# Supervision des réseaux pair à pair structurés appliquée à la sécurité des contenus

Thibault Cholez

Nancy-Université
Université Henri Poincaré

**Département de Formation Doctorale en Informatique** **École doctorale IAEM Lorraine**
**UFR Faculté des Sciences et Technologies**

# Supervision des réseaux pair à pair structurés appliquée à la sécurité des contenus

## THÈSE

présentée et soutenue publiquement le 23 juin 2011

pour l'obtention du

**Doctorat de l'université Henri Poincaré – Nancy 1**

**(spécialité informatique)**

par

Thibault Cholez

**Composition du jury**

| | | |
|---|---|---|
| *Président :* | Claude Godart | Professeur à l'ESSTIN, Nancy Université |
| *Rapporteurs :* | Matthieu Latapy | Directeur de recherche CNRS au LIP6 |
| | Ludovic Mé | Professeur à Supélec-Rennes |
| *Examinateurs :* | Ernst Biersack | Professeur à EURECOM |
| | Isabelle Chrisment | Professeur à l'ESIAL, Nancy Université |
| | Olivier Festor | Directeur de recherche à l'INRIA Nancy - Grand Est |
| *Invité :* | Guillaume Doyen | Maître de conférences à l'UTT |

**Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503**

Loria
Laboratoire lorrain de recherche en informatique et ses applications

Mis en page avec la classe thloria.

## Remerciements

Cette thèse est le fruit d'un long travail dont l'aboutissement doit beaucoup à l'aide et au soutien de quelques personnes que je tiens à remercier avant tout.

Je tiens tout d'abord à remercier Matthieu Latapy et Ludovic Mé pour le temps qu'ils ont consacré à la relecture de ce manuscrit, ainsi que l'ensemble des membres du jury pour l'intérêt qu'ils portent à mes travaux.

Ensuite, j'adresse mes plus chaleureux remerciements à Isabelle Chrisment, ma directrice de thèse, pour ses précieux conseils, sa gentillesse et la confiance qu'elle m'a accordée durant toutes ces années. Je remercie tout autant Olivier Festor, directeur de l'équipe projet INRIA MADYNES, pour m'avoir accueilli au sein de l'équipe et pour m'avoir également accompagné tout au long de la thèse. Le temps qu'ils m'ont consacré et l'attention bienveillante permanente qu'ils m'ont portée ont été les meilleurs des encouragements. Les nombreuses discussions que nous avons pu avoir m'ont grandement enrichi, scientifiquement, professionnellement et humainement. Je remercie également Guillaume Doyen pour m'avoir accueilli à l'UTT et donné l'opportunité d'approfondir mes travaux de recherche.

J'adresse un remerciement particulier à plusieurs personnes avec qui j'ai eu la chance de travailler, stagiaires et ingénieurs, dont le sérieux et la qualité du travail ont permis, outre l'obtention de résultats, un enrichissement mutuel. Je remercie ainsi pour leur précieuse collaboration Frédéric Beck, Juan Pablo Timpanaro, Christopher Hénard, Andreea Orosanu et Guillaume Montassier. Je remercie également Raphaël Manfredi pour les discussions enrichissantes échangées concernant l'implantation d'une protection pour le client gtk-gnutella.

Je n'oublie pas l'ensemble de mes collègues de l'équipe MADYNES au sein de laquelle l'ambiance de travail est très agréable et propice aux échanges aussi bien professionnels que personnels, ainsi que mes amis du LORIA pour leur bonne humeur quotidienne.

Ces travaux ont été financés en partie par le projet MAPE (Measurement and Analysis of Peer-to-peer Exchanges for pedocriminality fighting and traffic profiling) de l'Agence Nationale de la Recherche, sous le contrat ANR-07-TLCOM-24, par la région Lorraine, ainsi que par le projet GIS 3SGS ACDAP2P (Surveillance, Sûreté et Sécurité des Grands Systèmes : Approche collaborative pour la détection d'attaques dans les réseaux P2P). Je tiens particulièrement à remercier le conseil régional de Lorraine pour avoir soutenu mes travaux et plus généralement pour son engagement en faveur de la recherche. Je remercie également le LORIA et l'INRIA Nancy Grand-Est pour les excellentes conditions de travail offertes aux chercheurs.

Plus personnellement, je ne saurai assez remercier l'ensemble de ma famille et plus particulièrement mes parents pour leur patience et leur soutien quotidien. Votre sollicitude est la chose la plus précieuse.

A toutes et à tous, merci.

# Table des matières

# Introduction générale

## 1 Contexte

Cette thèse porte sur l'étude des **réseaux pair à pair**, appelés également « réseaux poste à poste [1] », mais plus communément nommés par l'anglicisme « réseaux peer to peer », dont nous utiliserons l'abréviation « **réseaux P2P** ». Les réseaux P2P sont un type de **réseau informatique** dont la particularité est de permettre la découverte et l'échange direct de **services** entre les **machines terminales**, qui jouent à la fois le rôle de client et de serveur.

Nous présentons dans ce manuscrit de nouvelles méthodes de supervision capables d'appréhender les problèmes de sécurité affectant les informations partagées aux sein des réseaux pair à pair structurés. En effet, les réseaux P2P étant pour la plupart publics et faiblement sécurisés, ceux-ci souffrent de nombreuses attaques affectant la fiabilité de leurs services. La nature complètement distribuée et dynamique des réseaux P2P, leur taille importante, et les attaques dont ils sont victimes en font des systèmes particulièrement difficiles à superviser et à sécuriser efficacement.

Cette section présente tout d'abord le contexte et les problématiques de cette thèse en énonçant les principes généraux des réseaux P2P, la terminologie associée et les caractéristiques ayant permis leur essor. Nous présenterons ensuite les défis liés à l'émergence de ce nouveau paradigme de communication puis, plus particulièrement, les problèmes traités dans ce manuscrit.

### 1.1 Principes généraux du pair-à-pair

#### 1.1.1 Un réseau d'applications

Les réseaux pair à pair sont un type de réseau informatique, c'est à dire un ensemble interconnecté d'ordinateurs autonomes capables d'échanger des informations. Chaque réseau P2P respecte ainsi un protocole de communication qui lui est propre et qui définit la syntaxe et la sémantique des données échangées entre les nœuds, ainsi que le comportement attendu suite à l'envoi ou à la réception de ces données. Les réseaux P2P sont cependant tributaires d'autres protocoles pour fonctionner car ceux-ci sont construits pour la plupart au dessus du réseau Internet (c'est pourquoi on parle parfois de « réseau d'overlay »). Les réseaux P2P reposent ainsi sur le protocole IP « Internet Protocol » pour router et acheminer les données vers les machines terminales connectées à l'Internet, mais également sur les protocoles TCP « Transmission Control Protocol » et UDP « User Datagram Protocol » pour contrôler et permettre la délivrance des messages à l'application pair à pair. Nous verrons par la suite que certains réseaux P2P mettent également en œuvre des algorithmes de routage et d'adressage complexes à l'instar de ceux utilisés par le réseau IP, ce qui termine d'affirmer leur appartenance au domaine des réseaux. Cependant,

1. poste-à-poste est la traduction française officielle de l'anglais peer-to-peer dont pair-à-pair est cependant un synonyme valide [gdtedn06] [qdllf06], les traits d'union étant omis lors d'une utilisation en tant qu'adjectif.

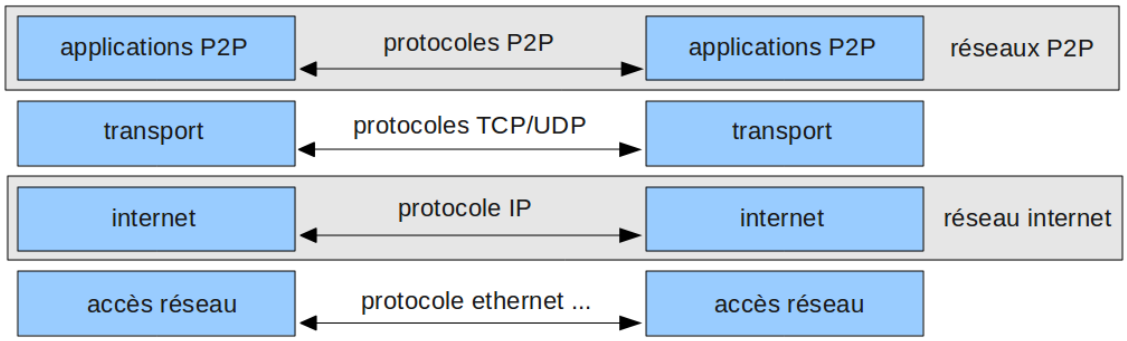


FIGURE 1 – Le pair-à-pair dans le modèle en couche TCP/IP

pour l'utilisateur final, le réseau pair à pair apparaît comme un système distribué dans le sens où l'existence de la multitude d'ordinateurs autonomes sous-jacents lui est transparente, celle-ci étant masquée par l'application pair à pair qui fournit les services.

La figure 1 permet de situer la place du pair-à-pair dans la pile des protocoles de l'Internet. Le pair-à-pair se situe au niveau de la couche application du modèle TCP/IP. Outre les informations nécessaires à l'exécution de leurs services, les applications P2P doivent prendre en charge la présentation des données, notamment lorsque les clients sont exécutés sur différentes architectures matérielles. Il convient aussi de noter qu'il n'existe pas un seul protocole P2P mais une multitude, autant qu'il existe de réseaux P2P différents. La majorité des applications, ou clients P2P, est conçue pour un réseau P2P spécifique et ne prend donc en charge qu'un seul protocole. Quelques clients P2P sont cependant multi-protocoles et peuvent rejoindre plusieurs réseaux P2P et ainsi accéder à leurs services.

### 1.1.2 Une architecture de services

Nous avons vu que, bien que situés au niveau applicatif, les réseaux P2P peuvent être considérés comme des réseaux de communication à part entière. Cependant, et contrairement au réseau IP, leur place dans la hiérarchie des protocoles fait que les réseaux P2P ne sont pas conçus pour fournir un support à une couche supérieure qui implanterait une multitude de services mais pour permettre l'exécution d'un service particulier (par exemple le partage de fichiers) qui est la raison même du réseau. Cette caractéristique des systèmes P2P, qui rend le réseau P2P difficilement dissociable du service qu'il fournit, fait qu'ils sont assimilés à une architecture de services. Pour bien définir les systèmes pair à pair, il faut donc considérer ces deux composantes importantes et indissociables : il s'agit de réseaux de communication construits pour assurer des services spécifiques sur le modèle pair à pair. Ils sont ainsi naturellement comparés à l'architecture fournissant la majorité des services sur l'Internet à savoir le modèle client-serveur. Dans ce modèle, une entité centrale puissante, le serveur, est responsable de fournir les services aux clients qui les consomment.

Dans le modèle pair à pair, chaque client est à la fois consommateur et fournisseur de services. La suppression du serveur central rend le système distribué et confère aux systèmes P2P des propriétés intéressantes. Leur principal intérêt est de pouvoir fournir des services avec une forte disponibilité et ce à un très faible coût. Pour cela, le réseau pair à pair mutualise les ressources (bande passante, capacité de stockage, puissance de calcul) des nombreux pairs connectés au

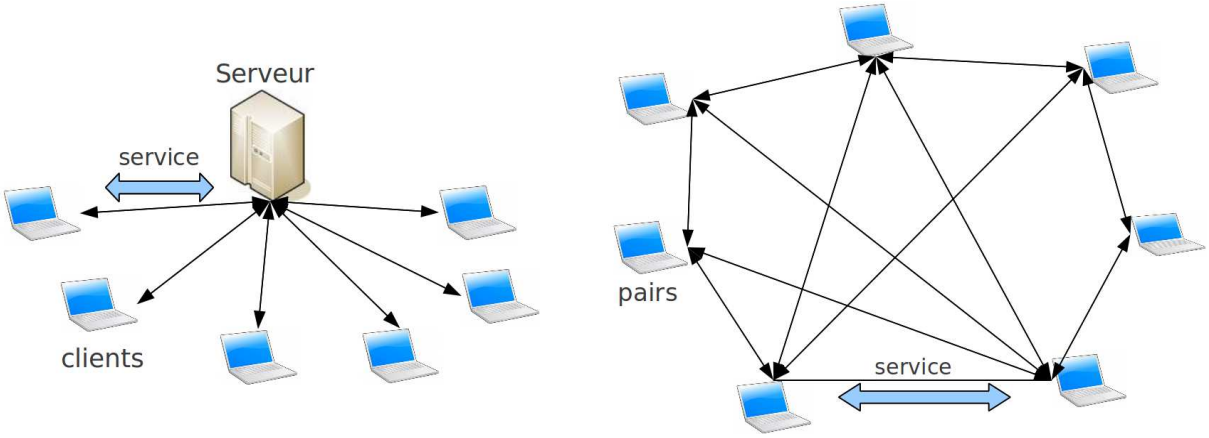


FIGURE 2 – Comparaison des modèles client-serveur et P2P

| Propriété | Modèle client-serveur | Modèle pair à pair |
|---|---|---|
| coût de l'infrastructure | élevé | faible |
| tolérance aux pannes | faible | élevée |
| contribution des machines terminales | non | oui |
| répartition de la charge | non | oui |
| passage à l'échelle | limité / coûteux | naturel |
| administration / contrôle | élevé | faible |
| gestion de la qualité de service | élevée | faible |
| sécurité | élevée | faible |

TABLE 1 – Caractéristiques des deux architectures de services

réseau, chacun étant actif et contribuant à la réalisation des services rendus par le système P2P. Ces pairs étant des machines terminales d'utilisateurs, leur fonctionnement n'est pas fiable. Les réseaux P2P sont donc très dynamiques, le va-et-vient constant des pairs dans le réseau (appelé « churn ») changeant en permanence l'état des liens. Les systèmes P2P atteignent cependant une bonne qualité de services par la mise en œuvre de mécanismes de réplication palliant la défaillance ou le mauvais comportement des pairs. La figure 2 rappelle la différence d'établissement des services entre les paradigmes pair à pair et client-serveur. Le tableau 1 synthétise les forces et les faiblesses de ces deux architectures. Les principales faiblesses des réseaux P2P viennent des difficultés de gestion de tels réseaux publics et massivement distribués. Il est en effet difficile d'assurer la qualité et la sécurité des services P2P alors que ces réseaux sont constitués en partie de pairs malveillants. Cette thèse contribue à l'amélioration de la gestion de réseaux P2P à travers une meilleure supervision de ceux-ci.

