système hybride de réputation pouvant s'appliquer aux deux objets tout en ne faisant pas d'hypothèse sur la forme de pollution à combattre. Chaque pair évalue ses contacts en fonction de son expérience, et celle-ci est ensuite partagée entre pairs. Pour de grands réseaux P2P, une réputation locale est insuffisante car deux pairs ont une faible probabilité de se rendre service plusieurs fois. Cependant le mécanisme de réputation est fortement pénalisé si une partie des pairs réalise de mauvais vote. Ce mauvais vote peut être réalisé par des pairs honnêtes en cas de mélange d'indexation : un contenu partagé avec deux noms de fichiers distincts peut apparaître sain pour certains pairs et pollué pour d'autres. Des attaques volontaires peuvent également nuire au mécanisme de réputation (entente malveillante, attaque Sybil). Outre le surcoût engendré, l'efficacité relative des systèmes de réputation limite donc leur intérêt.

Les auteurs de [30] et [40] explorent une autre méthode pour lutter contre le mélange d'indexation basée sur l'analyse des noms de fichiers. Ils proposent ainsi quatre métriques susceptibles de détecter si un contenu est victime du mélange d'indexation en considérant les mots-clés communs aux différents noms de fichiers associés au contenu. Des contenus dont les noms de fichier sont très différents ont un indice de pollution élevé et sont considérés comme pollués au delà d'un seuil. Les auteurs appliquent chaque métrique sur les contenus partagés au sein du réseau P2P eDonkey (obtenus par l'explorateur présenté dans le section 3.2.4) afin déterminer si une métrique est discriminante et quel seuil de détection doit être appliqué. Cette approche est prometteuse mais encore incomplète. Il est en effet important de pouvoir estimer précisément les faux positifs et négatifs engendrés par cette détection. Or, il est facile d'exhiber, pour chacune des métriques, des cas ou un contenu sain est considéré comme pollué car un seul utilisateur le partage avec un nom différent. Plusieurs points doivent donc être améliorés pour en faire une solution viable contre la pollution, notamment la prise en compte du nombre de pairs affichant chaque nom de fichier trouvé ou encore l'obtention des différents noms sans recours à un explorateur.

Au terme de cette partie, il apparaît que la stratégie de pollution mélangeant l'indexation des fichiers a été très peu étudiée dans la littérature, malgré les problèmes importants de sécurité qu'elle pose. Plus particulièrement dans le cas de KAD, aucune étude n'a été réalisée sur la

quantification et la lutte contre la pollution. Par ailleurs, l'état de l'art ne propose pas de solution directement applicable capable de protéger un client en temps réel contre la pollution. Il y a donc un réel besoin de méthodes de supervision et de contre-mesures capables de détecter une pollution mélangeant l'indexation des fichiers et d'en protéger les utilisateurs.

4.3 Attaque Sybil

4.3.1 Attaque Sybil

Principe

En présentant les réseaux P2P, nous avons établit que la réalisation d'un service fiable par un ensemble de pairs non-fiables (pannes, comportements malveillants) est assurée par trois éléments :

- le fait que les pairs soient indépendants les uns des autres
- le fait que les pairs soient aléatoirement répartis sur l'overlay
- l'utilisation de mécanismes de réplication

La réplication est primordiale tant au niveau routage (entrées multiples dans la table de routage) afin de maintenir la connectivité des pairs, qu'au niveau service (indexation d'une donnée répliquée sur plusieurs pairs) afin d'assurer sa continuité. La réplication, associée au fait que les pairs soient indépendants et aléatoirement répartis, assure une distribution des services P2P au niveau géographique (une donnée est indexée par des pairs de différents pays) et limite l'impact des pairs malveillants : tant que le nombre de pairs malveillants coordonnés reste faible à l'échelle du réseau, une donnée est majoritairement indexée sur des pairs sains. Le service devient ainsi robuste aux défaillances ponctuelles des pairs ou de parties du réseau IP. L'attaque Sybil met à mal les deux premières hypothèses et, en leur absence, les mécanismes de réplication ne suffisent plus à assurer la fiabilité des services P2P.

L'attaque Sybil, telle que décrite initialement par Douceur [14], consiste pour une même entité à insérer un grand nombre de pairs dans le réseau. Les pairs ainsi générés ne sont pas indépendants et l'entité qui les a créés peut représenter une part significative du réseau. Une donnée répliquée en théorie sur plusieurs pairs indépendants peut de ce fait être uniquement indexée par des sybils. La faiblesse des réseaux P2P permettant cette attaque est le peu de contraintes imposées sur la génération des identités des pairs. Une même entité peut ainsi créer autant d'identités différentes connectées au réseau que ses ressources le lui permettent. Douceur montre qu'en l'absence d'une autorité de certification centralisée des identités, l'attaque Sybil ne peut être évitée que ce soit par une identification réalisée indépendamment par chaque pair ou partagée. Cependant, des mécanismes peuvent être conçus pour limiter l'ampleur de l'attaque en la rendant plus difficile.

Telle que définie dans [14], l'attaque Sybil viole seulement l'indépendance des pairs. En pratique, ses nombreuses mises en oeuvre impliquent également un positionnement non-aléatoire des pairs insérés, si bien que l'attaque Sybil comprend par extension cet élément. Le fait d'insérer les pairs à des endroits spécifiques du réseau rend l'attaque plus efficace en augmentant localement la représentativité de l'attaquant dans le réseau. Ceci permet plusieurs applications introduites par [7] telles que : prendre le contrôle d'une donnée indexée sur une DHT (tous les réplicats sont stockés par les pairs insérés) ou isoler un pair du réseau (tous ses contacts sont

des pairs insérés). Nous présentons de telles applications de l'attaque Sybil dans le paragraphe suivant.

Applications

L'article [58] présente une attaque Sybil réalisée à grande échelle sur le réseau KAD. L'attaque nécessite deux étapes. Les auteurs utilisent tout d'abord un explorateur afin de découvrir l'ensemble des pairs du réseau, puis, ils injectent ensuite les attaquants directement dans la table de routage de chaque pair en créant les identifiants des attaquants de manière à occuper une place précise dans la table de routage de la victime. Les auteurs ont ainsi inséré depuis une unique machine 2^{16} attaquants dans une zone de la DHT contenant d'ordinaire 8000 pairs (càd $1/256^{eme}$ du réseau). Cette attaque leur permet d'intercepter la grande majorité des requêtes de recherche et de publication à destination de cette zone, et ainsi d'en contrôler les services. Cette attaque est cependant très intrusive mais les auteurs ont montré que des attaques beaucoup plus localisées sont également possibles.

Celles-ci consistent à prendre le contrôle de données indexées sur la DHT en choisissant précisément l'identifiant des pairs insérés de manière à les rendre responsables de la donnée ciblée. Selon la manière dont est établie la responsabilité du stockage ou de l'indexation des données entre les pairs du réseau, la création des identifiants varie. Dans Chord, les attaquants sont insérés de manière à être les plus proches successeurs de l'identifiant de la donnée visée. Dans Kademlia, les attaquants choisissent leur identifiant de manière à être proches de la donnée selon la métrique XOR. Les auteurs de [58] ont ainsi montré qu'il était possible d'éclipser une donnée très populaire du réseau KAD, en l'occurrence le mot-clé « the », avec seulement 32 attaquants. Les attaquants insérés deviennent ainsi responsables de la donnée en recevant toutes les requêtes de publication relatives. Il leur suffit de dénier les demandes de recherche pour l'identifiant ciblé pour que les pairs du réseau ne puissent plus accéder à l'information éclipsée. Toute recherche de contenus utilisant le mot-clé éclipsé aboutit à l'absence de résultats, bien que les fichiers soient toujours partagés par les pairs. L'article [27] perfectionne encore davantage l'attaque éclipse dans KAD en créant une chaine d'attaquants dont la proximité est toujours croissante avec l'identifiant ciblé. De cette manière, la recherche d'un pair progresse indéfiniment vers l'identifiant sans jamais l'atteindre et l'information n'est pas trouvée quand le délais alloué à la recherche expire.