## 1.2 Terminologie

Pour éviter toute confusion dans la suite de ce manuscrit, nous définissons la terminologie propre aux réseaux pair à pair, à savoir :

- les termes d'**application P2P** et de **client P2P** désignent un logiciel informatique fournissant à l'utilisateur un service selon le paradigme pair à pair ;

- le terme **utilisateur** désigne un individu utilisant une application P2P ;
- les termes de **nœud** et de **pair** désignent une instance d'un client P2P sur une machine connectée à l'internet ;
- le terme d'**architecture P2P**, ou topologie logique, désigne le modèle organisationnel définissant les liens de communication entre les pairs ;
- le terme de **réseau P2P** désigne l'ensemble des nœuds communicants par un même protocole P2P et interconnectés à un moment donné ;
- le terme de **système P2P** désigne un ensemble constitué des machines d'un réseau P2P, de leurs ressources partagées et des services fournis par ce dernier ;

### 1.3 Essor des réseaux pair à pair

Les réseaux P2P sont apparus au début des années 2000 avec Napster qui fut créé en 1999 par Shawn Fanning et considéré comme étant le premier réseau P2P. Napster fournissait alors un service de partage de fichiers entre les pairs et, en cela, reste encore caractéristique de la majorité des réseaux P2P actuels qui continuent d'offrir ce service. La principale faiblesse de Napster fut de conserver un serveur unique pour indexer les fichiers partagés, et dont seul le transfert était distribué entre les pairs. Ce composant central fut fatal au réseau car il a suffi d'une décision de justice ordonnant la fermeture du serveur pour infraction à la législation sur le droit d'auteur pour mettre fin au système.

Cette faiblesse fut corrigée par la seconde génération de réseaux P2P qui vit émerger de nombreuses architectures P2P, toutes présentant des avantages et inconvénients mais ayant le point commun d'éviter tout composant central : soit en utilisant un réseau de serveurs autonomes (eDonkey, DirectConnect) ; soit en distribuant intégralement l'indexation des fichiers sur les pairs, d'abord de manière non structurée (Gnutella, FastTrack), puis structurée (Overnet, KAD, FreeNet) grâce aux apports de la communauté scientifique ayant conçu de nombreux algorithmes pour mettre en œuvre des tables de hachage distribuées (Chord, CAN, Pastry, Kademlia). Enfin, BitTorrent proposa une architecture hybride, reposant sur des serveurs Web pour l'indexation des fichiers et créant pour la distribution de chaque fichier un réseau de pairs spécifique. Ces nouvelles architectures P2P se sont rapidement développées et les applications P2P de partage de fichiers ont gagné en popularité jusqu'à générer plus de la moitié du trafic global de l'Internet [Ipo09].

Aujourd'hui, les réseaux P2P ne se limitent plus au seul partage de fichiers mais fournissent de nombreux autres services parmi lesquels : des réseaux de partage de fichiers anonymes (FreeNet, I2P GNUNet, ANts P2P), des systèmes de fichiers distribués (Tahoe-LAFS), des applications de voix sur IP (Skype, iGlance), d'annuaire distribué (P2PSIP), de messagerie instantanée (XMPP), de streaming de données vidéo (Joost, SopCast, PeerCast, PPLive) ou audio (Spotify). Comme le partage de fichiers (BitTorrent, eMule, Gnutella) constitue encore l'application majoritaire, le pair-à-pair est souvent confondu avec cette seule application.

## 2 Problématiques

L'importance que revêtent les réseaux P2P dans l'Internet d'aujourd'hui a mis en évidence plusieurs questions d'ordre éthique, mais surtout juridique et technique. Si les questions d'ordre éthique, comme la neutralité du réseau Internet par rapport aux différents protocoles existants, dépassent le cadre de cette thèse, nous présenterons brièvement les nouveaux défis juridiques et techniques liés au pair-à-pair puis décrirons plus particulièrement les problèmes traités par cette thèse dans la seconde partie de cette section.

## 2.1 Défis majeurs

### Questions juridiques

Les réseaux P2P sont principalement utilisés pour le partage de fichiers. Or, l'absence de serveur central pouvant contrôler les échanges et l'autonomie des pairs font que de nombreux contenus illégaux sont partagés grâce aux applications P2P. On peut classer ces contenus en trois catégories : la grande majorité des contenus échangés sont des contenus multimédia et des logiciels soumis aux droits d'auteurs, d'autres posent des problèmes de sécurité (virus, logiciels espions) et enfin les derniers concernent la protection de la personne (contenus pédophiles). Devant la difficulté de stopper des réseaux P2P qui sont désormais complètement distribués, des lois [Leg06] pénalisent directement le développement et la diffusion de la technologie P2P. Cependant, la technologie en soi est neutre et seule son utilisation pour partager un contenu particulier saurait être répréhensible.

Les forces de l'ordre ou des entreprises mandataires peuvent être amenées à collecter les adresses IP des pairs soupçonnés d'avoir commis des infractions. Il est cependant très difficile d'identifier précisément les pairs (et à fortiori les utilisateurs) en supervisant les réseaux P2P. Les causes d'erreurs sont multiples tant au niveau du réseau P2P (pollution des mécanismes d'indexation, injection de faux pairs), qu'au niveau du réseau IP (machine terminale corrompue, réseau WiFi insuffisamment sécurisé). Les investigations sont donc compliquées et doivent être menées avec la plus grande rigueur pour éviter les faux positifs qui sont dommageables tant pour la justice que pour les personnes accusées à tort. En particulier, la pollution des réseaux P2P fait qu'un contenu peut être téléchargé de manière non intentionnelle par un utilisateur. Ainsi, un internaute pensant télécharger en toute légalité un logiciel, mais téléchargeant en réalité un fichier interdit (car portant atteinte à la propriété intellectuelle des auteurs ou présentant un caractère pédophile), du fait par exemple d'un nom trompeur dû à la pollution des mécanismes d'indexation, ne pourra donc être condamné en l'absence d'élément intentionnel permettant de caractériser le délit et alors même qu'il aura été observé commettant une infraction. La pollution de l'indexation sur les réseaux P2P ne permet donc pas de s'assurer de l'intention réelle de l'auteur d'un téléchargement, et peut donc contribuer à troubler la tranquillité d'honnêtes citoyens. Les solutions de supervision proposées dans cette thèse visent à résoudre ce problème par la collecte d'informations plus complètes permettant de vérifier l'intention du pair.

### Questions techniques / scientifiques

Au delà des questions juridiques, les réseaux P2P restent une technologie récente et encore sujette à la résolution de nombreux défis scientifiques. En premier lieu, des défis concernant la sécurité des réseaux. En effet, le comportement autonome des pairs et l'absence de contrôle centralisé font que les réseaux P2P sont vulnérables à de nombreuses attaques (attaque Sybil, pollution, etc.) pouvant grandement affecter la sécurité et la qualité de service délivrées par les systèmes P2P. La pollution consiste à partager un contenu erroné ou dont le nom ne correspond pas au contenu, ce qui pose des problèmes de sécurité (diffusion de contenus illégaux, virus). L'attaque Sybil quant à elle consiste, pour un même attaquant physique, à insérer un grand nombre de «faux» pairs dans le réseau afin d'en altérer le routage des messages ou le stockage des données. Une fois insérés, ces nœuds peuvent réaliser plusieurs actions malveillantes (surveillance des échanges, pollution, suppression d'information, déni de service distribué, etc.). Détecter et limiter les effets de tels comportements de manière totalement distribuée restent encore aujourd'hui des défis majeurs auxquels nous nous attachons dans cette thèse. Outre les attaques, le comportement égoïste de nombreux pairs ne partageant pas leurs ressources nuit également

aux systèmes P2P et nécessite l'élaboration de mécanismes incitatifs.

De nouvelles architectures P2P sont encore conçues pour améliorer la sécurité des réseaux P2P, en améliorer les performances, ou pour fournir de nouvelles propriétés. Ainsi, une meilleure prise en compte de la topologie du réseau Internet sous-jacent, lors de l'établissement des communications entre pairs, peut améliorer les performances tout en diminuant le trafic entre AS (Systèmes Autonomes). Une autre propriété souhaitée est de pouvoir garantir l'anonymat des communications tout en limitant l'impact sur les performances. Enfin, les réseaux P2P existants sont en soi un important sujet d'étude. De nombreuses mesures sont réalisées pour mieux comprendre leur étendue, leur fonctionnement et leur dynamique, ceci, afin de les améliorer si possible, ou de concevoir de nouvelles architectures palliant les faiblesses constatées.

Nous avons présenté, sans être exhaustif, les principaux défis liés à l'émergence des réseaux P2P. Tous ces problèmes sont cependant intimement liés. Par exemple, la fermeture de Napster pour des raisons juridiques a favorisé l'émergence de réseaux P2P décentralisés, tout comme les investigations menées sur ces mêmes réseaux encouragent l'émergence de réseaux P2P anonymes.

## 2.2 Problèmes traités

L'objectif de cette thèse est de concevoir et d'appliquer de nouvelles méthodes de supervision capables d'appréhender les problèmes de sécurité affectant les données stockées dans les réseaux P2P structurés. Comme décrits précédemment, les problèmes de sécurité concernant les contenus sont de deux types, d'une part les réseaux P2P sont utilisés pour diffuser des contenus illégaux dont il est difficile d'identifier précisément les accès, et d'autre part des contenus légitimes peuvent disparaître ou être pollués si le mécanisme d'indexation du réseau P2P est corrompu (attaque Sybil). Les solutions de supervision mises en œuvre doivent ainsi permettre de détecter les comportements des pairs malveillants afin d'en limiter les effets. Nous avons choisi d'utiliser le réseau P2P KAD comme cas d'étude pour y appliquer les méthodes de supervision développées, car il s'agit d'un des réseaux P2P structurés les plus développés à ce jour et ne repose sur aucun composant centralisé.

Le premier enjeu est donc d'être capable de superviser précisément l'activité des pairs pour un ensemble de contenus critiques préalablement identifiés, en collectant les messages émis pour ces contenus tout en assurant les propriétés suivantes :

- collecter l'ensemble des requêtes relatives aux contenus ;
- ne pas annoncer ou partager de fichiers illégaux ;
- limiter les faux-positifs ;
- ne pas affecter les contenus non concernés ;
- utiliser peu de ressources.

L'architecture de supervision proposée doit permettre de mieux comprendre l'activité des pairs pour les contenus étudiés et fournir également un moyen efficace de constater précisément l'accès des pairs à ces contenus en vérifiant leur intentionnalité. Son application à l'étude et à la quantification des contenus pédophiles partagés dans KAD permet en outre d'apporter de nouvelles connaissances dans ce domaine spécifique.

Le second enjeu est de pourvoir superviser les attaques réalisées contre le mécanisme d'indexation du réseau P2P, qui est mis en œuvre par une table de hachage distribuée (DHT). Il s'agit de détecter aussi bien le vecteur de l'attaque (attaque Sybil) que les effets sur les contenus (pollution). L'objectif ici est double : (1) obtenir une vision globale des attaques affectant la DHT afin d'en estimer l'ampleur, et (2) pouvoir éviter les attaquants en fournissant des méthodes de détection automatiques et des mesures de protection aux clients P2P. La détection des attaques

par les clients est également soumise à des contraintes. Les mécanismes doivent :
- être automatiques (non intervention de l'utilisateur) ;
- s'exécuter en temps réel ;
- être compatibles avec les anciens clients ;
- être sujets à un faible taux d'erreur (faux positifs et négatifs) ;
- limiter le surcoût de messages.

# 3 Organisation des contributions

Dans le cadre de cette thèse, l'application de nouvelles méthodes de supervision à la sécurité des contenus partagés permet d'une part, d'identifier précisément les pairs accédant à des contenus spécifiques, d'autre part, de détecter les pairs réalisant des attaques et d'en protéger le réseau. Ces deux contributions constituent les parties principales de cette thèse présentées dans les parties II et III, et sont précédées par un état de l'art en partie I. Nous terminons ce manuscrit par une conclusion générale.

### Première partie : État de l'art

Dans le chapitre 1, nous présentons les différentes architectures P2P et motivons notre choix pour l'étude des réseaux pair à pair structurés. Nous présentons ensuite les différentes architectures de supervision existantes, chacune étant intimement liée à l'architecture du réseau P2P observé, ainsi que les mesures obtenues par chaque architecture.

Le chapitre 2 s'intéresse aux travaux relatifs aux problèmes de sécurité affectant les réseaux P2P, d'abord en considérant leurs contenus (contenus illégaux, pollution) puis, en considérant l'attaque Sybil dont nous étudions par ailleurs les diverses contre-mesures proposées tout en constatant leur difficile mise en pratique.

Enfin, le chapitre 3 présente le réseau P2P KAD qui sert de support à la fois pour la conception et pour la mise en œuvre de nos contributions. Nous décrivons ainsi en détail le fonctionnement de KAD ainsi que les expériences déjà réalisées sur ce réseau.

### Deuxième partie : Supervision des contenus et contrôle de leur accès

Afin d'étudier l'accès aux contenus indexés par la DHT, notre première intuition fut d'injecter de nombreux faux pairs dans le réseau afin d'y attirer les messages. Cependant, les principaux clients de KAD ont mis en œuvre des mécanismes de protection contre de telles attaques. Ces protections pragmatiques sont différentes de celles présentées dans la littérature et ne sont ni décrites ni évaluées. Nous les décrivons et les évaluons dans le chapitre 4 et montrons ainsi que, bien que constituant une réelle amélioration, ces protections demeurent insuffisantes car la DHT de KAD reste vulnérable à l'injection de faux pairs.

Le chapitre 5 présente notre architecture de supervision, HAMACK, capable de superviser précisément et de contrôler l'accès aux contenus dans le réseau P2P KAD en capturant l'ensemble des requêtes d'un pair depuis la recherche d'un mot-clé. Nous décrivons ainsi les principes de la DHT permettant la mise en œuvre de notre supervision. Nous présentons ensuite les fonctionnalités de l'architecture avant d'en évaluer l'efficacité.

Dans le chapitre 6, nous l'appliquons à la supervision passive des contenus pédophiles dans KAD. Sans aller jusqu'au contrôle d'accès, le but est de récolter des données permettant une quantification des contenus pédophiles dans le réseau. Nous décrivons ainsi l'environnement de collecte des données puis analysons celles-ci.

### Troisième partie : supervision des attaques contre la DHT et élaboration de mécanismes de protection

Nous nous intéressons dans le chapitre 7 à la détection d'attaques sur le réseau P2P KAD, en présentant tout d'abord une nouvelle forme de pollution particulièrement dangereuse affectant le réseau P2P. Nous réalisons une détection automatique de cette pollution au début du téléchargement et estimons la proportion de contenus pollués au sein du réseau. Nous proposons une modification du protocole KAD permettant une détection plus efficace mais restant vulnérable aux attaques ciblées sur la DHT. Nous étudions dans un second temps, pour l'ensemble de la DHT, les positionnement anormaux des pairs qui traduisent ces attaques. Nous utilisons pour cela deux méthodes, une basée sur l'analyse de la distance entre les pairs et l'autre sur la proximité entre les pairs et les données indexées. Nous concluons sur la présence de nombreux pairs suspects sur le réseau KAD.

Nous développons finalement, dans le chapitre 8, un moyen de détecter efficacement les attaques de la DHT en analysant la distribution des identifiants des pairs. Après avoir analysé des distributions saines de pairs, nous présentons un modèle de détection que nous évaluons ensuite. En cas de détection, des contre-mesures permettent de protéger un client des pairs malveillants. Nous terminons par l'évaluation de ces contre-mesures et leur application dans un client du réseau KAD.

# Première partie

# État de l'art

# Chapitre 1

# Architectures et supervision des réseaux pair à pair

**Sommaire**



## Introduction

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 différencions les architectures en deux grandes familles : celles ayant recours à des serveurs et celles complètement distribuées. Après avoir décrit les différentes approches et comparé leurs propriétés, nous nous intéressons en particulier aux réseaux P2P structurés.

Plusieurs campagnes de mesures ont été réalisées afin de comprendre le fonctionnement et la dynamique des réseaux P2P déployés. Chaque réseau P2P ayant une architecture propre et les réseaux n'étant pas interopérables, les méthodes d'observation développées sont intimement liées au réseau cible. De même, les données observables dépendent à la fois du réseau cible et de l'architecture de supervision mise en œuvre. Nous distinguons cinq architectures de supervision pour les réseaux P2P, à savoir : l'observation du trafic P2P, les mesures réalisées sur serveurs, l'utilisation de pots de miel, l'utilisation d'explorateurs et enfin l'injection de sondes dans le réseau. Nous décrivons et comparons ces approches dans la seconde moitié de ce chapitre.

## 1.1 Architectures pair à pair

### 1.1.1 Réseaux pair à pair avec serveur

Les architectures ayant recours à des serveurs sont les plus simples et furent parmi les premières développées. Elles peuvent être classées en trois catégories selon le type de serveurs util-

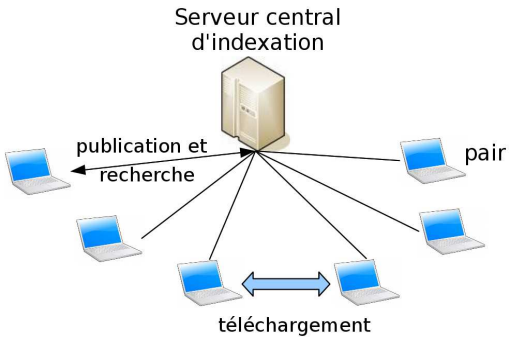


FIGURE 1.1 – Architecture P2P avec serveur central

isés : centralisés (e.g. Napster), distribués (e.g. eDonkey), ou différenciés (e.g. 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.

#### 1.1.1.1 Serveur central

Napster utilisait ainsi un unique serveur (en réalité un cluster de serveurs dédiés) [KGG02] responsable d'indexer les fichiers partagés par les pairs. L'architecture de Napster est présentée dans le schéma 1.1. Les pairs annoncent les fichiers qu'ils partagent sur le serveur central d'indexation qui recense ainsi l'ensemble des fichiers partagés par les pairs. Les pairs effectuent des recherches sur le serveur permettant d'obtenir la liste des fichiers souhaités et leur source. Le téléchargement d'un fichier se fait en établissant une connexion directe avec le pair fournisseur. L'arrêt du serveur en 2001 a naturellement entraîné l'arrêt complet du réseau. Le point critique pour Napster fut donc la disponibilité du serveur.

#### 1.1.1.2 Serveurs distribués

Le réseau eDonkey [2] [HBMS04] [Tut04] utilise un ensemble de serveurs distribués et opérés par des entités différentes. L'architecture d'eDonkey est présentée dans le schéma 1.2. Chaque serveur a individuellement 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 serveur 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

2. également appelé eDonkey 2000 ou ed2k

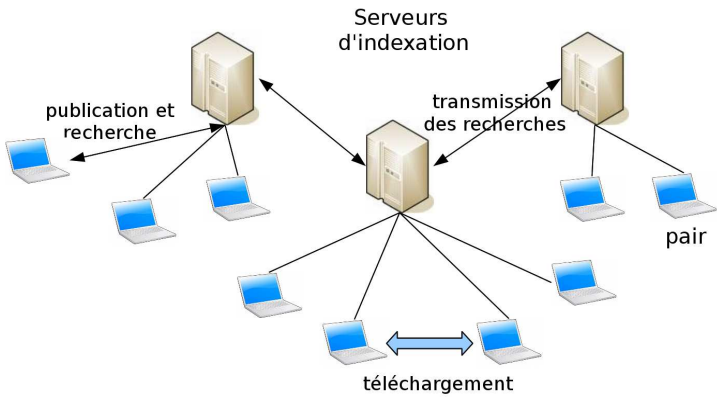


FIGURE 1.2 – Architecture P2P avec serveurs distribués

recherche des sources pour ce fichier est distribuée sur les serveurs. Les sources peuvent également être partagées entre deux pairs intéressés par un même contenu. Le téléchargement a finalement lieu en pair-à-pair.

En distribuant les serveurs, la disponibilité du réseau eDonkey semble à 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 No1 » et « DonkeyServer No2 » ont été fermés pour des raisons légales [Cha08]. A son apogée, Razorback2 [3] atteignait ainsi plus d'un million de pairs connectés. Les ressources matérielles pour opérer le serveur étaient alors conséquentes : outre une machine très puissante, le serveur nécessitait une connexion à Internet de 50Mbps, ce qui pose la question du financement de tels serveurs. 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 en plus un réel problème de confiance envers les serveurs inconnus.

#### 1.1.1.3 Serveurs différenciés

Le réseau P2P Bittorrent a la particularité d'avoir une architecture hybride, reposant sur deux types de serveurs distincts préalables aux échanges pair à pair tel que présenté par [LUKM05] et illustré par le schéma 1.3. Bittorrent repose tout d'abord sur des serveurs web pour indexer les contenus partagés. Les serveurs web permettent d'obtenir la liste des fichiers correspondant aux mots-clés souhaités ainsi que pour chacun d'eux un fichier de méta-données appellé fichier « torrent ». Ce fichier de méta-données comporte plusieurs informations, d'une part des informations intrinsèques au(x) fichier(s) à diffuser (taille, somme de contrôle, etc.) et d'autre part l'adresse d'un serveur de suivi appelé « tracker ». Le « tracker » est un second type de serveur dont le rôle est, pour un fichier donné, de connaître et de maintenir à jour la liste des pairs le partageant, ces pairs constituent l'essaim ou « swarm ». Chaque pair partageant un fichier géré par le serveur de suivi lui annonce sa présence et un pair souhaitant le télécharger lui demande la liste des sources potentielles. Finalement, l'échange de données se fait de pair à pair.

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 [Gir09]), ou sont bloqués par les

3. www.razorback2.com

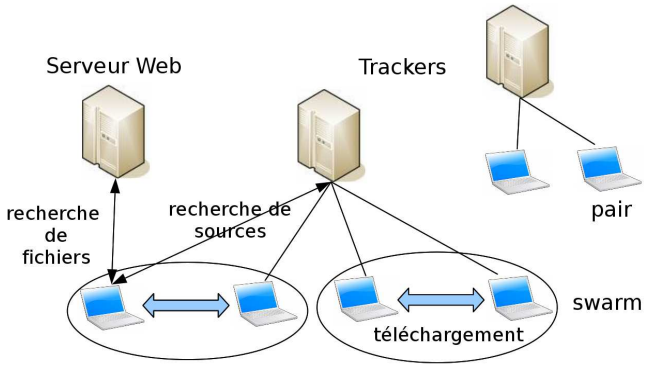


FIGURE 1.3 – Architecture P2P avec serveurs différenciés

fournisseurs d'accès à Internet (FAI) dans certains pays (The Pirate Bay[Cha10]), 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 » [Cha09b] est d'ores et déjà offerte par deux DHT basées sur Kademlia à savoir Vuze DHT [4] 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 [5]) 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 de méta-données. Ces informations sont accessibles dans la DHT à partir du hash du fichier de méta-données, appelé également « Magnet link » [Tor09]. Si les serveurs web n'ont plus la charge de fournir les fichiers de méta-données, l'indexation des liens par rapport aux mots-clés des contenus leur est encore dévolue.

Les architectures pair à pair reposant sur des serveurs présentent donc des limites importantes. Outre la question financière, se posent les problèmes de la confiance et de la disponibilité des serveurs. Pour améliorer la qualité de service, les utilisateurs tendent à se regrouper sur un petit nombre de serveurs populaires qui deviennent le centre du réseau P2P. La fermeture ou le filtrage de ces serveurs impacte alors fortement l'accès aux services des réseaux P2P concernés et introduit un problème de confiance vis-à-vis des autres serveurs remplaçants. Créer des architectures P2P sans serveur est donc une évolution naturelle pour obtenir des réseaux P2P plus robustes.

### 1.1.2 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 dits non-structurés, bien que leur topologie soit organisée. Soit l'indexation d'une donnée est attribuée à un groupe de pairs déterminé par l'utilisation d'une table de hachage distribuée (DHT), il s'agit des réseaux P2P dits structurés.

4. http ://wiki.vuze.com/w/Distributed_hash_table
5. Peer EXchange

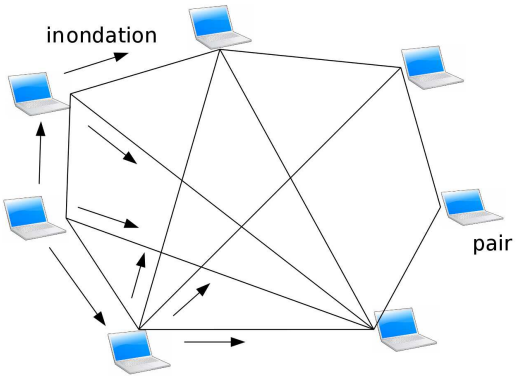


FIGURE 1.4 – Architecture P2P du réseau Gnutella

#### 1.1.2.1 Réseaux P2P non-structurés

Les pairs des réseaux non-structurés 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 soit par inondation, chaque pair envoyant la requête à tous ses voisins, ce qui implique un grand nombre de messages, soit 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**[6] d'une requête définit le nombre de pairs contactés avant d'arrêter sa propagation ;
- la **méthode de propagation** définit la sélection des pairs à qui propager la requête ;

Nous présentons trois réseaux P2P non-structurés mettant en œuvre 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, créé en l'an 2000. Nous présentons ici la version de référence de Gnutella à savoir 0.4 [For00] ayant recours à l'inondation pour propager les messages comme illustré par le schéma 1.4. 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 sauts maximum prévu par le protocole. L'identifiant unique de chaque requête permet aux pairs de ne pas ré-émettre 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 souhaité.