Les auteurs de [7] et [55] présentent une autre forme d'attaque éclipse ne visant pas à faire disparaître une donnée de la DHT mais à déconnecter un ou plusieurs pairs du reste du réseau ou à partitionner celui-ci. Pour réaliser cette attaque, étant donné un ensemble de pairs victimes à déconnecter, les sybils doivent être insérés de manière à remplir intégralement la table de routages de ces pairs. Tout accès des victimes au réseau P2P est alors contrôlé par les attaquants. Une telle attaque a été mise en oeuvre avec succès contre le réseau KAD [66] et a montré qu'il était ainsi possible de partitionner le réseau avec peu de ressources. Au delà des DHT, l'attaque Sybil a également été prouvée comme étant efficace pour perturber l'échange de fichier au sein d'un swarm du réseau Bittorrent [28]. Les attaquants insérés dans le swarm peuvent alors mentir sur la disponibilité des parties du fichier et en ralentir ainsi la diffusion. Cette attaque est d'autant plus efficace que le nombre d'attaquants est élevé et que ceux-ci sont insérés tôt dans la vie de du swarm.

Enfin, et contrairement aux précédentes applications, l'attaque Sybil n'est pas toujours uti-

lisée à des fins malveillantes. Ainsi, nous avons présenté, dans la section étudiant les différentes méthodes de supervision pour les réseaux P2P, une méthode de supervision particulière basée sur l'injection de sondes. Or, l'injection de multiples sondes dans le réseau P2P par une entité souhaitant le superviser est équivalent à une attaque Sybil. Sans entrer à nouveau dans les détails, nous rappelons ces attaques Sybil à des fins de supervision ont été appliquées avec succès sur plusieurs réseaux P2P dont Gnutella [26] [1], Bittorrent [16] [67] et KAD [46]. Les pairs insérés dans ces expériences ont adopté un comportement normal (mis à part la supervision des messages) et n'ont, par conséquent, pas dégradé le réseau.

L'attaque Sybil est donc un des problèmes de sécurité majeurs des réseaux P2P. Cette menace pèse indifféremment sur toutes les architectures P2P. Cependant, elle s'est avérée particulièrement puissante dans le cadre des réseaux P2P structurés où des applications ont montré qu'il était possible de faire disparaître des contenus indexés sur le réseau ou encore de partitionner celui-ci. Bien que ceci n'est pas été réellement mesuré, le contrôle du mécanisme d'indexation par une attaque Sybil peut également permettre la diffusion massive de pollution. Il est donc primordial de pouvoir détecter et superviser de telles attaques afin d'identifier les pairs malveillants pour en protéger les pairs légitimes.

Chapitre 5

Approches collaboratives pour la sécurité des services

Suite à l'échelle démesurée qu'a prise Internet, ce réseau est devenu la cible quotidienne de nombreuses attaques. Alors qu'elles étaient au nombre de 6 par jour en 1988, en 2003 nous en recensions 137 529. Il en va de même pour les failles de sécurité qui étaient de 171 en 1991 et de 7 236 en 2007[1]. Alors qu'il est possible de détecter une attaque ou intrusion au sein d'un réseau à l'aide d'un système de détection d'intrusion (IDS[2]) lorsque la source de l'attaque est unique ou faiblement distribuée, il devient impossible pour ces IDS classiques de détecter les attaques coordonnées.

Dans ce chapitre, nous allons décrire les mécanismes qui permettent de détecter ces nouveaux types d'attaques distribuées, comme par exemple les attaques de deni de service distribuées (DDoS[3]). Il existe un très grand nombre de botnets actifs sur Internet, qui sont utilisés principalement pour envoyer des messages de SPAM et effectuer des attaques en DDOS. Afin de détecter ces attaques qui proviennent de réseaux multiples et géographiquement distribués, il est impératif de combiner les preuves révélant une activité anormale au sein des réseaux et d'identifier la source de ce problème.

Dans la section suivante, nous présentons une vue globale sur les attaques distribuées. La section 5.2 sera consacrée à la présentation de l'ensemble des principales architectures des systèmes de détection d'intrusions collaboratives. Enfin, ce chapitre se terminera par une conclusion générale sur la problématique et la nécessité de faire coopérer les IDS d'une manière éfficace.

5.1 Les attaques distribuées

Les attaques réseaux sont pour la plupart lancées automatiquement à partir de machines infectées par des virus, chevaux de Troie ou vers, à l'insu de leur propriétaire. Dans ce contexte, les attaques de déni de service sont parmi les plus courantes et relativement faciles à mettre en oeuvre. Ce type d'attaques a des conséquences économiques importantes par ses effets.

[1]Source : *Computer Emergency Response Team* (CERT), 2006
[2]Intrusion Detection System
[3]Distributed Deny of Service

5.1.1 Les attaques de déni de service distribuées

Le déni de service distribué est une attaque générée par une ou plusieurs machines. Comme le montre la figure 5.1, elle consiste à inonder, de façon concertée et à partir de plusieurs sources, une machine cible par des paquets de types TCP-SYN, ICMP ou/et UDP.

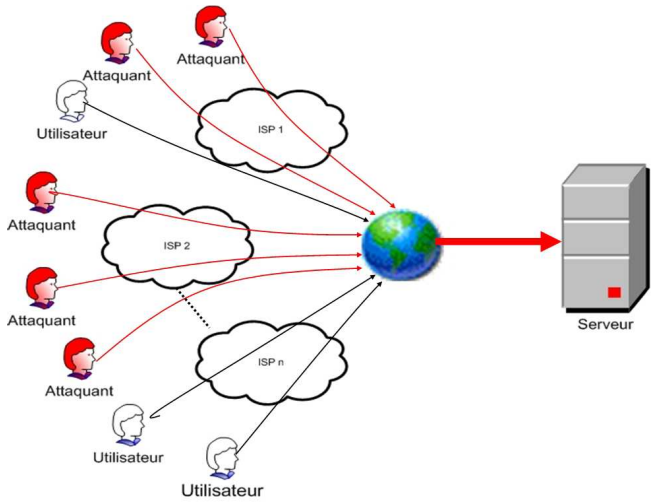

FIG. 5.1 – Exemple d'attaque de déni de service distribuée

D'après [47], les attaques DDoS sont définies par : *"an overwhelming quantity of packets being sent from multiple attack sites to a victim site. These packets arrive in such a high quantity that some key resource at the victim (bandwidth, buffers, CPU time to compute responses) is quickly exhausted"*. Dans des conditions normales, un serveur offrant des services à ses clients doit traiter des requêtes provenant de plusieurs clients. Lorsque ces derniers ne peuvent en bénéficier pour des raisons anormales provoquées par un tiers, on parle alors de déni de service distribué parce que le serveur ne peut plus servir les clients ou il prend plus de temps pour répondre ou traiter les requêtes. On parle alors de défaillance totale et défaillance partielle, respectivement.

Ces défaillances sont dues à la consommation des cycles CPU[4] et de la bande passante de la victime [47]. Les victimes d'une attaque DDoS peuvent être tout équipement offrant un service applicatif ou d'interconnexion comme par exemple, un serveur web ou un routeur. Pour résumer :

- l'attaque DDoS est une attaque destinée à limiter ou empêcher le bon fonctionnement ou la disponibilité d'une machine ;
- l'attaque DDoS consiste à paralyser temporairement des serveurs afin qu'ils ne puissent plus ou mal fournir leur service ;
- l'attaque DDoS est une consommation artificielle de la mémoire ou de la bande passante de la victime ;
- l'attaque DDoS utilise un grand nombre de nœuds pour réaliser l'attaque.

[4]Central Processing Unit

5.1.2 Historique des attaques majeures

De temps à autre, les médias signalent une attaque DDoS contre un site Internet bien connu. Ce paragraphe présente une brève synthèse d'attaques DDoS qui ont été considérées comme majeures. Nous ne citons ces attaques que pour montrer leur efficacité, leurs types et leurs particularités (protocoles, débit, etc) :

2000 : Au mois de février 2000, un certain nombre de grands sites américains (Yahoo, Buy, Stamps, eBay, CNN, Amazon, MSN et ZDNet) ont été victimes des attaques de type déni de service (DoS). Pour ces victimes, les pertes ont été majeures[5] avec des coûts avoisinant 1.7 milliards de dollars americains ;

2002 : Le 21 octobre 2002, une attaque DDoS contre les serveurs root de noms de domaines du service DNS[6] a été déclenchée afin de paralyser le réseau mondial par l'impossibilité d'accéder aux serveurs indexés [65]. Effectivement, 7 des 13 serveurs DNS racines ont fait l'objet d'une attaque DDoS par des paquets ICMP, TCP-SYN et UDP. Chaque serveur était victime d'une attaque DDoS avec un débit variant entre 50 Mbps et 100 Mbps. Le débit total de l'attaque sur les serveurs était approximativement de 1.8 Mpaquets/s. Les sources des paquets utilisées par ces attaques étaient des adresses falsifiées (spoofed), et ceci afin de camoufler l'origine de ces attaques.

2003 : Une attaque DDoS a été lancée contre le site de la chaîne de télévision Al-Jazeera. Les serveurs Web d'Al-Jazeera ont été inaccessibles temporairement pendant quelques jours ;

2004 : Au mois de juin, une attaque DDoS a été lancée contre plusieurs serveurs tenus par la société américaine spécialisée dans la mise à disposition des entreprises de serveurs de cache *Akamai Technologies*. Une centaine des machines contrôlables à distance ont été utilisées pour mettre en oeuvre l'attaque. L'attaque a rendu inaccessibles plusieurs sites clients de l'entreprise ;

2005 : Au mois d'avril, tous les serveurs du jeu Final Fantasy XI ont été indisponibles durant 3 heures[7] ;

2007 : Au mois de mai, l'Estonie a été victime d'une attaque DDoS massive et coordonnée. Les cibles de ces attaques étaient des serveurs du gouvernement, des entreprises de télécommunications, etc. Les services fournis par ces sites ont été indisponibles pour quelques heures et voire quelques jours pour certains d'entre eux[8]. Ces attaques ont obligé certaines organisations du pays à couper l'accès de leur site web à toutes les IP étrangères au pays pendant plusieurs jours.

2008 : Une attaque DDoS a été lancée contre les sites web de *Radio Free Europe-Radio Liberty (RFE/RL)* qui sont une radio et un groupe de communication privés. L'attaque a rendu impossible la consultation de ces sites par les internautes[9] et a bloqué la diffusion des émissions de cette radio. Le débit de l'attaque était approximativement 50000 requêtes /sec, ce qui est un débit extrêmement élevé. Le 19 avril 2008 le site américain CNN[10]

[5]Notable hacks (http ://www.pbs.org/wgbh/pages/frontline/shows/ hackers/whoare/notable.html)
[6]Domain Name System
[7]Final Fantasy XI' under attack (news.cnet.com)
[8]Cyber Assaults on Estonia Typify a New Battle Tactic (Washington Post Foreign Service)
[9]RFE/RL Websites Hit By Mass Cyberattack (http ://www.rferl.org/content/article/1109642.html)
[10]CNN Web site targeted (http ://edition.cnn.com/2008/TECH/04/18/cnn.websites/)

était victime d'une attaque DDoS par un débit de 14 Mo de données par seconde. Des ralentissements sur le site ont été ressentis par les internautes asiatiques. Bien que CNN avait été prévenu avant l'attaque, les contre-mesures qui étaient mises en place n'ont pas protégé le serveur.

2009 : Une attaque DDoS a été lancée par une Cybermilice russe contre le Kirghizistan. L'attaque a isolé complètement trois FAI dans ce pays pour plusieurs heures.

Ces exemples démontrent à quel point ce type d'attaques est nuisible et coûteux. Par exemple, un site web de commerce peut être à l'arrêt pour une période donnée entraînant des pertes financieres conséquentes, auxquelles s'ajoutent le travail nécessaire pour restaurer le système. Enfin, les DDoS sont des attaques extrêmement dangereuses et relativement courantes et il n'est pas rare de voir apparaître une nouvelle attaque de ce type. Les dégâts logiques que font ces attaques sont potentiellement forts puisque des équipements principaux des réseaux tels que les routeurs sont susceptibles d'être victimes du DDoS. Pour ces raisons, ces attaques sont considérées comme des crimes dans certains pays du monde comme par exemple, en Grande Bretagne[11] et en Suède[12].

5.2 Les IDS collaboratifs

Une intrusion est définie comme un ensemble d'actions qui tentent de compromettre la confidentialité, l'intégrité ou la disponibilité d'une ressource. La détection d'intrusions permet de détecter des activités anormales en analysant le trafic réseau. Cette détection est réalisée par un IDS qui peut être utilisé pour détecter une grande variété d'actions comme le débordement de tampon (*buffer overflow*), le balayage de port (*port scan*), l'essai de prise d'empreinte du système d'exploitation (*OS fingerprintings*), etc. Dans la section suivante, nous présentons les classifications existantes et les caractéristiques des IDS.

5.2.1 Classification des IDS

Comme il est illustré dans la figure 5.2, les IDS peuvent être classifiés selon plusieurs critères comme par exemple, les techniques de détection, l'environnement de détection, le type de réaction, la méthode de capture de paquets, etc.

En outre, une autre classification existe. Cette dernière se base sur l'environnement de détection en spécifiant trois types d'IDS :

1. les NIDS (Network Based Intrusion Detection System) qui surveillent l'état de la sécurité au niveau du réseau ;
2. les HIDS (Host Based Intrusion Detection System) qui surveillent l'état de la sécurité au niveau des hôtes ;
3. Les IDS hybrides quant à eux utilisent les NIDS et HIDS pour avoir des alertes plus pertinentes.

[11] Legal Information Service (www.jisclegal.ac.uk/cybercrime/Archived_cybercrime.htm)
[12] Les attaques DDoS bientôt un crime en Suède (http ://www.journaldunet.com)