La distribution complète du réseau le rend insensible aux problèmes affectant les précédentes

6. aussi appelé TTL pour Time to Live

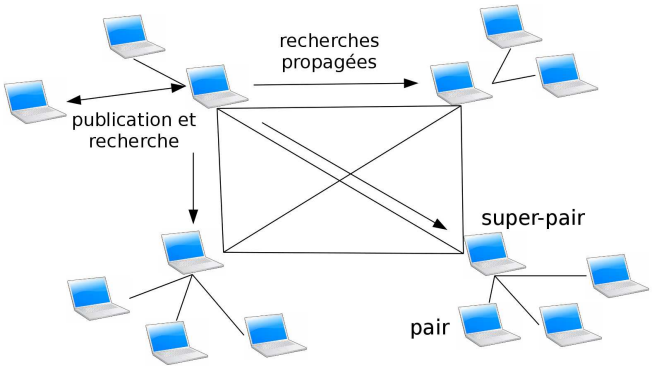


FIGURE 1.5 – Architecture P2P du réseau FastTrack

architectures reposant sur des serveurs. Cependant, cette architecture a également montré d'importantes limites dues à la méthode de propagation employée. L'inondation passe en effet très mal à l'échelle, le nombre de messages émis augmentant rapidement en $O(N)$ avec la taille $N$ 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) [For02] délègue la propagation des requêtes à 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 [LKR04]. 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 contactés lors d'une recherche ce qui permet à l'architecture de mieux passer à 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 terme de bande passante. Le choix des super-pairs est donc crucial car ceux-ci doivent être suffisamment puissants et disponibles. Le schéma 1.5 présente l'architecture de FastTrack. Contrairement aux serveurs qui exécutent un code spécifique, les clients de FastTrack exécutent tous le même code et sont donc succeptibles de jouer le rôle de super-pairs au gré des changements de topologie.

### GIA

L'architecture proposée par GIA en 2003 [CRB+03] 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 vient à mourir. Ces inconvénients n'affectent pas les réseaux P2P structurés.

#### 1.1.2.2 Réseaux P2P structurés

Alors que la majorité des réseaux P2P avec serveurs ou non-structurés a été développée de manière pragmatique, puis étudiée a postériori par la communauté scientifique, les réseaux P2P structurés sont directement issus de la recherche académique qui en a conçu les algorithmes. 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 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 [7], N étant le nombre de pairs dans le réseau. Les réseaux P2P structurés passent donc mieux à l'échelle que les réseaux 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 terme de traitement, car seuls les pairs terminaux évaluent la requête, les autres la routant simplement.

Nous décrivons dans cette section l'architecture générale de deux réseaux P2P structurés, Chord et Kademlia, afin d'en exhiber les caractéristiques. Chord est une des premières DHT conçue, son schéma a ensuite été largement repris et amélioré, notamment par Kademlia qui a été implanté dans de nombreuses applications réelles. D'autres réseaux P2P structurés existent. Ils partagent les mêmes propriétés générales mais peuvent différer quant à l'espace d'adressage (CAN[RFH+01]) et à l'algorithme de routage (Pastry[RD01], Tapestry) employés. Nous verrons plus particulièrement dans le chapitre 3, au travers de l'exemple du réseau KAD, comment une application complète de partage de fichiers peut être construite sur une DHT.

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

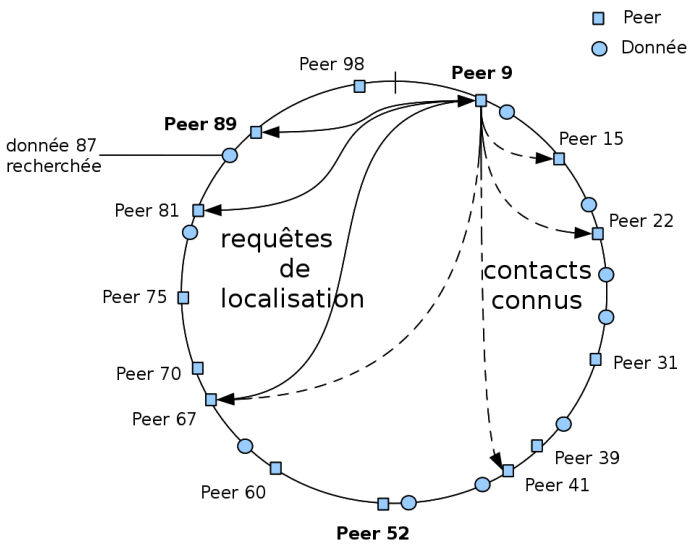


FIGURE 1.6 – Organisation logique du réseau Chord

### Chord

Chord [SMK+01] organise son espace d'adressage suivant un anneau dont les $2^m$ adresses (ou identifiants, id) possibles sont ordonnés 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 la longueur de l'identifiant est définie par : $m = 160$ bit) qui garantit une distribution homogène des ressources sur l'anneau Chord. Chord se limite à une fonction de routage, c'est à 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 fichiers 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 à 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 à trouver un compromis entre la taille de la table de routage comportant $m$ entrées et le routage nécessitant en moyenne $O(log(N))$ messages. Pour cela, un pair, d'identifiant *currentID*, choisit chaque pointeur de sa table de telle sorte que le pair sélectionné soit le seul représentant de l'intervalle $[(currentID + 2^i) - (currentID + 2^{i+1})[$ avec $i + 1 < m$. Un pair connaît 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. Le schéma 1.6 illustre le fonctionnement du réseau chord : le pair 9 souhaitant localiser la ressource 87, dispose tout d'abord des contacts connus dans sa table de routage, puis émet des requêtes de localisation successives jusqu'à trouver le pair 89 responsable de la ressource.

Les principales limites de Chord sont, d'une part sa topologie en anneau obligeant à 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 dont les algorithmes ont été utilisés pour implanter des DHT accueillant des millions d'utilisateurs.

### Kademlia

Dans Kademlia [MM02], les pairs et les ressources possèdent un identifiant de 160 bits définissant leur place dans l'espace logique du système. Kademlia propose une métrique basée sur l'opération XOR pour évaluer les distances sur l'espace d'adressage, quand Chord utilise seulement la notion de "plus proche successeur". Le schéma 1.7 illustre l'utilisation de la métrique XOR pour calculer les distances dans un espace d'adressage à 4 bits pour un pair ayant l'identifiant « 0111 ». De plus, Kademlia stocke des groupes de contacts pour différents intervalles de distance au pair courant et non un seul.

Le premier avantage de cette métrique est d'être symétrique ($A$ XOR $B = B$ XOR $A$) ce qui permet à un pair de Kademlia, lors de la réception de n'importe quelle requête, de pouvoir mémoriser le pair émetteur pour peupler sa table de routage. Par ailleurs, grâce à cette symétrie, la proportion de requêtes reçues venant d'une certaine distance correspond aux besoins en contacts de la table de routage pour cette même distance, minimisant ainsi le nombre de messages nécessaires à son maintien. Le second avantage de Kademlia est un routage moins rigide que Chord. Chord retient dans sa table de routage le pair précédant 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 stricte entre pairs et permet un choix des contacts moins contraignant.

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 (appelé 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 $0 < i \leq 159$ (en pratique, les niveaux les plus bas ne sont pas peuplés par manque de pairs assez proches). Ceci est illustré par les schémas 1.8 et 1.9. Le premier niveau de l'arbre permet donc de contacter des pairs situés dans la moitié opposée 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. Kademlia fournit en outre une politique de sélection des pairs retenus dans la table de routage en fonction de leur temps de session mesuré. Le fait de privilégier les pairs ayant un temps de session important permet d'apporter plus de stabilité à la table de routage car ces derniers ont une probabilité plus grande de rester connectés.

Pour localiser les 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 $\alpha$ contacts les plus proches selon sa table de routage (par exemple, $\alpha = 3$). Ceux-ci ne propagent pas directement la requête mais répondent avec les $\alpha$ contacts les plus proches qu'ils connaissent. A l'issue de cette première étape, les $\alpha$ plus proches contacts reçus sont interrogés et ainsi de suite. Quand les pairs retournés ne se rapprochent plus de l'identifiant *searchid*, la requête de service peut alors être envoyée car les contacts les plus proches sont alors atteints. Conduire la recherche de manière itérative et parallèle génère plus de messages mais rend le

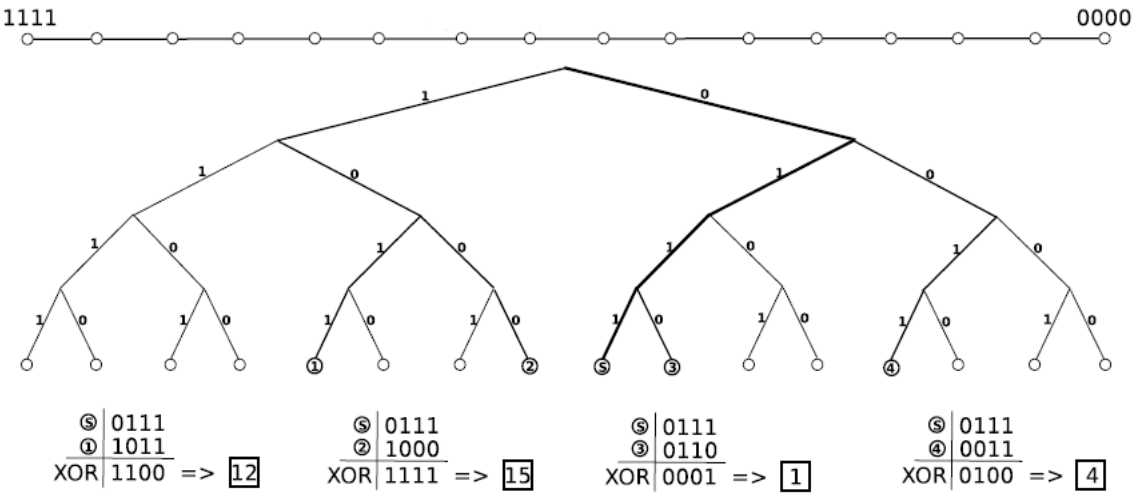


FIGURE 1.7 – Utilisation de la métrique XOR pour calculer les distances entre un pair (P5) et d'autres dans un espace d'adressage à 4 bits, d'après René Brunner [Bru06]

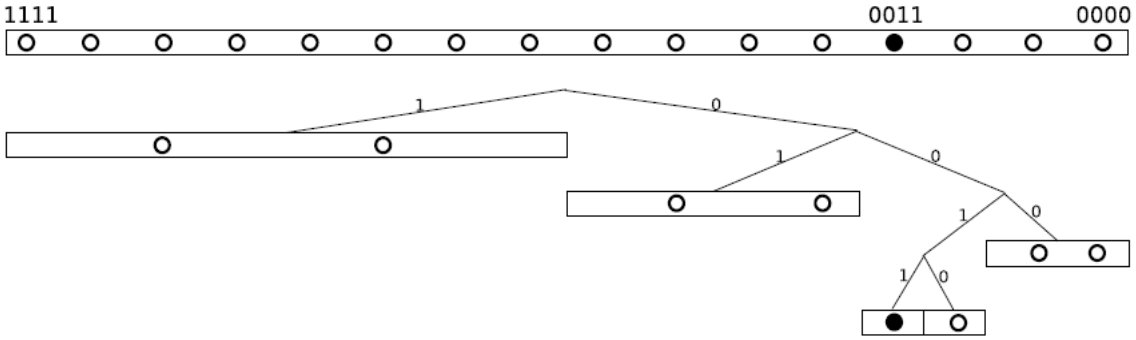


FIGURE 1.8 – Schéma de la table de routage du pair 0011, d'après René Brunner [Bru06]

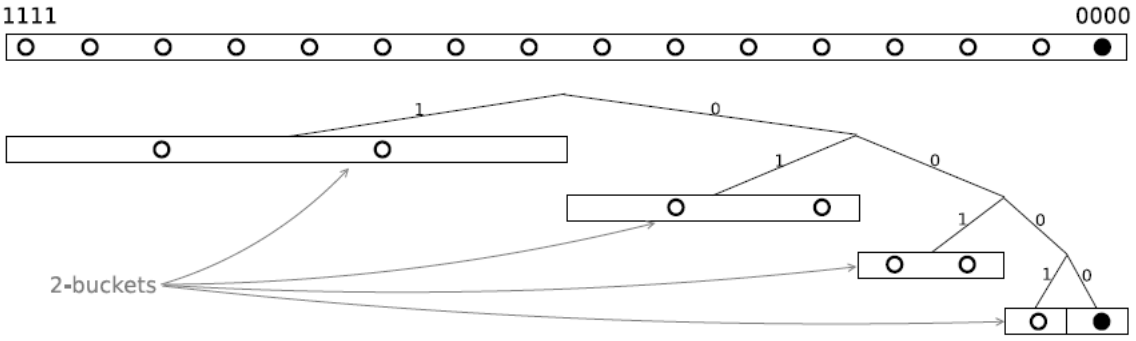


FIGURE 1.9 – Schéma de la table de routage du pair 0011 selon la distance XOR à son propre identifiant, d'après René Brunner [Bru06]

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) à savoir Overnet, KAD, Bittorrent Mainline DHT et Vuze 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-documenté. Le succès de Kademlia est dû à sa simplicité de mise en œuvre (comparable à Chord) ainsi qu'à sa grande efficacité. Outre l'efficacité du routage $O(log(N))$, 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 à d'autres réseaux P2P structurés tels que Pastry et Tapestry) et la réplication nécessaire au bon fonctionnement du système pair à pair est intrinsèquement présente dans Kademlia.

### 1.1.3 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, [KGG02] 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 donnée sur la DHT et fournissent des mécanismes de routage très efficaces pour les localiser en requérant un faible nombre de messages. La structure peut également être maintenue à moindre coût. Les réseaux P2P structurés ont donc prouvé conceptuellement et en pratique leur capacité à passer à l'échelle et à supporter des services P2P.