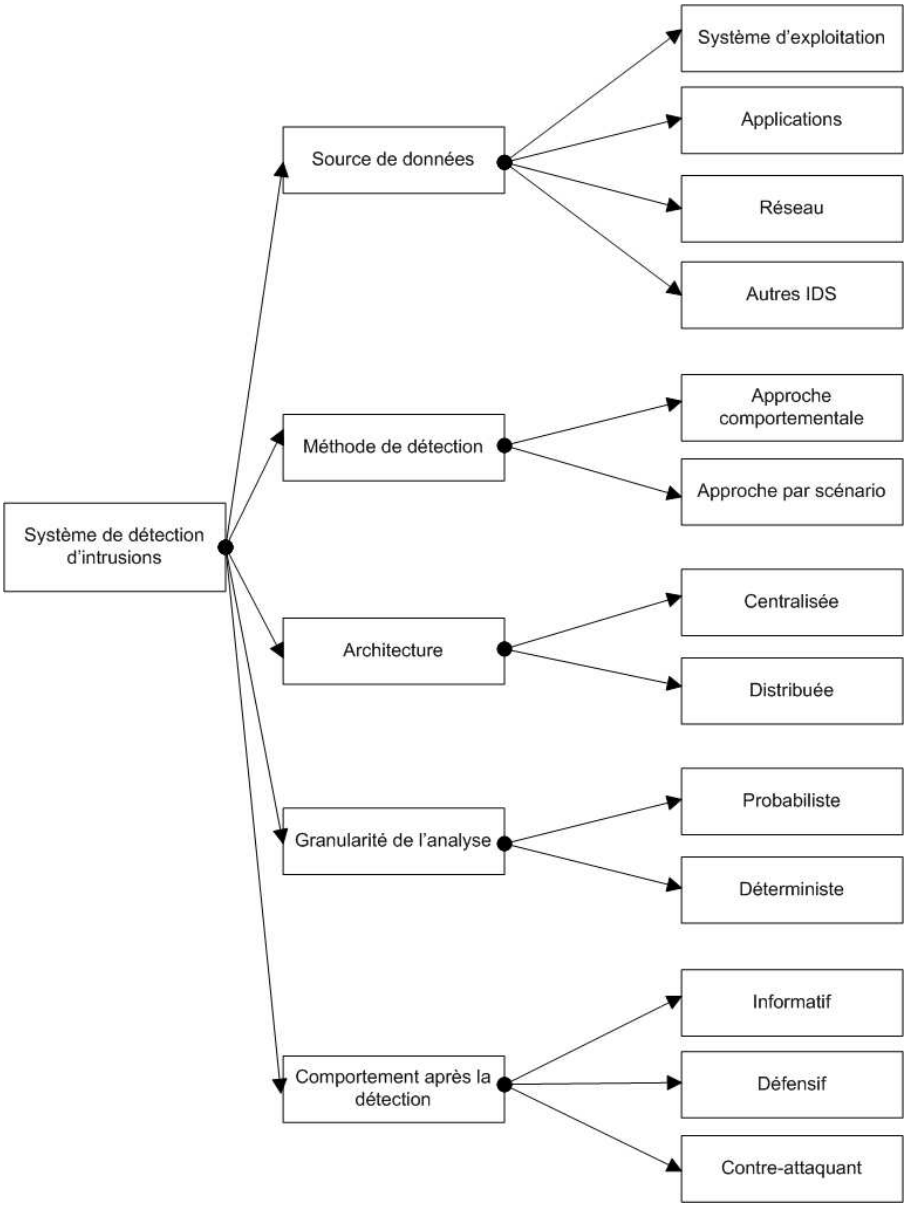

FIG. 5.2 – Classifications des IDS

Le HIDS focalise ses analyses et ses détections sur les activités internes d'un système local plutôt que sur les interfaces externes (son réseau). Par contre, le NIDS fait l'analyse en temps réel du trafic réseau, mais peut également fonctionner en mode passif. Le NIDS écoute tout le trafic de son réseau. Il analyse aussi le trafic et génère les alertes en cas d'anomalies (balayage de ports, par exemple). En outre, le NIDS fonctionne à toutes les couches couches protocolaires applications, transport et réseau. En outre, il est capable de décoder les paquets au niveau 2, comme le système de détection Snort.

Les problèmes majeurs des NIDS sont : l'équilibre de charge, la localisation, la détection de nouvelles attaques et les fausses alarmes (positives et négatives). Concernant ces dernières, la différence entre ces deux types d'alarmes est que les faux positifs surviennent lorsque l'IDS considère certaines activités légitimes comme intrusives ou malveillantes, et les faux négatifs se révèlent lorsque l'IDS ne détecte pas une vraie intrusion ou une vraie malveillance. En pratique, le terme de faux négatifs est utilisé pour décrire l'incapacité de l'IDS à détecter les intrusions dans certaines circonstances.

5.2.2 Architectures existantes

La recherche académique ainsi que les industriels ont proposé de nombreux mécanismes plus ou moins adaptés à la détection d'intrusions et d'attaques distribuées. Une comparaison entre ces mécanismes est assez complexe, car ils ne sont pas équivalents en termes de fonctionnalités et d'architectures. Nous présentons, dans ce qui suit, les principales architectures existantes.

Aggregate-based Congestion Control (ACC)

Cette approche a été proposée par [42] en 2002. L'idée principale se divise en deux techniques : Local Aggregate Congestion Control (LACC) et les messages Pushback. Nous détaillons cette approche puisqu'elle représente un modèle coopératif dans lequel chaque routeur peut bénéficier d'informations de détection fournies par d'autres routeurs. En son sein, un routeur détecte une attaque DDoS en se basant sur les caractéristiques du trafic à l'aide du LACC et le fait savoir en utilisant la technique Pushback aux routeurs voisins possédant les mêmes caractéristiques du trafic. Par conséquent les routeurs vont réduire la bande passante de ce trafic (Figure 5.3).

Local Aggregate Congestion Control (LACC) : LACC est une proposition de méthode de reconnaissance de congestion dont le principe est le suivant. Les routeurs surveillent le taux de rejet de paquets pour chaque file d'attente afin de vérifier qu'il ne dépasse pas un seuil défini par l'administrateur et qui représente un signe de congestion. Lorsqu'une congestion est détectée par un routeur, ce dernier définit l'agrégat du trafic suspect. Un agrégat est une collection de paquets d'un ou plusieurs flux, qui possède le même préfixe d'adresse IP destination.

La détection s'appuie sur les flux qui possèdent le même préfixe d'adresses IP destination. LACC calcule le taux d'arrivée des paquets par rapport aux paquets rejetés par le routeur. Si ce taux dépasse un seuil déterminé, l'agrégat est considéré comme illégitime. Par conséquent, LACC suppose que cet agrégat forme une signature de congestion et commence à appliquer à chaque paquet coïncidant avec cette signature de congestion une limite de débit se traduisant par le placement dans une file d'attente virtuelle.

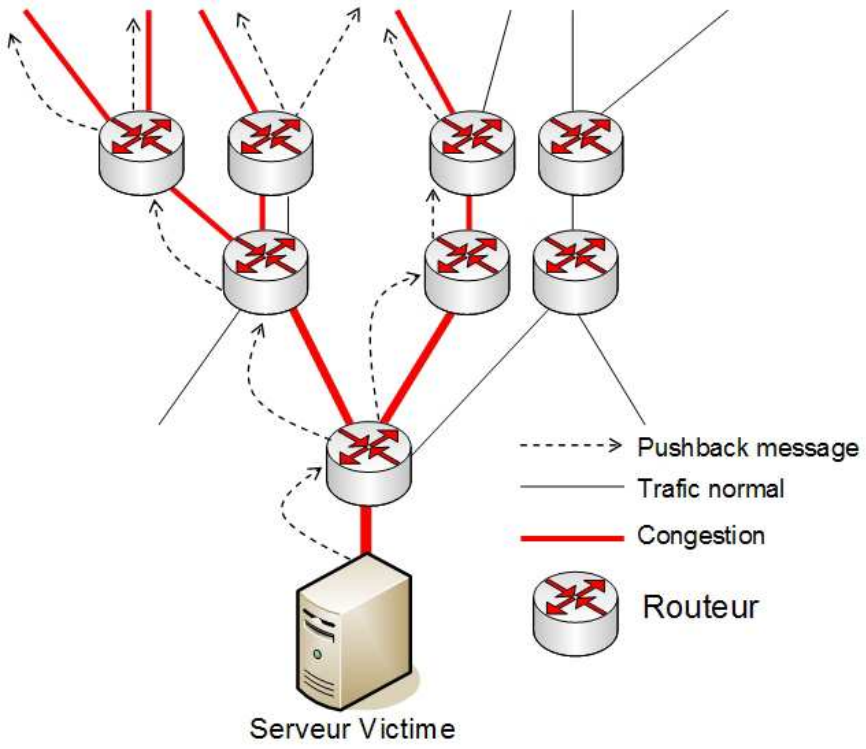

FIG. 5.3 – Détection d'une attaque DDoS par l'approche *Aggregate-based Congestion Control*

Les messages Pushback : ACC permet à un routeur ayant défini un agrégat de demander à tous les routeurs en amont (*upstream*) de limiter le trafic appartenant à cet agrégat. Cette limitation peut se propager jusqu'à l'origine de la congestion. Par conséquent, ces messages font la traçabilité de l'attaque DDoS. Si un trafic particulier entraîne la congestion, un Pushback propagé permettra de savoir exactement quel trafic est envoyé vers le routeur victime. Ceci va aider le routeur congestionné à prendre une décision et à rapporter le statut des routeurs.

L'avantage d'ACC est qu'il permet de décongestionner même le cœur du réseau, ce qui était impossible avec les autres techniques de détection de congestion. Les résultats d'implémentation de Pushback [23] sont positifs même si une partie du trafic légitime est perdue.

Pour conclure, malgré les avantages de Pushback, cette approche possède trois inconvénients :

1. Déploiement : il faut que tous les routeurs aient la technique du contrôle de la congestion basée sur les agrégats (ACC) ;
2. Ressources : chaque routeur doit maintenir pour une période donnée l'état du trafic [48] ;
3. Détection : si les sources de l'attaque sont très distribuées, le volume de trafic illégitime détecté par les routeurs *upstream* ne sera pas important, ce qui représente un inconvénient important de l'approche Pushback ;

Cooperative Intrusion Traceback and Response Architecture (CITRA)

L'architecture CITRA[13] [60] fournit une nouvelle infrastructure pour l'intégration des systèmes de détection d'intrusions, des pare-feu et des routeurs afin de remonter les attaques à leur source et de les bloquer le plus proche possible de celle-ci.

Cette infrastructure est développée pour améliorer l'automatisation de la détection et la réaction. L'objectif de cette automatisation est de faciliter la prise immédiate de mesures lorsque l'attaque est en cours. En effet, un administrateur prend beaucoup de temps pour analyser le trafic, alors que les dommages causés par l'attaque peuvent être énorme. Pour analyser et contrer les attaques, il est nécessaire d'avoir beaucoup d'informations sur le trafic, à l'aide de systèmes de gestion de réseau, des routeurs, des commutateurs et d'autres systèmes. Dans ce contexte, CITRA est une approche concrète de la mise en œuvre de ce mécanisme.

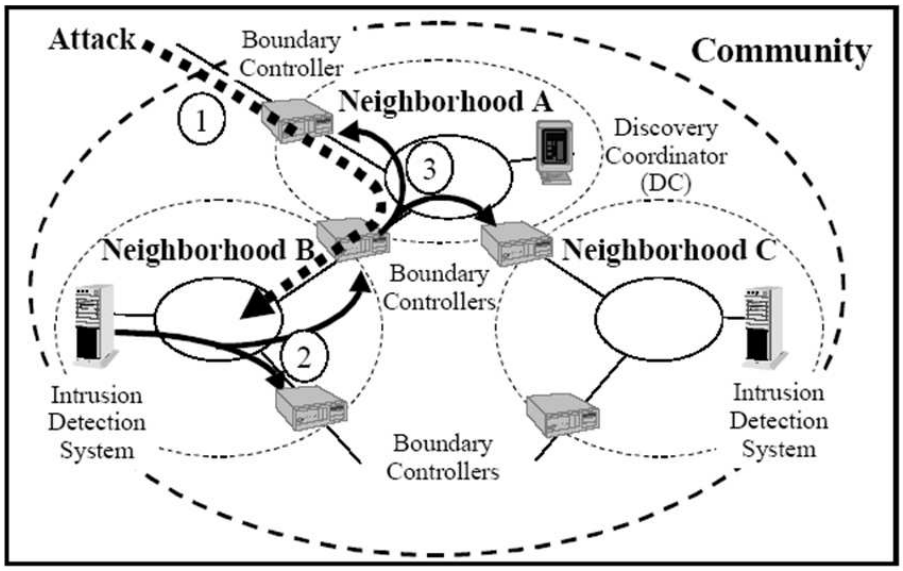

FIG. 5.4 – Extrait de [60]. Architecture de l'infrastructure CITRA

CITRA repose sur un protocole de la couche application, appelé IDIP[14] qui définit les informations permettant aux équipements de CITRA d'échanger des messages pour détecter les attaques. IDIP [53] est conçu pour fournir un transport fiable et sécurisé pour la communication des messages. Il repose sur le protocole IPSec, pour se protéger contre les attaquants et de garantir l'intégrité des messages émis.

Comme le montre la figure 5.4, CITRA repose sur le composant *discovery coordinator* qui gère la détection, ainsi que la remontée à la source de l'attaque. Le *discovery coordinator* gère l'ensemble des domaines qui sont appelés *neighborhood*. Ces derniers sont reliés par un autre composant appelé *boundary controller* qui est responsable du contrôle de la frontière entre deux domaines. Ce dernier sert à transmettre les messages via IDIP à tous les *neighborhood*.

Le déploiement de cette architecture est une limite parce qu'elle demande aux domaines différents de travailler ensemble et de partager des informations.

[13]Cooperative Intrusion Traceback and Response Architecture
[14]Intruder Detection and Isolation Protocol

Worminator

Worminator [41] a été crée afin de résoudre le problème de la génération de très grand nombre d'alertes par les IDS. Son objectif est de se focaliser sur les attaques les plus sophistiquées et les plus dangereuses : les scans furtifs et les propagations de vers. Actuellement, on peut affirmer que même les meilleurs IDS ne détectent pas les activités lentes et furtives étalées dans le temps et l'espace car ils produisent des événements qui rendent la detection des attaques par vers et le scan de grandes plages d'IP très difficiles. En effet, il n'est pas trivial de repérer ce genre de danger depuis un point du reseau par rapport au large réseau qui est Internet.

Avec l'augmentation des *blacklists*, de vers et les zero-day vers[15], il y a un besoin d'identifier rapidement les attaques générées des sources distribuées. Afin de détecter la propagation de zero-day vers et les scans furtifs on peut noter que :

1. si un scan est détecté par un IDS, cela ne peut avoir aucune signification ;
2. si des scans similaires se passent depuis la même source et sont détectés par des IDS se situant sur d'autres sites, on considère que ce n'est pas juste du bruit ou une "coïncidence" ;
3. si un scan est détecté sur plusieurs sites (5.5) mais pas tous, cela ressemble plus à un scan ciblé qu'à un ver.

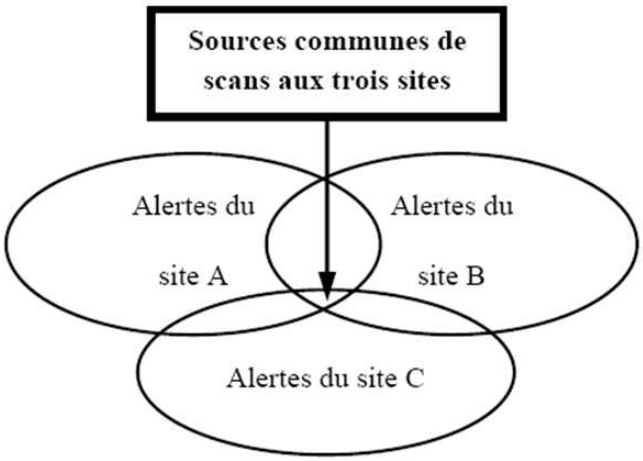

FIG. 5.5 – Corrélation d'alertes générées par plusieurs sites

Deux mécanismes sont utilisés dans Worminator. Le premier est la construction de filtres de Bloom qui sont utilisés pour protéger la confidentialité des données pendant leurs échanges entre les différents sites. Le deuxième est un mécanisme de contrôle d'échange d'informations qui est effectué par un algorithme de planification des corrélations dans un système distribué. Cet algorithme de planification calcule dynamiquement les paramètres de communication entre les nœuds afin de déterminer quels nœuds devraient communiquer pour échanger ses filtres de Bloom. L'utilisation de ces filtres très compacts assure une utilisation optimale de la bande

[15] c'est-a-dire les vers dont la signature n'est pas référencée par les éditeurs de solutions antivirales

passante ainsi que des ressources de la machine. Dans ce modèle des *watchlists* (listes d'adresses IP et de ports sources suspectées d'avoir un mauvais comportement) sont utilisées. Le but du système de détection distribué n'est pas d'analyser le réseau ou les évènements des stations d'autres sites, mais plutôt de corréler le résumé des alertes pour identifier les attaquants. C'est pourquoi, les *watchlists* encapsulent les informations appropriées en vu de leur échange. Il y a plusieurs désavantages quant au partage des alertes dans un système distribué : les organismes peuvent avoir des règles de politiques d'alertes privées et/ou ne pas vouloir partager leurs adresses IP afin de cacher qui communique avec qui ; le deuxième désavantage est que la taille des fichiers d'alertes augmente en fonction du trafic. Tant que les paramètres ne réduisent pas le bruit, une solution préférable serait de coder les informations importantes de façon compacte et utile. Pour cela, Worminator utilise le filtre de Bloom qui est une structure de données sous forme de hash qui fournit deux fonctionnalités : l'insertion et la vérification. Alors qu'aucune donnée ne peut être extraite après avoir été insérée dans un filtre de Bloom, il est possible de façon probabiliste d'identifier si des données ont été insérées ou non.