S'il n'y a plus de différence de performances majeures entre les réseaux non-structurés et structurés, les propriétés intrinsèques de ces derniers et leur large diffusion ont motivé notre choix pour développer des solutions adaptées à cette architecture P2P et plus précisément au système Kademlia. Les avantages et inconvénients des différentes architectures P2P sont synthétisées dans le tableau 1.1. Parmi les inconvénients des réseaux utilisant des serveurs, la pérennité de ces derniers et leur coût sont des inconvénients rédhibitoires limitant le recours à de telles architectures. Les réseaux non-structurés passent difficilement à l'échelle s'ils mènent des recherches exhaustives, sinon, la localisation complète des données n'est pas assurée ce qui est problématique dans le cadre de celles qui sont faiblement répliquées. Les réseaux P2P structurés proposent l'architecture la plus intéressante avec pour seule limite la recherche exacte sur les données. L'article [CCR05] compare diverses architectures non-structurées (Gnutella et GIA) à une DHT (Pastry) sur des paramètres censés désavantager les DHT, à savoir, le coût de maintien de la topologie, la prise en compte de l'hétérogénéité des nœuds et la prise en charge de

recherches complexes. Les auteurs montrent qu'avec quelques optimisations de la DHT, celle-ci est plus performante que les réseaux non structurés dans tous ces domaines.

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

TABLE 1.1 – Comparaison des architectures P2P

## 1.2 Supervision des réseaux pair à pair

Nous avons présenté les différentes architectures P2P existantes au travers d'exemples de réseaux P2P réels. Ces réseaux 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 de quelques réseaux P2P largement déployés. Les données collectées permettent la connaissance de plusieurs objets des réseaux P2P tels que :

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

Selon les architectures P2P, certaines données sont plus ou moins délicates à obtenir. Les architectures P2P mises en œuvre 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 cette section les différentes méthodes de supervision existantes, en les regroupant par famille, et en décrivant leur spécificité. En revanche, puisque nous souhaitons étudier des réseaux P2P ex-

istants, la conception de plateformes globales de supervision et de gestion de réseaux P2P où tous les pairs sont instrumentés a priori, par exemple, tel que proposé dans les travaux [Doy05] et [GSR+09], sort du cadre de notre étude.

## 1.2.1 Méthodes de supervision

### 1.2.1.1 Collecte 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 méthode 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. Ainsi, les traces collectées à partir d'un lien significatif du réseau Internet (campus universitaire, fournisseur d'accès à Internet, ...) sont ensuite 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 exploite également les données applicatives, ce qui nécessite la connaissance du protocole P2P étudié.

La première approche est présentée dans l'article [SW04]. Les auteurs ont ainsi 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 collectées permettent d'apprécier la dynamique générale des réseaux P2P (comment sont distribuées les connexions dans le temps, géographiquement, ou entre les nœuds) et leur impact sur le réseau IP. Si les flux permettent d'étudier les caractéristiques des réseaux P2P au niveau IP, ou de mesurer l'activité des pairs, ils ne renseignent aucunement sur le fonctionnement interne du réseau P2P ou sur les données partagées. Cependant, 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 [SG07][Tut04]. L'article [KBFc04] propose une méthode plus aboutie 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 de détection utilise deux heuristiques, l'une basée sur l'identification des couples (IP source, IP destination) utilisant à la fois les protocoles TCP et UDP pour communiquer, et l'autre basée sur une reconnaissance de motifs (IP, port) caractérisant les communications entre pairs. 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 par une analyse des données applicatives. En revanche, cette approche ne permet pas de distinguer les différents réseaux P2P.

Cette distinction peut être faite en analysant la partie applicative des données collectées qui est la seconde manière d'analyser le trafic collecté. Cette analyse permet aux auteurs de [Tut04] de séparer les flux servant à la signalisation de ceux servant au transfert de fichiers dans le cadre du réseau eDonkey, ce qui fournit une connaissance supplémentaire. L'étude [LRW03] 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 de comprendre davantage 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. Cet effort est d'autant plus nécessaire aujourd'hui qu'une reconnaissance de trafic basée sur les ports est désuète, alors qu'une reconnaissance générique du trafic P2P en limite l'intérêt. En outre, cette approche de supervision ne peut appréhender l'ensemble du réseau P2P.

#### 1.2.1.2 Supervision des serveurs

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 possibilités existent pour effectuer des mesures sur les serveurs, d'une part l'instrumentation du serveur dont on souhaite acquérir les données, d'autre part l'interrogation du serveur via les mêmes 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) pour les traiter ultérieurement. Mise à part l'insertion du serveur dans le réseau, l'instrumentation a l'avantage d'être passive, la supervision ne nécessitant l'envoi d'aucun message spécifique.

La solution de supervision la plus directe consiste à obtenir un journal synthétisant l'activité du serveur. Dans le cadre du réseau Bittorrent [IUKB+04], 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. Cette approche montre également des limites dans le cadre du réseau eDonkey [LBGL05] [GLLB04] car le journal fourni par défaut par le principal programme serveur (lugdunum) n'enregistre qu'une partie des informations. Le journal contient ainsi 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 [ALM09a] 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 analysant et anonymisant 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 pour 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, dans le cadre du réseau Bittorrent, l'activité d'un torrent peut être facilement supervisée 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.

La supervision de serveurs est par définition inapplicable aux réseaux P2P complètement distribués pour lesquels d'autres approches sont considérées. Ces autres méthodes de supervision ne reposent que sur des communications entre clients et peuvent être regroupées en 3 catégories : les pots de miel, les explorateurs et enfin les sondes distribuées.

### 1.2.1.3 Pots de miel

Les pots de miel (en anglais « honeypots ») viennent du domaine de la sécurité informatique et sont utilisés pour attirer les utilisateurs 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 » [8]. 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 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 [BSCF07] et [LN07] proposent des architectures permettant de traquer les pairs recherchant des contenus illégaux diffusés dans les réseaux P2P. L'architecture proposée par [LN07] 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'à une réelle architecture de honeypot. Elle a cependant l'avantage de découvrir les fournisseurs des fichiers. L'architecture de [BSCF07] 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-clés malveillants. Une analyse statistique des données collectées pour l'ensemble des mots-clés 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.

8. « *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`

La principale limite de cette architecture est d'utiliser de faux fichiers comme appâts qui n'apparaissent pas comme étant populaires aux autres pairs. De plus, la génération automatique des noms limite encore davantage l'attractivité des appâts.

Dans [ALM09b], 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 données 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 [ALM09b] ou de mots-clés [BSCF07] précis au sein du réseau. L'étude des mots-clés 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.

#### 1.2.1.4 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, l'explorateur 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 pouvant être obtenues en interrogeant les pairs. Nous présentons dans cette section trois types d'explorateurs, chacun conçu pour une architecture P2P différente.

Les explorations du réseau Gnutella [KGG02] et [RIF02] 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 nœuds (PONG) qui seront à leur tour contactés séquentiellement. L'exploration séquentielle est cependant très lente car l'explorateur doit attendre un certain temps la réponse de pairs déconnectés. Pour pouvoir explorer rapidement de grands réseaux, [RIF02] distribue l'exploration sur plusieurs clients (50), synchronisés par un serveur alors que [KGG02] émet des annonces avec un TTL important ce qui surcharge davantage le réseau. Pour chaque pair, l'explorateur iden-

tifie son adresse IP, la liste de ses voisins ainsi que le nombre et la taille des fichiers partagés. [KGG02] 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. L'article [LKXR05] présente un explorateur du réseau Kazaa capable d'explorer les données partagées en émettant de nombreuses requêtes de recherche sur l'ensemble des supernœuds depuis 10 machines exécutant un total de 1600 processus-légers. Les mots-clés recherchés doivent être définis a priori, l'explorateur émettant des requêtes pour un ensemble de mots-clés définis. Multiplier les mots-clés étudiés rend cette exploration très intrusive.

Les stratégies d'exploration mises en œuvre sur le réseau eDonkey [LHKM04] [HKL+06] [YMS+06] sont différentes. L'explorateur utilisé dans [LHKM04] et [HKL+06] consiste en deux étapes. 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 [LHKM04] et, plus généralement, de comprendre le comportement des pairs par rapport à l'application de partage de fichiers [HKL+06]. 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 [YMS+06] à obtenir la liste de pairs par un autre moyen, en émettant des recherches de sources pour certains fichiers, eux mêmes obtenus à partir de mots clés (cette approche a également été utilisée pour explorer le réseau Napster [KGG02]). La liste des pairs ainsi obtenue est malheureusement contrainte par la langue des mots-clés initiaux (dans l'étude : l'anglais et le chinois). Ensuite, les pairs trouvés ne sont pas tous contactables directement en raison des pare-feux. 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 [SENB07b] et [SBEN07] explorent le réseau KAD à l'aide d'un outil nommé Blizzard. Celui-ci connait quelques pairs a 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 sont réalisées par deux processus légers asynchrones partageant la liste des pairs. Blizzard est ainsi capable de découvrir le réseau KAD en quelques dizaines de minutes, ou une zone ($1/256^{eme}$) en quelques dizaines 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 [YFX+09] 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 ce qui rend difficile la terminaison de l'exploration. 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. Les données indexées au sein des DHT sont cependant invisibles aux explorateurs.

Les explorateurs sont un outil de supervision pouvant être adapté 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 limitations majeures. D'une part, les explorateurs ne permettent pas d'appréhender les contenus partagés directement par les pairs (sauf fonctionnalité particulière du client autorisant la découverte directe des contenus d'un pair), 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. La dernière méthode de supervision étudiée ici, l'utilisation de sondes, ne présente pas ces limites.

#### 1.2.1.5 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 utilisées 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. [And01] et [Mar02] 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, query...), 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, [Sri01] et [KLVW04] étudient en particulier les chaînes de caractères recherchées dans le réseau ce qui permet d'évaluer la popularité des contenus. [KLVW04] note également que les fichiers trouvés en réponse pourraient être analysés. Les auteurs de [AC07] 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 utilisés : les unes participent au partage d'un fichier, les autres à la DHT. Ainsi, [BMGS09] essaie d'identifier les pairs participant au partage d'un fichier en injectant des sondes dans le groupe de pairs 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 [CW07] ont injecté plusieurs sondes dans chacune des deux DHTs utilisées par Bittorrent afin de mesurer et comparer leurs performances. Les articles [FPJ+07] et [WH10] 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 (i.e. le hash d'un « torrent »), même si 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, [WH10] utilise en premier lieu un explorateur 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 diffère selon les études. Si l'augmentation du nombre de sondes 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 [MRGS09] [Mem08] 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 requêtes 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 pas de constater précisément les transferts de fichiers.

### 1.2.2 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 posent 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. Le tableau 1.2 résume les différentes manières de superviser un réseau P2P avec serveur. Cependant, la vue du réseau reste là aussi partielle au(x) serveur(s) supervisé(s). 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'autres moyens de supervision. Nous ne nous attarderons donc pas davantage sur la collecte de trafic et les mesures sur serveurs.

Trois méthodes de supervision sont possibles lorsqu'il s'agit de superviser des réseaux P2P complètement distribués : les pots de miel, les explorateurs et les sondes distribuées. Dans le cadre de cette thèse, nous nous intéressons plus particulièrement à la supervision des réseaux P2P structurés et notamment à leurs contenus. Dès lors, si l'on considère la supervision de contenus partagés dans les réseaux P2P, aucune des solutions de supervision actuelles n'est satisfaisante. Les forces et faiblesses de chacune de ces approches de supervision appliquées à l'étude des contenus sont présentées dans le tableau 1.3.

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 pots de miel sont capables de constater les transferts de fichiers mais le nombre de contenus supervisables par un même pot de miel 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. D'une part, il est difficile d'étudier un ensemble de contenus lié à une thématique (définie par ensemble de mots-clés) avec cette approche, d'autre

| Méthode de supervision | Avantages | Inconvénients |
|---|---|---|
| **Trafic du serveur** | – Connaissance des pairs<br>– Connaissance des contenus<br>– Supervision passive | – Difficulté du traitement des données<br>– Vue partielle du réseau<br>– Difficulté d'instrumenter un serveur existant (accord de l'administrateur, impact sur les performances, ...) |
| **Journaux du serveur** | – Traitement facile des données<br>– Supervision passive | – Informations enregistrées limitées<br>– Vue partielle du réseau<br>– Difficulté d'obtention des journaux de serveurs existants |
| **Requêtes sur serveurs** | – Plusieurs serveurs contactables<br>– Facilité de mise en œuvre | – Requêtes spécifiques à un contenu<br>– Supervision active |

TABLE 1.2 – Comparaison des méthodes de supervision des serveurs de réseaux P2P

part, un pot de miel ne peut découvrir les pairs partageant déjà un fichier mais uniquement ceux souhaitant l'obtenir. Les sondes distribuées permettent d'intercepter les requêtes de recherche et de publication mais n'ont pas la connaissance à priori des contenus ainsi supervisés. Par ailleurs, les transferts de fichiers sont invisibles avec cette méthode, au contraire des pots de miel. Pour finir, les sondes comme les pots de miel sont sujets aux faux positifs du fait de la pollution de la DHT, certains fichiers pouvant être référencés par des mots-clés sans relation. Collecter des adresses IP sans s'assurer précisément de l'activité et de l'intentionnalité du pair conduit à des erreurs de détection. Par exemple, l'annonce usurpée d'adresses IP dans un tracker BitTorrent a conduit à une suspension de leur connexion Internet [PKK08], malgré le fait que les hôtes en question aient été des imprimantes n'ayant jamais téléchargé ou partagé une partie de fichier.

## Conclusion

Nous avons tout d'abord présenté dans ce chapitre les différentes architectures P2P existantes en décrivant leur fonctionnement et en comparant leurs propriétés respectives. Dans le cadre de cette thèse, nous avons choisi de nous intéresser à la supervision des réseaux P2P structurés pour plusieurs raisons :

- il s'agit de l'architecture la plus prometteuse pour mettre en œuvre le pair-à-pair (totalement distribuée, passant à l'échelle, garantissant la localisation des ressources) ;
- ils partagent une base théorique commune facilitant la réutilisation des travaux développés pour leur modèle ;
- ils sont déployés à très grande échelle avec plusieurs millions de pairs simultanés (KAD,

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

TABLE 1.3 – Comparaison des méthodes de supervision appliquées aux contenus des réseaux P2P complètement distribués

BitTorrent Mainline DHT) ;

Nous ne considérerons pas dans notre étude les réseaux P2P anonymes, ceux-ci étant encore trop peu développés, en grande partie à cause de leurs problèmes de performances les rendant inadaptés au partage de fichiers volumineux tels que les vidéos. Par ailleurs, dans l'application qui nous préoccupe en particulier, à savoir la supervision de contenus pédophiles sur les réseaux P2P, il a déjà été prouvé que certains sont couramment diffusés sur le réseau eD2k. A notre connaissance, cette observation n'a pas encore été réalisée pour les réseaux anonymes (FreeNet, GnuNet...).

Nous avons ensuite présenté une vue exhaustive des méthodes de supervision appliquées aux réseaux P2P, en comparant pour chacune d'elles les contraintes d'utilisation quant au réseau cible et aux informations collectées. L'étude des différentes architectures de supervision a montré qu'aucune n'est actuellement satisfaisante quand il s'agit de superviser de manière non intrusive l'activité des contenus partagés. En particulier, le problème des faux-positifs liés à la pollution pouvant amener les utilisateurs à accéder non intentionnellement à des contenus n'est pas considéré. Pour cela, nous proposons dans le chapitre 5 une nouvelle approche de supervision permettant d'allier les avantages des architectures de pots de miel et de sondes distribuées sans en présenter les inconvénients. La supervision des contenus au sein des réseaux P2P est donc d'autant plus délicate que ces derniers sont victimes de problèmes de sécurité majeurs tels que la pollution ou l'attaque Sybil. Ces enjeux font l'objet du prochain chapitre.

# Chapitre 2

# Sécurité des contenus dans les réseaux pair à pair

**Sommaire**



## Introduction

Les problèmes de sécurité affectant les réseaux P2P sont dus à l'absence de contrôle centralisé et à l'autonomie des pairs. Le meilleur exemple illustrant ces limites est le comportement égoïste adopté par la majorité des pairs, bien qu'étant pénalisant pour le réseau. Ce comportement est appelé « free riding » et dégrade énormément la qualité de services des réseaux P2P, en concentrant l'ensemble de la charge sur un nombre réduit de pairs, ce qui viole le principe fondamental stipulant que chaque pair est à la fois client et serveur. Certaines études ont estimé l'impact de ces comportements sur le réseau : [AH00] et [HCW05] ont ainsi supervisé Gnutella et constaté la tragédie du bien commun [Har68] qui est la conséquence des comportements égoïstes : 70% des utilisateurs ne partagent rien et 50% des ressources sont partagées par seulement 1% des utilisateurs. Les auteurs mettent ainsi en évidence les limites d'un système basé sur le volontarisme, sans contrôle, et où chacun est anonyme ce qui se traduit également par d'importants problèmes de sécurité.

La sécurité des réseaux P2P est un problème général convrant plusieurs attaques. Plusieurs études de synthèse recensent les problèmes de sécurité des réseaux P2P [SM02] [Wal03] et leurs solutions possibles [UPvS11]. Nous classifions les problèmes de sécurité des réseaux P2P en trois catégories : ceux éprouvant la robustesse de l'architecture P2P, ceux visant les services du système P2P et ceux utilisant l'infrastructure à des fins malveillantes. La majorité des considérations de

sécurité concerne la robustesse de l'architecture pair à pair aux attaques, plus précisément, sa capacité à maintenir les communications entre pairs et à localiser les données malgré certains comportements malveillants. Cette catégorie inclut notamment les attaques de type routage, visant à corrompre les requêtes des pairs ou leur table de routage afin de perturber la localisation des pairs et pouvant aller jusqu'à la partition du réseau. L'infrastructure P2P peut également être utilisée à des fins malveillantes et ainsi être le support d'un botnet tel que Storm [HSD+08], ou encore à une attaque de déni de service distribué ciblant un système tiers [NR06]. Dans ce chapitre, nous restreignons notre étude des considérations de sécurité à celles impactant les services d'un système P2P dans le cadre d'une application de partage de fichiers.

Autrement dit, nous nous intéressons aux problèmes de sécurité affectant les contenus partagés dans les réseaux P2P. Nous définissons la sécurité des contenus par trois propriétés devant être vérifiées, à savoir :

1. la description d'un contenu doit correspondre au contenu réel ;
2. l'indexation des contenus doit être pérenne au sein du réseau ;
3. les contenus « dangereux » pour les utilisateurs doivent être proscrits.

Il existe principalement deux problèmes de sécurité affectant les contenus des réseaux P2P. D'une part, ils sont utilisés pour **diffuser des contenus illégaux** tels que des virus ou des contenus à caractère pédophile. Cette simple diffusion est un problème en soi mais est d'autant plus grave que ces contenus malveillants peuvent être téléchargés de manière non intentionnelle par des utilisateurs du fait de la **pollution** des réseaux P2P. D'autre part, des entités peuvent perturber le réseau en créant et coordonnant de nombreux faux pairs ce qui leur permet d'accroître leur influence sur le réseau. Cette attaque, appelée **attaque Sybil**, a de nombreuses applications dont certaines posent problème pour la sécurité des contenus du réseau (attaque éclipse, pollution...). Tous ces problèmes de sécurité sont intimement liés. Ainsi, certaines formes de pollution servent à la diffusion de contenus malveillants. De même, l'attaque Sybil peut être utilisée pour réaliser la polution du réseau.

Nous présentons dans ce chapitre l'état de l'art relatif à ces problèmes de sécurité, en énonçant leur principe, les observations réalisées et les solutions envisagées au travers de nombreuses études.

## 2.1 Contenus illégaux et pollution

Un des problèmes majeurs des réseaux P2P de partage de fichiers est la diffusion de la pollution en leur sein. Un fichier est dit pollué si le contenu fourni par le service P2P ne correspond pas 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 une fois téléchargé ou encore, complètement fictif. La pollution dégrade significativement la qualité de service des réseaux P2P. D'une part, en cas de pollution, l'utilisateur doit vérifier le contenu du fichier et rechercher un autre fichier sain. D'autre part, la diffusion de la pollution consomme inutilement les ressources limitées du réseau P2P. La pollution, lorsqu'elle est associée à la diffusion de contenus malveillants, pose surtout des problèmes de sécurité pour les utilisateurs. Ainsi, un contenu pornographique ou à caractère pédophile peut être diffusé par des pairs malveillants avec le nom de fichier d'un dessin animé, ou un virus informatique avec des noms de fichiers exécutables légitimes. Nous étudions dans cette sous-partie les différentes recherches menées sur la diffusion de contenus illégaux et la pollution des réseaux P2P, en nous intéressant à la compréhension du phénomène et son impact sur la sécurité.

### 2.1.1 Diffusion de contenus illégaux

Très peu d'études académiques se sont intéressées à l'étude et à la quantification des différents contenus illégaux partagés sur les réseaux P2P. Parmi elles, les auteurs de [HWCG06] ont montré que des fichiers illégaux, notamment à caractère pédophile, sont diffusés sur le réseau P2P Gnutella et représentent 1,6% des recherches et 2,6% des réponses. Les pairs impliqués dans cette activité sont peu nombreux mais en revanche très actifs et semblent former une sous-communauté caractérisable. En particulier, 57% des pairs partageant ces contenus ne partagent rien d'autre. Cette étude reste cependant préliminaire quant aux données collectées et à leur analyse. D'autres études récentes visant les contenus pédophiles ont été menées. D'une part dans le cadre du projet MAPE (Mesure et Analyse des Échanges Pair-à-pair pour combattre la pédocriminalité et caractériser le trafic) [9] de l'Agence Nationale de la Recherche auquel cette thèse contribue, et d'autre part du projet européen MAPAP (Measurement and Analysis of P2P Activity Against Paedophile Content) [10]. Ces deux projets ont donné lieu à de nombreux rapports [Lat09] montrant la réalité des activités pédophiles sur les réseaux P2P. En particulier, l'étude [MLlGG08] présente des statistiques sur les mots-clés pédophiles observés dans les recherches des pairs ou les noms de fichiers partagés sur un important serveur eDonkey. Les statistiques amènent de nouvelles connaissances telles que la popularité des différents mots-clés ou les âges les plus recherchés. Les données collectées confirment également que le réseau ed2k, et par conséquent l'application eMule qui permet également la connexion à KAD, est utilisé pour diffuser ces contenus. Par ailleurs, d'après l'étude [Dav09], il existe une corrélation importante entre les contenus propagés et les activités pédophiles puisque 85% des personnes collectionnant ces contenus ont également commis des actes de pédophilie. Ceci motive d'autant plus le fait de pouvoir superviser précisément ces utilisateurs dans les systèmes P2P.

La diffusion de contenus illégaux a donc conduit à la supervision des réseaux P2P par les forces de l'ordre, notamment dans le cadre de la lutte contre les contenus pédophiles, mais également par des entreprises mandatées par des ayants droits, dans le cas d'infractions au droit d'auteur. Dans [PKK08] et [BFB08] est analysée cette supervision respectivement dans les réseaux Bittorrent et Gnutella. Les auteurs de [BFB08] étudient la pertinence des filtres d'adresses IP créés par des utilisateurs pour éviter les pairs supervisant le réseau. Ils montrent qu'en l'absence d'utilisation d'une liste noire, un pair est certain de communiquer avec ces adresses suspicieuses mais que la probabilité d'être supervisé chute à 1% en filtrant les cinq blocs d'adresses IP les plus actifs. Les auteurs de [PKK08] ont injecté quant à eux des adresses IP de diverses machines connectées (imprimantes, routeurs) dans la liste de pairs partageant un contenu (maintenue par un tracker). Bien que n'ayant pas participé au partage de fichiers, les adresses IP utilisées ont été l'objet de demande de fermeture de connexion dans le cadre de la loi américaine DMCA [11]. Cette étude démontre ainsi que la supervision des réseaux P2P n'est actuellement pas assez rigoureuse et peut être facilement corrompue par une pollution très basique. Les auteurs présentent également d'autres sources potentielles de faux positifs mais, étudiant seulement BitTorrent, ne mentionnent pas d'autres formes de pollution problématiques, notamment touchant les mécanismes d'indexation des DHTs que nous décrirons ultérieurement. La diffusion de contenus illégaux est d'autant plus problématique qu'elle peut être associée à certaines formes de pollution des contenus et ainsi toucher des utilisateurs à leur insu.

9. Measurement and Analysis of Peer-to-peer Exchanges for pedocriminality fighting and traffic profiling
10. Programme «Safer Internet Programme : Empowering and Protecting Children Online» de l'Union Européenne
11. Digital Millenium Copyright Act

### 2.1.2 Stratégies de pollution

Nous différencions deux formes de pollution. La première consiste à injecter de vrais fichiers erronés dans le réseau, la seconde consiste à corrompre le référencement des fichiers dans le réseau P2P en y ajoutant de fausses entrées.

#### 2.1.2.1 Pollution des contenus

L'article [CWC05] 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 soit 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 pollution 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 recherche, difficilement détectables, et se diffusent facilement. Quelques centaines d'annonces permettent ainsi de réduire la disponibilité des fichiers réels au profit des faux.

L'article [TP10] 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 ressource via une URI. Une contre-mesure possible est la validation des fichiers par des codes correcteurs d'erreurs, d'ores et déjà utilisés dans le réseau ed2k (AICH - Advanced Intelligent Corruption Handling) [12], 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.

#### 2.1.2.2 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 [LKXR05]. 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 apparaît 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 [LNR06] étudient cette pollution dans deux réseaux P2P : FastTrack 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 émises par chaque pair pour un même

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

contenu 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 [LNR06] 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 du 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 falsifiant 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 peut être téléchargé mais son contenu est sans relation avec le nom du fichier sélectionné par l'utilisateur. Cette pollution est plus néfaste, car un fichier indésirable est téléchargé par l'utilisateur, et engendre d'importants problèmes de sécurité, un contenu offensant ou illégal pouvant être téléchargé non intentionnellement. Par ailleurs, cette pollution n'est pas détectable par les métriques présentées dans [LNR06] car elle ne nécessite pas la création de nombreuses entrées erronées. Nous étudierons cette forme de pollution sur le réseau KAD dans le chapitre 7.

Une seule étude [LMSW10] a récemment considéré la question de la pollution sur le réseau P2P KAD. Les auteurs n'étudient pas la pollution réelle du réseau mais décrivent et expérimentent deux différentes méthodes de pollution et leurs applications. La première consiste en l'insertion de nœuds à proximité de l'information à corrompre sur la DHT, par exemple un mot-clé. Lorsque les nœuds malveillants reçoivent des requêtes, ils répondent avec de fausses informations, par exemple de faux fichiers. KAD limitant une recherche à 300 réponses, le fait de générer 300 fausses informations empêche tout autre résultat valide. L'insertion d'un tel nœud peut polluer 40% des recherches émises, et de trois nœuds jusqu'à 95% des recherches. La seconde attaque est basée sur la limite de stockage imposée aux pairs pour l'indexation d'une information particulière, par exemple, 50000 fichiers par mot-clé. L'attaque consiste à remplir la table des pairs indexant une donnée avec de fausses entrées de telle sorte qu'il ne puisse plus accepter de nouvelles entrées valides. Cette attaque est efficace et permet de polluer 80% des requêtes de recherche en ciblant les 12 pairs les plus proches d'une information. L'attaque doit cependant être ré-actualisée périodiquement à cause du va-et-vient des pairs responsables de l'information. La pollution est ensuite utilisée pour réaliser une attaque DDoS en mentionnant la victime comme source des faux fichiers annoncés. Les auteurs ne mentionnent cependant pas si ces attaques sont réellement appliquées dans le réseau.