Les filtres de Bloom apportent une structure compacte de données : moins de 10 kbit sont capables de vérifier des dizaines de milliers d'entrées. Quand le filtre de Bloom est saturé, il donne des faux positifs mais jamais de faux négatifs (dans ce cas, il faut corréler à nouveau les listes d'alertes). Ils apportent aussi un très haut niveau de sécurité car les entreprises peuvent corréler les *watchlists* sans révéler les adresses IP et les incite ainsi à participer à la correlation. L'implementation de Worminator montre que l'utilisation des filtres de Bloom fonctionne bien en déterminant quelles sources sont considérées comme suspectes parmi l'ensemble des alertes recueillies en un site et permet ainsi d'éliminer le bruit de l'IDS du dit site. Worminator se révèle être un outil efficace pour détecter les IP sources suspectées de scanner un large réseau grâce à la corrélation d'alerte effectuée avec les filtres de Bloom. Par contre, l'inconvénient de cette approche est la génération d'une grande quantité d'information en cas d'attaque.

Distributed Overlay for Monitoring InterNet Outbreaks (DOMINO)

DOMINO [69] est une solution distribuée et repartie pour le monitoring sur l'Internet. Domino est un système sécurisé d'échange de données composé des détecteurs d'intrusion hétérogènes repartis à travers l'Internet. L'objectif de DOMINO est d'améliorer la détection et le temps de détection des codes malicieux sur le réseau. Un réseau DOMINO est une architecture dynamique composée de nœuds situés à travers l'Internet. Le but de ce type d'architecture est de proposer un système de partage d'information pour améliorer la capacité de détection d'intrusion de tous les participants. Comme présenté sur la figure 5.6, l'architecture DOMINO est composée de trois parties : les axes superposés, les satellites de communication et les contributeurs terrestres.

- les axes superposés centraux : (AXIS OVERLAY) Les nœuds centraux sont les équipements principaux de l'architecture DOMINO. Ils sont responsables du volume des informations échangées entre les différents participants, d'où l'importance de leur disponibilité. Tous les nœuds centraux possèdent une base de données d'activité dans laquelle on trouve des fichiers journaux, les vulnerabilités détectées et les alertes, des firewalls, un protocole d'échange de résumés (*DOMINO Summary Exchange Protocol*) et des mécanisme de réaction ;
- La communauté de satellite est un petit réseau qui implémente une version locale du

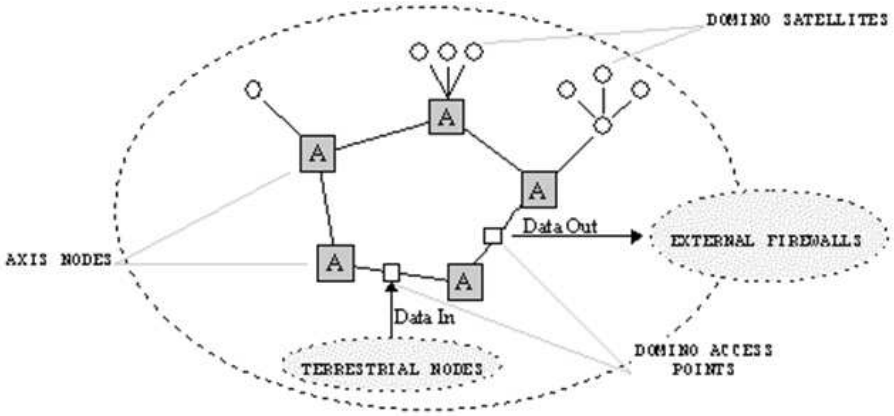

FIG. 5.6 – Architecture DOMINO [69]

protocole DOMINO ;

– Les contributeurs terrestres sont les sources d'informations les moins crédibles. Ils permettent d'avoir une quantité énorme de données. Ces nœuds n'implémentent pas le protocole DOMINO. Ils ont la possibilité de donner un rapport journalier des scans effectués sur les ports de données.

Les concepteurs de DOMINO veulent aussi développer un système de génération automatique de règles de firewall. L'inconvénient de DOMINO est la détection incertaine des attaques.

5.3 Conclusion

La problématique à laquelle tentent de répondre les concepteurs des IDS collaboratifs, est la collecte et la correlation des preuves dans le cadre d'attaques distribuées et coordonnées. Les approches que nous avons étudiées montrent que (1) chacune d'elles se basent sur différents types d'architecture et (2) différents algorithmes permettent la corrélation de ces informations. D'après cette étude, il est évident qu'aucune des approches n'est parfaite, elles ont toutes des avantages et des inconvénients. Ceci démontre la difficulté à l'heure actuelle pour le monde de la recherche de trouver une solution optimale pour répondre à cette problématique. Nous pouvons déduire que les résultats obtenus, après les implémentations des approches, confirment l'idée qu'une mise à l'échelle de l'Internet est très complexe pour ce genre de système. Toutefois, les chercheurs continuent à explorer les solutions fondées sur la combinaison de multiples IDS permettant de combattre la croissance des attaques sur l'Internet qui sont, en outre, de plus en plus sophistiquées.

Chapitre 6

Conclusion et travaux à venir

Dans le cadre du projet GIS 3SGS, nous étudions la possibilité d'utiliser une approche collaborative pour la détection d'attaques sur les réseaux pair à pair. Dans ce contexte, le premier travail consiste à effectuer l'état de l'art des réseaux pair à pair et de leur sécurité en terme de failles et solutions collaboratives pour la détection. Pour ce faire, à travers les différents chapitres de ce délivrable nous avons (1) présenté les différentes architectures de réseaux pair à pair et leurs propriétés ; (2) recensé, détaillé et comparé les différentes solutions de supervision pour ce type de réseau ; (3) mis en évidence les différentes failles de sécurité auxquelles les réseaux pair à pair sont sujets, et enfin (4) présenté les solutions collaboratives pour la détection d'intrusion dans les réseaux IP. Etant donné cet état de l'art, il apparaît que des solutions isolées existent et sont performantes pour la détection d'attaques isolées. Toutefois, dans un contexte d'attaques coordonnées, ciblant différentes parties de la topologie du réseau virtuel, une approche collaborative prend tout son sens et sera l'étude qui sera menée par la suite.

Préalablement, il est nécessaire de mieux comprendre quels sont les contenus qui sont surveillés ou attaqués et quel est le mode opératoire mis en oeuvre par les attaquants. Pour ce faire, notre travail actuel consiste, conformément à la description initiale du projet, à caractériser les comportements malveillants. Plus précisément, nous partons de l'hypothèse que les fichiers populaires et leurs mots clés associés sont les cibles privilégiées de ces attaques. Nous voulons établir une corrélation entre les fichiers populaires et la surveillance de ces derniers et déterminer le mode opératoire ainsi que la provenance de ces attaques. Les différents objectifs identifiés sont présentés ci dessous.

Corrélation entre le préfixe commun et la surveillance des mots clés : Cette corrélation a été mise en évidence par les travaux menés au LORIA. Nous souhaitons ici conforter ce résultat en prenant en considération l'ensemble des fichiers populaires et ainsi valider l'indicateur sur une quantité importante d'informations.

Corrélation entre la popularité des fichiers et les attaques : Nous faisons l'hypothèse que plus un fichier est populaire et plus ce dernier risque de subir des attaques ou d'être surveillé. Cette hypothèse devra être confirmée ou infirmée par le biais de mesures.

Identification des attaquants : Nous faisons l'hypothèse que quelques grandes compagnies sont responsables de la majorité des attaques sur la DHT Kad et on suppose que ces attaques ont pour but de protéger les droits d'auteur des produits. Nous souhaitons donc identifier les organisations responsables de la surveillance des fichiers sur le réseau.

Corrélation entre producteurs et fichiers attaqués : Nous souhaitons vérifier si les titres de fichiers issus de produits de certains producteurs sont particulièrement surveillés. Nous faisons l'hypothèse que certains producteurs souhaitant protéger leurs droits d'auteur surveillent ou font surveiller des fichiers sur la DHT Kad.

Corrélation entre la surveillance des mots clés et celle d'un fichier : Certains mots clés sont génériques. Ainsi, si l'un d'entre eux est surveillé, cela ne signifie pas pour autant que le fichier le soit. Nous souhaitons déterminer la probabilité qu'un fichier soit attaqué si ces mots clés le sont. Nous supposons alors que si un attaquant surveille un fichier, il va également observer tous les mots clés issus de ce fichier, et qu'ainsi la majorité des nœuds participant à l'attaque sur les mots clés d'un même fichier proviennent d'un même sous réseau.

L'ensemble de ces travaux sont en cours et seront présentés dans le délivrables suivants, portant de la caractérisation des comportements malveillants.

Bibliographie

[1] William Acosta and Surendar Ch. Trace driven analysis of the long term evolution of gnutella peer-to-peer traffic. In *In Proceedings of the Passive and Active Measurment Conference (PAM'07*, 2007.

[2] Frederic Aidouni, Matthieu Latapy, and Clemence Magnien. Ten weeks in the life of an edonkey server. *Parallel and Distributed Processing Symposium, International*, 0 :1–5, 2009.

[3] Oussama Allali, Matthieu Latapy, and Clemence Magnien. Measurement of edonkey activity with distributed honeypots. In *IPDPS '09 : Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*, pages 1–8, Washington, DC, USA, 2009. IEEE Computer Society.

[4] Kelsey Anderson. An analysis of the traffic on the gnutella network, March 2001.

[5] Rémi Badonnel, Radu State, Isabelle Chrisment, and Olivier Festor. A Management Platform for Tracking Cyber Predators in Peer-to-Peer Networks. In *The Second International Conference on Internet Monitoring and Protection - ICIMP 2007*, Santa Clara, CA/USA, 2007.

[6] Kevin Bauer, Damon McCoy, Dirk Grunwald, and Douglas Sicker. Bitstalker : Accurately and efficiently monitoring bittorrent traffic. In *Proceedings of the 1st IEEE Workshop on Information Forensics and Security*, London, United Kingdom, December 2009.

[7] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. *SIGOPS Oper. Syst. Rev.*, 36(SI) :299–314, 2002.

[8] Guillaume Champeau. The pirate bay abandonne son tracker au profit des liens magnet. `http://www.numerama.com/magazine/14507-the-pirate-bay-abandonne-son-tracker-au-profit-des-liens-magnet.html`, 2009.

[9] Guillaume Champeau. Le fournisseur d'accès de the pirate bay condamné à le débrancher. `http://www.numerama.com/magazine/15718-le-fournisseur-d-acces-de-the-pirate-bay-condamne-a-le-debrancher-maj.html`, 2010.

[10] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. Making gnutella-like p2p systems scalable. In *SIGCOMM '03 : Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 407–418, New York, NY, USA, 2003. ACM.

[11] Nicolas Christin, Andreas S. Weigend, and John Chuang. Content availability, pollution and poisoning in file sharing peer-to-peer networks. In *EC '05 : Proceedings of the 6th ACM conference on Electronic commerce*, pages 68–77, New York, NY, USA, 2005. ACM.

[12] Cristiano Costa and Jussara Almeida. Reputation systems for fighting pollution in peer-to-peer file sharing systems. In *P2P '07 : Proceedings of the Seventh IEEE International Conference on Peer-to-Peer Computing*, pages 53–60, Washington, DC, USA, 2007. IEEE Computer Society.

[13] Scott A. Crosby and Dan S. Wallach. An analysis of bittorrent ?s two kademlia-based dhts, 2007.

[14] John R. Douceur. The sybil attack. In *IPTPS '01 : Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260, London, UK, 2002. Springer-Verlag.

[15] D. Dumitriu, E. Knightly, A. Kuzmanovic, I. Stoica, and W. Zwaenepoel. Denial-of-service resilience in peer-to-peer file sharing systems. *SIGMETRICS Perform. Eval. Rev.*, 33(1) :38–49, 2005.

[16] Jarret Falkner, Michael Piatek, John P. John, Arvind Krishnamurthy, and Thomas Anderson. Profiling a million user dht. In *IMC '07 : Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 129–134, New York, NY, USA, 2007. ACM.

[17] The Gnutella Developer Forum. Gnutella 0.4 specifications. `http://rfc-gnutella.sourceforge.net/developer/stable/index.html`, 2000.

[18] The Gnutella Developer Forum. Gnutella 0.6 specifications. `http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html`, 2002.

[19] Astrid Girardeau. Mininova doit faire le maxi ménage. `http://ecrans.fr/Mininova-doit-faire-le-maxi-menage,7939.html`, 2009.

[20] Jean-Loup Guillaume, Matthieu Latapy, and Stevens Le Blond. Statistical analysis of a P2P query graph based on degrees and their time-evolution. In *6th International Workshop on Distributed Computing (IWDC 04)*, Kolkata India, 2004.

[21] Sidath B. Handurukande, Anne-Marie Kermarrec, Fabrice Le Fessant, Laurent Massoulié, and Simon Patarin. Peer sharing behaviour in the edonkey network, and implications for the design of server-less file sharing systems. In *EuroSys Conference (EuroSys'06)*, 2006.

[22] Oliver Heckmann, Axel Bock, Andreas Mauthe, and Ralf Steinmetz. The edonkey file-sharing network. In Peter Dadam and Manfred Reichert, editors, *GI Jahrestagung (2)*, volume 51 of *LNI*, pages 224–228. GI, 2004.

[23] John Ioannidis and Steven M. Bellovin. Implementing pushback : Router-based defense against ddos attacks. In *NDSS*. The Internet Society, 2002.

[24] M. Izal, Guillaume Urvoy-Keller, Ernst W. Biersack, P. A. Felber, A. Al Hamra, and L. Garcés-Erice. Dissecting bittorrent : Five months in a torrent's lifetime. In *Passive and Active Network Measurement*, pages 1–11. 2004.

[25] Thomas Karagiannis, Andre Broido, Michalis Faloutsos, and Kc claffy. Transport layer identification of p2p traffic. In *IMC '04 : Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 121–134, New York, NY, USA, 2004. ACM.

[26] Alexander Klemm, Christoph Lindemann, Mary K. Vernon, and Oliver P. Waldhorst. Characterizing the query behavior in peer-to-peer file sharing systems. In *IMC '04 : Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 55–67, New York, NY, USA, 2004. ACM.

[27] Michael Kohnen, Mike Leske, and Erwin P. Rathgeb. Conducting and optimizing eclipse attacks in the kad peer-to-peer network. In *NETWORKING '09 : Proceedings of the 8th International IFIP-TC 6 Networking Conference*, pages 104–116, Berlin, Heidelberg, 2009. Springer-Verlag.

[28] Marlom A. Konrath, Marinho P. Barcellos, and Rodrigo B. Mansilha. Attacking a swarm with a band of liars : evaluating the impact of attacks on bittorrent. In *P2P '07 : Proceedings of the Seventh IEEE International Conference on Peer-to-Peer Computing*, pages 37–44, Washington, DC, USA, 2007. IEEE Computer Society.

[29] Stefan S. Krishna, P. Krishna Gummadi, and Steven D. Gribble. A measurement study of peer-to-peer file sharing systems, 2002.

[30] Matthieu Latapy, Clémence Magnien, Guillaume Valadon, Lip Cnrs, Université Pierre, and Marie Curie. First report on database specification and access including content rating and fake detection system. http. In *antipaedo.lip6.fr/. 7 Clémence Magnien, Matthieu Latapy, Jean-Loup Guillaume, and Bénédicte Le Grand. First*, 2008.

[31] Stevens Le Blond, Jean-Loup Guillaume, and Matthieu Latapy. Clustering in P2P exchanges and consequences on performances. In *4th International Workshop on Peer-To-Peer Systems (IPTPS'05)*, Ithaca, NY États-Unis, 2005.