Il est à noter que 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 des fichiers « torrent »), 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. De tels sites web sont régulièrement victimes de poursuites comme mentionné en section 1.1.

### 2.1.3 Diffusion de la pollution

Afin de comprendre les facteurs influençant la diffusion de la pollution, l'article [DKK+05] 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, quelles que soient les 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-mêmes). Les architectures P2P sans hiérarchie sont plus résistantes à l'insertion aléatoire de pairs malveillants car moins de nœuds sont contactés lors de la recherche. 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.

Liang et al. [LKXR05] établissent ainsi un bilan précis de la pollution affectant le réseau P2P Kazaa. 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 apparaît que les fichiers populaires sont extrêmement pollués (73% des fichiers, 62% des fichiers distincts) en dépit du mécanisme de notation des fichiers fourni par les clients, alors que d'autres fichiers ne sont pas du tout affectés.

Les mêmes auteurs [LKXR05] identifient les sources originelles de la pollution dans Kazaa comme étant des sociétés polluant intentionnellement le réseau pour en réduire l'attrait. La pollution est ensuite diffusée par les pairs eux mêmes. L'article [LCC+06] propose ainsi un modèle de propagation de la pollution au sein du réseau P2P Kazaa, 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.

### 2.1.4 Solutions proposées

Plusieurs solutions ont été envisagées pour lutter contre la pollution. L'article [LKXR05] classe les solutions potentielles en deux catégories : celles nécessitant le téléchargement du fichier (vérification manuelle ou semi-automatique) et celles pouvant être appliquées avant même celui-ci (systèmes de confiance ou réputation). Nous présentons ici des solutions du deuxième type car elles seules permettent de protéger les utilisateurs en détectant la pollution a priori. Les articles [LNR05] et [LNR06] proposent une méthode capable de détecter les sous-réseaux diffusant la pollution sur Kazaa 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 peut 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) et d'une autorité de confiance le fournissant. Par ailleurs, d'autres formes de pollution (corruption d'indexation) ne sont pas détectables par la méthode proposée.

L'article [DHX09] décrit formellement un mécanisme de validation des fichiers indexés basé sur les filtres de bloom. L'authenticité des fichiers est calculée par une fonction analysant les

noms indexés, sans que celle-ci soit spécifiée. Les auteurs distinguent les faux fichiers qui sont invalides des fichiers incorrects qui ne correspondent pas à leur description, ce que nous appelons une corruption d'indexation. Les pairs indexant vérifient ainsi l'authenticité d'une annonce par rapport à l'existant. Chaque donnée indexée par un pair est signée par la source originale qui crée une preuve dans un filtre de bloom. Sa consultation permet de détecter l'injection de faux fichiers. Cette solution a néanmoins besoin d'un mécanisme de réputation pour détecter si les premiers pairs publiant un certain contenu sont honnêtes. De surcroit, son application n'est pas possible dans un réseau existant (les anciens pairs ne pouvant signer) et aucune indication n'est donnée quant au changement d'identité des pairs (« whitewash ») ou a une attaque massive (Sybil), ni quant au coût de la solution.

L'article [SRRS09] présente une autre approche basée sur le filtrage collaboratif des fichiers pollués. Leur solution appelée « Winnowing » repose sur les pairs indexant un contenu pour filtrer les annonces erronées. Ceux-ci effectuent tout d'abord plusieurs tests : ils vérifient que l'émetteur fait partie du réseau P2P pour limiter l'usurpation d'adresse IP et qu'un des mots-clés du nom de fichier correspond à l'entrée publiée. Ils collectent ensuite les votes d'utilisateurs sur la pollution du fichier par rapport au mot-clé. Les votes contraires des quelques pollueurs sont limités par une pondération inversement proportionnelle au nombre de votes émis par une adresse IP. Cependant, cette limite reste locale à un pair indexant. Un même pair peut donc corrompre les votes sur l'ensemble de la DHT. Les points forts de cette solution sont l'absence d'autorité centrale et de contrainte forte qui la rendent applicable au réseau existant. Cependant cette solution ne fonctionne pas si les nœuds indexant sont eux mêmes corrompus par une attaque ciblée ou s'ils diffusent directement la pollution. Par ailleurs, le coût de la protection est très élevé en terme de messages, les vérifications doublant le coût de l'indexation, indépendamment des votes, ce qui est problématique étant donné que ce service est de loin le plus utilisé sur la DHT.

Des mécanismes de réputation ont été appliqués aux pairs ou aux contenus afin de limiter la pollution. L'article [CA07] présente un système hybride de réputation pouvant s'appliquer aux deux objets tout en ne faisant pas d'hypothèse a priori 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 votes. Ce mauvais vote peut être réalisé par des pairs honnêtes en cas de corruption de l'indexation d'un fichier : 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 [LMV08] et [lGLMV09] explorent une autre méthode pour lutter contre la corruption de l'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 la section 1.2.1.4) afin de 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. Il suffit en outre qu'une seule source partage le fichier avec un nom différent pour qu'il soit considéré comme pollué ce qui semble être difficilement applicable étant donné le nombre de sources des

fichiers populaires. 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 où encore l'obtention des différents noms sans explorateur.

Après avoir expérimenté des méthodes de pollution dans KAD, les auteurs de [LMSW10] discutent de solutions possibles à ce problème. Concernant le problème des tables d'indexation remplies, une contre-mesure limitant le nombre d'entrées annoncées par une même adresse IP limiterait l'efficacité de l'attaque. Le problème de l'insertion de nœuds est en revanche plus délicat. Il semble irréaliste de contraindre les nœuds à une position sur la DHT selon leur seule adresse IP (problème des NAT, adresse IP dynamique, ...), et inclure le port ou une partie générée par l'utilisateur pour construire un identifiant unique rendrait par ailleurs la solution trop permissive (attaque par force brute). Le problème reste donc ouvert si l'on exclut toute forme d'autorité centralisée. D'autres solutions seront étudiées en section 2.2.3 dans le cadre de l'attaque Sybil.

Au terme de cette section, il apparaît tout d'abord un manque de connaissances récentes sur les formes de pollution, les principaux réseaux cibles des précédentes mesures de pollution étant obsolètes (Kazaa, Overnet). Ainsi, la corruption du système d'indexation affectant les DHT n'a pas été 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 corrompant l'indexation des fichiers afin de protéger les utilisateurs de contenus hautement indésirables.

## 2.2 L'Attaque Sybil

### 2.2.1 Principe

En présentant les réseaux P2P, nous avons établi 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 pérennité. 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 [Dou02], 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 provient du 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. L'auteur montre qu'en l'absence d'une autorité de certification centralisée des identités, l'attaque Sybil ne peut être évitée entièrement. Cependant, des mécanismes peuvent être conçus pour limiter l'ampleur de l'attaque en la rendant plus difficile.

Telle que définie dans [Dou02], l'attaque Sybil viole seulement l'indépendance des pairs. En pratique, ses nombreuses mises en œuvre impliquent également un positionnement non-aléatoire des pairs insérés, si bien que l'attaque Sybil comprend par extension cette propriété. 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 [CDG+02] 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.

### 2.2.2 Applications

L'article [SENB07a] 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, injectent 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 (i.e. $1/256^{eme}$ 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 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 [SENB07a] 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 [KLR09] perfectionne encore davantage l'attaque éclipse dans KAD en créant une succession de Sybils dont la proximité est toujours croissante avec l'identifiant ciblé. De cette manière, le routage semble progresseer indéfiniment vers l'identifiant sans jamais l'atteindre car le délai alloué à la recherche expire sans que l'information n'ait été trouvée. L'article [LNR06] présente une forme de pollution du mécanisme d'indexation d'Overnet pouvant être affiliée à l'attaque Sybil. Les nœuds malveillants sont postionnés autour d'un mot-clé indexé sur la DHT et annoncent de faux fichiers. Cette attaque peut également affecter le réseau KAD [LMSW10].

Les auteurs de [CDG+02] et [SNDW06] décrivent une autre forme d'attaque 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 routage de ces pairs. Tout accès des victimes au réseau P2P est alors contrôlé par les attaquants. Une telle attaque a été mise en œuvre avec succès contre le réseau KAD [WTCT+08] 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 [KBM07]. 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 du swarm.

L'attaque Sybil est également le principal obstacle à l'utilisation de mécanismes de réputation dans les réseaux P2P. Les mécanismes de réputation visent à améliorer la qualité de service des réseaux P2P, en pénalisant les comportements égoïstes, ou leur sécurité, en limitant notamment la pollution [KSGM03] [MM03]. Ceux-ci souffrent cependant largement de l'attaque Sybil [LSM06] et plus généralement de la faible valeur de l'identité dans ces réseaux. L'article [HZNR09] présente les diverses attaques affectant les mécanimes de réputation et décrit notamment l'attaque Sybil. Par conséquent, certains auteurs considèrent ce problème dès la conception de leur mécanisme de réputation [CF05] [LGC+09] afin d'y être plus résistant. Les mécanismes de réputation ne permettent donc pas de lutter contre l'attaque Sybil, eux mêmes y étant très vulnérables.

Enfin, et contrairement aux précédentes applications, l'attaque Sybil n'est pas toujours utilisée 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. Ces attaques Sybil ont été appliquées avec succès à des fins de supervision sur plusieurs réseaux P2P dont Gnutella [KLVW04] [AC07], Bittorrent [FPJ+07] [WH10] et KAD [MRGS09]. 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 majeur 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. Le contrôle du mécanisme d'indexation par une attaque Sybil permet également la diffusion massive de pollution. Il est donc primordial de pouvoir détecter de telles attaques afin d'identifier les pairs malveillants pour en protéger les pairs légitimes.

### 2.2.3 Solutions contre l'attaque Sybil

De nombreuses solutions ont été imaginées pour limiter l'attaque Sybil [UPvS11] par l'ajout de contraintes à l'insertion d'un pair dans le réseau. Ces protections peuvent être classées en trois catégories : les contraintes structurelles, les vérifications de ressources et la certification des identifiants. Nous présentons dans cette partie l'état de l'art de ces solutions en soulignant pour chacune ses forces et faiblesses. Nous considérerons notamment la question de l'applicabilité de chaque solution à un réseau P2P existant et non initialisé à zéro.

De nombreuses solutions ayant été proposées pour lutter contre l'attaque Sybil sur les DHT, plusieurs études synthétiques ont été rédigées. [SL04] différencie les attaques affectant le routage

des messages de celles affectant le stockage des informations et simule pour chacune d'elles l'impact sur la qualité de service dans Chord. [LSM06] recense de nombreuses études sur l'attaque Sybil en les classifiant selon plusieurs catégories, les principales étant : l'absence de solution (18 références), l'utilisation d'une autorité de certification (27), ou d'un test des ressources des pairs (15). Parmi les domaines d'application affectés par l'attaque Sybil, les mécanismes de réputation sont majoritaires (28 références). Enfin, l'étude [UPvS11], présente en détail de nombreuses solutions proposées contre l'attaque Sybil, l'attaque éclipse et leur impact sur le routage des réseaux P2P structurés. Il apparaît que peu d'études essayent de limiter l'impact de l'attaque sur les services liés aux données indexées mais se concentrent plutôt sur la robustesse du routage. De même très peu d'entre elles considèrent le cas de réseaux P2P existants.

#### 2.2.3.1 Contraintes structurelles

De nombreuses solutions à l'attaque Sybil sont structurelles dans le sens où elles imposent de nouvelles contraintes sur la topologie même du réseau P2P ou sur son algorithme de routage, afin de limiter le nombre de Sybils et les méfaits possibles.

Les auteurs de [CDG+02] proposent d'ajouter une seconde table de routage dans Pastry, qui est moins optimisée mais plus robuste à l'insertion de Sybils. En cas d'échec de routage par la première table, la seconde table ainsi que des routes redondantes sont utilisées pour assurer le service. Cependant cette solution repose sur une entité de certification centralisée pour limiter le nombre de Sybils et empêcher leur localisation précise sur la DHT, ce qui est contraire au paradigme P2P. Dans [DLLKA05], les auteurs proposent une manière de limiter les effets d'une attaque Sybil dont le but est d'empêcher le routage des requêtes dans Chord en analysant les relations d'amorçage entre les pairs. Pour chaque contact utilisé, un pair doit ainsi connaître sa place dans l'arbre d'amorçage. Lors du routage, un pair interrogé ne renvoie pas le meilleur contact éligible comme dans Chord mais l'ensemble des contacts connus. Au prix d'un important surcoût, l'instigateur peut maîtriser à chaque étape le chemin amenant à la cible en évitant les nœuds suspicieux, car étant trop proches dans l'arbre d'amorçage. Ceci rend le routage plus robuste à l'attaque Sybil. Cependant, ces relations hors-ligne pré-supposées entre les pairs ne sont pas réalistes en pratique. Les DHTs existantes reposent sur des listes publiques de pairs à contacter pour rejoindre le réseau, ce qui n'est pas compatible avec ce mécanisme qui nécessite un lien privé entre les pairs.