[32] Fabrice Le Fessant, Sidath B. Handurukande, Anne-Marie Kermarrec, and Laurent Massoulié. Clustering in peer-to-peer file sharing workloads. In *Peer-to-Peer Systems III, Third International Workshop (IPTPS'04), La Jolla, CA, USA, February 26-27, 2004, Revised Selected Papers*, Feb 2004.

[33] Hogyun Lee and Taekyong Nam. P2p honeypot to prevent illegal or harmful contents from spreading in p2p network. volume 1, pages 497 –501, feb. 2007.

[34] Uichin Lee, Min Choi, Junghoo Cho, M. Y. Sanadidi, and Mario Gerla. Understanding pollution dynamics in p2p file sharing. In *In Proceedings of the 5th International Workshop on Peer-toPeer Systems (IPTPS'06*, 2006.

[35] Nathaniel Leibowitz, Matei Ripeanu, and Adam Wierzbicki. Deconstructing the kazaa network. In *WIAPP '03 : Proceedings of the The Third IEEE Workshop on Internet Applications*, page 112, Washington, DC, USA, 2003. IEEE Computer Society.

[36] J. Liang, R. Kumar, and K. Ross. Understanding kazaa, 2004.

[37] Jian Liang, Rakesh Kumar, and et al. Pollution in p2p file sharing systems. In *IN IEEE INFOCOM*, pages 1174–1185, 2005.

[38] Jian Liang, Naoum Naoumov, and Keith W. Ross. Efficient blacklisting and pollution-level estimation in p2p file-sharing systems. In Kenjiro Cho and Philippe Jacquet, editors, *AINTEC*, volume 3837 of *Lecture Notes in Computer Science*, pages 1–21. Springer, 2005.

[39] Jian Liang, Naoum Naoumov, and Keith W. Ross. The index poisoning attack in p2p file sharing systems. In *INFOCOM*. IEEE, 2006.

[40] Jean loup Guillaume, Matthieu Latapy, Clémence Magnien, and Guillaume Valadon. Technical report on the content rating and fake detection system measurement and analysis of p2p activity against paedophile content project, 2009.

[41] M. Locasto M, J. Parekh, A. Keromytis, and S. Stolfo. Towards collaborative security and p2p intrusion detection. In *the 2005 IEEE workshop on information assurance and security*, pages 333–39, 2005.

[42] Ratul Mahajan, Steven M. Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker. Controlling high bandwidth aggregates in the network. *SIGCOMM Comput. Commun. Rev.*, 32(3) :62–73, 2002.

[43] Evangelos P. Markatos. Tracing a large-scale peer to peer system : An hour in the life of gnutella. *Cluster Computing and the Grid, IEEE International Symposium on*, 0 :65, 2002.

[44] Petar Maymounkov and David Mazières. Kademlia : A peer-to-peer information system based on the xor metric. In *IPTPS '01 : Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 53–65, London, UK, 2002. Springer-Verlag.

[45] Ghulam Memon. Characterizing traffic in widely-deployed dht, 2008.

[46] Ghulam Memon, Reza Rejaie, Yang Guo, and Daniel Stutzbach. Large-scale monitoring of DHT traffic. In *International Workshop on Peer-to-Peer Systems (IPTPS)*, Boston, MA, April 2009.

[47] Jelena Mirkovic. *D-WARD : source-end defense against distributed denial-of-service attacks*. PhD thesis, University of California, Los Angeles, August 2003.

[48] Jelena Mirkovic, Sven Dietrich, David Dittrich, and Peter Reiher. *Internet Denial of Service : Attack and Defense Mechanisms*. Prentice HALL-Professional Technical Refrence, 2005.

[49] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *SIGCOMM '01 : Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA, 2001. ACM.

[50] Matei Ripeanu, Adriana Iamnitchi, and Ian Foster. Mapping the gnutella network. *IEEE Internet Computing*, 6 :50–57, 2002.

[51] Antony Rowstron and Peter Druschel. Pastry : Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218 :329–? ?, 2001.

[52] Walid Saddi and Fabrice Guillemin. Measurement based modeling of edonkey peer-to-peer file sharing system. In *ITC20'07 : Proceedings of the 20th international teletraffic conference on Managing traffic performance in converged networks*, pages 974–985, Berlin, Heidelberg, 2007. Springer-Verlag.

[53] D. Schnackenberg, K. Djahandari, and D. Sterne. Infrastructure for intrusion detection and response. In *DARPA Information Survivability Conference and Exposition. DISCEX '00.*, volume 2, pages 3–11, Hilton Head, SC, USA, 2000.

[54] S. Sen and Jia Wang. Analyzing peer-to-peer traffic across large networks. *Networking, IEEE/ACM Transactions on*, 12(2) :219–232, April 2004.

[55] A. Singh, T. W. Ngan, P. Druschel, and D. S. Wallach. Eclipse attacks on overlay networks : Threats and defenses. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, 2006.

[56] Kunwadee Sripanidkulchai. The popularity of gnutella queries and its implications on scalability, 2001.

[57] Moritz Steiner, Ernst W Biersack, and Taoufik En-Najjary. Actively monitoring peers in kad. In *IPTPS'07, 6th International Workshop on Peer-to-Peer Systems, February 26-27, 2007, Bellevue, USA*, 02 2007.

[58] Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack. Exploiting kad : possible uses and misuses. *SIGCOMM Comput. Commun. Rev.*, 37(5) :65–70, 2007.

[59] Moritz Steiner, Taoufik En-Najjary, and Ernst W Biersack. A global view of kad. In *IMC 2007, ACM SIGCOMM Internet Measurement Conference, October 23-26, 2007, San Diego, USA*, 10 2007.

[60] Dan Sterne, Kelly Djahandari, Brett Wilson, Bill Babson, Dan Schnackenberg, Harley Holliday, and Travis Reid. *Recent Advances in Intrusion Detection*, volume 2212/2001, chapter Autonomic Response to Distributed Denial of Service Attacks, pages 134–149. SpringerLink, January 2001. http ://www.springerlink.com/content/l3wgy1xe26gccekb.

[61] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord : A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01 : Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, New York, NY, USA, 2001. ACM.

[62] TorrentFreak. Bittorrent's future ? dht, pex and magnet links explained. `http://torrentfreak.com/bittorrents-future-dht-pex-and-magnet-links-explained-091120/`, 2009.

[63] Kurt Tutschku. A measurement-based traffic profile of the edonkey filesharing service. In Chadi Barakat and Ian Pratt, editors, *PAM*, volume 3015 of *Lecture Notes in Computer Science*, pages 12–21. Springer, 2004.

[64] Uzi Tuvian and Lital Porat. Hash collision attack vectors on the ed2k p2p network, 2010.

[65] Paul Vixie, Gerry Neeringer, and Mark Schleifer. Report on the DDoS Attack on the DNS Root Servers : Events of 21-Oct-2002. Technical report, November 2002.

[66] Peng Wang, James Tyra, Eric Chan-Tin, Tyson Malchow, Denis Foo Kune, Nicholas Hopper, and Yongdae Kim. Attacking the kad network. In *SecureComm '08 : Proceedings of the 4th international conference on Security and privacy in communication netowrks*, pages 1–10, New York, NY, USA, 2008. ACM.

[67] Scott Wolchok and J. Alex Halderman. Crawling BitTorrent DHTs for fun and profit. In *Proc. 4th USENIX Workshop on Offensive Technologies (WOOT)*, August 2010.

[68] Jia Yang, Hao Ma, Weijia Song, Jian Cui, and Changling Zhou. Crawling the edonkey network. In *GCCW '06 : Proceedings of the Fifth International Conference on Grid and Cooperative Computing Workshops*, pages 133–136, Washington, DC, USA, 2006. IEEE Computer Society.

[69] V. Yegneswaran, P. Barford, and S. Jha. Global intrusion detection in the domino overlay system. In *In Proceedings of Network and Distributed System Security Symposium (NDSS*, 2004.

[70] Jie Yu, Chengfang Fang, Jia Xu, Ee-Chien Chang, and Zhoujun Li. Id repetition in kad. In Henning Schulzrinne, Karl Aberer, and Anwitaman Datta, editors, *Peer-to-Peer Computing*, pages 111–120. IEEE, 2009.