L'article [SCDR04] propose une solution contre l'attaque Sybil visant à déconnecter un pair du réseau selon l'évaluation du degré de connectivité des pairs. L'hypothèse est que les Sybils participant à l'attaque ont un degré de connectivité supérieur aux pairs normaux. Un pair ne communique directement qu'avec les pairs avec lesquels il a établi un lien réciproque. Pour vérifier la connectivité, chaque nœud peut être contraint de donner ses listes de contacts sortant et entrant. Une vérification anonyme (utilisant des intermédiaires) de la connectivité des voisins d'un pair permet de savoir si ceux-ci sont trop connectés (suspicieux) et qu'ils annoncent bien le lien avec ce dernier. Cette solution empêche la corruption massive des tables de routage par les Sybils, mais ne contraint pas le placement de ceux-ci.

L'article [CKS+06] propose une solution contre la corruption des tables de routage par les Sybils et qui peut aboutir à l'isolement d'un pair. Leur solution est composée de trois éléments : une réinitialisation périodique des tables de routage, un changement périodique des identifiants et une limitation de la vitesse de mise à jour des tables de routage. Ces règles augmentent la dynamique des tables de routage rendant plus difficile le maintien des Sybils dans la table d'un pair particulier, ce qui tend à limiter la présence des Sybils dans une table de routage particulière à la hauteur de leur proportion dans le réseau. Cette approche souffre de plusieurs

limites, tout d'abord les changements périodiques d'identifiants aléatoires sont dépendants d'un serveur, ensuite la question de la pérennité des données stockées dans la DHT n'est pas évoquée malgré un changement des responsables, et enfin, le mécanisme engendre un surcoût de messages et une perte de performances significative.

Les contraintes structurelles peuvent donc limiter les effets de l'attaque au prix d'une perte d'efficacité du routage ou d'un surcoût de messages. Cependant, ces différentes solutions deviennent inefficaces quand le nombre de Sybils est trop élevé. En effet, elles ne limitent pas le nombre de Sybils pouvant rejoindre le système, et sont donc réactives, contrairement, par exemple, à des contrôles d'admission qui sont eux pro-actifs. Par ailleurs, ces différentes solutions se focalisent uniquement sur la robustesse du routage sous attaque Sybil et aux problèmes affectant le stockage des données comme l'attaque éclipse.

#### 2.2.3.2 Vérification des ressources

La vérification des ressources est fondée sur le principe que les différentes entités participant au réseau P2P disposent de certaines ressources équivalentes et qui peuvent être vérifiées pour chaque pair. Ainsi, une entité insérant trop de pairs dans le réseau ne pourra pas assumer la vérification pour chacun d'entre eux. Deux types de contraintes ont été élaborées, d'une part calculatoire et d'autre part sociale.

**Contrainte calculatoire**

Bien que préférant utiliser au final une autorité de certification centralisée, les auteurs de [CDG+02] mentionnent également une solution permettant de réduire le nombre de Sybils dans le réseau. Pour cela, les pairs doivent résoudre un puzzle calculatoire en générant une paire de clés dont l'empreinte de la clé publique par la fonction SHA-1 a les $p$ premiers bits égaux à 0. Le nombre d'opérations requises pour trouver cette paire est alors en $O(2^p)$ et l'empreinte de la clé publique par MD5 donne l'identifiant du pair. La limite de cette approche tient à l'hétérogénéité des pairs. Le calcul doit en effet pouvoir être validé par le pair le moins puissant du réseau, de ce fait $p$ ne peut être trop contraignant et alors, une entité disposant de machines rapides sera quand même en mesure de maintenir de nombreux Sybils dans le réseau, même si les identifiants sont invalidés périodiquement.

L'article [REMP07] décrit une autre manière collaborative de mettre en œuvre le contrôle d'accès. Les pairs décrivent ici un arbre de certification et leur place dans l'arborescence est définie selon l'ordre d'arrivée des pairs dans le réseau. Les pairs initiateurs du réseau forment le haut de l'arborescence et sont considérés comme fiables. Un nouvel arrivant doit faire valider son identifiant successivement par les pairs de la branche de l'arbre le reliant à la racine, en résolvant à chaque étape un puzzle calculatoire. Pour éviter qu'une entité ne gagne constamment des identifiants avec le temps, les certificats ne sont valables que temporairement (8 heures) et doivent être renouvelés. L'organisation initiale ne prévoit qu'une racine qui constitue un point critique contraire au paradigme P2P puisque chaque identifiant doit être validé en dernier lieu par ce pair. Une architecture avec plusieurs racines résout en partie ce problème mais introduit le risque d'une racine malveillante. Outre l'intérêt de limiter le nombre de Sybils, ces approches empêchent surtout les Sybils de choisir leur identifiant ce qui limite les risques d'attaques localisées. Cependant, ces solutions doivent être introduites dès le commencement du réseau et ne sauraient être appliquées à un réseau existant sans casser la rétro-compatibilité entre clients. Par définition, elles génèrent également un surcoût important pour les pairs.

#### Contrainte sociale

Une autre forme de vérification est basée sur les liens sociaux entre pairs. SybilGuard [YKGF06] détecte les attaques Sybil en demandant préalablement aux utilisateurs d'établir des liens de confiance entre eux. Ainsi les Sybils qui sont opérés par un seul utilisateur apparaissent comme fortement liés entre eux, mais peu avec la partie honnête du réseau P2P. Cette propriété est vérifiée localement par chaque pair en effectuant un parcours aléatoire du réseau. Les propriétés des graphes sociaux font que les chemins aléatoires de deux pairs honnêtes vont se croiser rapidement avec une grande probabilité. De plus, les chemins convergent dès lors qu'ils arrivent sur un pair depuis le même lien social. Deux pairs dont les liens sociaux se croisent avant un certain nombre de sauts peuvent alors communiquer. SybilLimit [YGKX08], des mêmes auteurs, améliore la proposition et limite le nombre de Sybils valide par attaque en O(log n). L'auteur de [LL08] décrit comment ces graphes sociaux peuvent ensuite être utilisés pour créer une DHT dont le routage est robuste à l'attaque Sybil mais la DHT proposée effectue le routage en un seul saut ce qui pose des problèmes de passage à l'échelle.

La principale limite de ces approches sociales est de ne pas contraindre le choix des identifiants. Ainsi, même avec un nombre réduit de Sybils, des attaques ciblées demeurent possibles. Par ailleurs, la détection est locale à chaque pair, ce qui fait que chaque Sybil est quand même faiblement connecté au réseau. L'établissement de liens sociaux a également un coût pour les utilisateurs qui doivent entrer manuellement ces relations, et s'échanger les clés de chiffrement hors du réseau. Sans cette coopération ou si les utilisateurs renseignent trop peu de liens, ces protections sont inefficaces. Enfin, ce mécanisme doit être implanté par l'ensemble des pairs pour être efficace ce qui limite son applicabilité à des réseaux déjà déployés et il engendre un surcoût important de messages pour vérifier les relations entre pairs.

### 2.2.3.3 Certification des identifiants

Une autre manière de limiter les attaques Sybil est de certifier les identifiants des pairs pouvant rejoindre le réseau et ainsi filtrer les Sybils au niveau du contrôle d'admission. Nous différencions pour cela les approches centralisées des approches distribuées.

#### Centralisée

Le papier [STY03] présente une architecture de contrôle d'admission au sein d'un groupe de pairs et compare pour cela différentes méthodes de chiffrement. Si l'attaque Sybil est mentionnée, le contrôle d'accès n'y est pas confronté. Les auteurs délèguent en effet à une première autorité de certification (CA) la génération d'un certificat de clé publique (PKC) utilisé ensuite pour réaliser le contrôle d'accès dans les sous-groupes.

Les auteurs de [FMR+09] visent à limiter l'insertion de nœuds à des positions ciblées de la DHT de Kademlia pour y corrompre l'indexation, notamment par une attaque éclipse. Ils introduisent pour cela un service de certification des identifiants (CA centralisé), attribuant un identifiant aléatoire et générant un certificat associé à la clé publique du pair et à son identifiant. L'autorité de certification génère également un identifiant authentifié en concaténant ces informations et un temps de validité. Cette nouvelle identité est ensuite vérifiée préalablement à toute communication entre deux pairs. Les mêmes auteurs étendent leurs travaux dans [AMRS08] en évaluant le coût de leur solution. Bien qu'efficace, l'utilisation d'un composant centralisé n'est cependant pas appropriée au paradigme P2P. Cette solution parait donc difficilement applicable à des réseaux existants.

#### Distribuée

L'article [DH06] présente une taxonomie des méthodes existantes pour assigner les identifiants des pairs et en quoi elles sont vulnérables à l'attaque Sybil. Il décrit ensuite une attribution des identifiants basée sur l'adresse IP et le port du pair. Chaque pair doit enregistrer son identifiant auprès d'un groupe de pairs sur la DHT qui est calculé de manière déterministe d'après l'empreinte de son préfixe IP. Ces pairs sont alors en mesure d'appliquer des règles de restriction en refusant l'enregistrement d'adresses IP trop représentées. Cette solution évite de plus les attaques ciblées en contraignant la position de chaque pair. Cette méthode est cependant très sensible au va-et-vient des pairs et nécessite un ré-enregistrement périodique des identifiants. Par ailleurs, la vérification des identifiants se fait en consultant le groupe de pairs responsable ce qui induit un coût important en terme de messages.

La protection la plus aboutie contre l'attaque Sybil [LMT08b] couple une autorité de certification distribuée à la solution SybilGuard, déjà présentée en section 2.2.3.2. SybilGuard permet ainsi de détecter les Sybils d'après leurs liens sociaux et l'autorité de certification permet de les révoquer. Le réseau possède une clé secrète distribuée sur l'ensemble des pairs en suivant un arbre de fragmentation. Un pair possède un couple de clé publique/privée et un morceau de la clé du réseau. Pour participer au réseau, il doit obtenir un certificat consistant à avoir sa clé publique signée par la clé du réseau, ce qui nécessite l'approbation d'un certain nombre de pairs qui est proportionnel à la taille du réseau. Son identifiant dans la topologie P2P étant l'empreinte imprévisible de ce certificat ce qui limite les attaques ciblées. Le fonctionnement de l'autorité de certification distribuée est présenté en détail dans [LMT09], notamment comment la répartition de la clé du réseau évolue dynamiquement avec ce dernier. La distribution de l'autorité de certification rend plus difficile un contrôle d'accès limitant les Sybils, le recours à SybilGuard est donc complémentaire. La procédure de révocation appliquée aux nœuds malveillants est décrite dans l'article [LMT08a]. Celle-ci consiste en l'établissement d'une preuve validée par un ensemble de pairs qui signe alors le certificat de révocation du pair malveillant. Une fois signé, il est ensuite stocké dans la DHT de manière déterministe afin d'être consultable par les autres pairs.

## Conclusion

Pour conclure ce chapitre, nous avons présenté les principaux problèmes de sécurité affectant les réseaux P2P, à savoir : les différentes formes de pollution et l'attaque Sybil. Cette dernière peut avoir de nombreuses applications parmi lesquelles la pollution mais aussi l'éclipse d'information dans le réseau ou la réalisation d'attaques de type DDoS. La pollution, associée à la diffusion de contenus illégaux et potentiellement dangereux, est aussi un problème critique. Des utilisateurs peuvent ainsi accéder de manière non intentionnelle à un contenu non désiré les heurtant, ou être supervisés à tort lors de l'accès à ce contenu non désiré. Ce problème crucial est peu traité dans la littérature et appelle, d'une part à la conception de méthodes de supervision robustes à la pollution, et d'autre part, à la conception de mécanismes permettant de détecter et d'éviter les fichiers pollués. La pollution ayant été principalement étudiée sur des réseaux aujourd'hui obsolètes, il manque de réelles connaissances quant aux formes actuelles de la pollution et à sa diffusion dans les réseaux populaires.

Nous avons également présenté dans ce chapitre les solutions proposées pour limiter ces problèmes de sécurité. Les tableaux 2.1 et 2.2 résument les propriétés de chaque solution, respectivement contre la pollution et contre l'attaque Sybil. Il apparaît que la lutte contre la pollution peut être considérée comme une extension de la lutte contre les attaques Sybil car aucune des protections proposées contre la pollution ne résiste à une pollution réalisée au coeur de la DHT

par des Sybils faisant une attaque ciblée sur la donnée à corrompre. Concernant l'attaque Sybil, parmi les nombreuses solutions proposées, nous avons constaté leur non applicabilité en pratique à des réseaux existants. Ainsi, certaines méthodes nécessitent que des règles soient utilisées dès l'amorçage du réseau (initialisation saine, politique d'attribution des identifiants, ...) ou que certaines primitives spécifiques à la protection soient ajoutées à tous les pairs à un moment donné. Or, les réseaux P2P actuels tels que KAD sont d'ores et déjà victimes de l'attaque Sybil et les contraintes fortes des solutions proposées empêchent leur application à KAD dont l'initialisation du réseau n'a pas été garantie, et dont les clients hétérogènes ne peuvent pas être mis à jour de manière synchrone. Ils doivent ainsi fournir une rétro-compatibilité avec des clients n'implantant pas encore le mécanisme.

Dans le chapitre 7, nous mettons en évidence la réalité des attaques dont les concepts ont été décrits ici. Nous montrerons notamment que le réseau KAD est largement affecté par une nouvelle forme de pollution du mécanisme d'indexation d'une part, et par l'insertion de pairs malveillants d'autre part. Nous proposons dans le chapitre 8 un moyen très efficace de détecter et de limiter ces attaques ciblées sur la DHT. Notre protection est complètement distribuée, parfaitement rétro-compatible avec les anciens clients et présente un surcoût négligeable.

| Protection | solution distribuée | attaques ciblées évitées | surcoût | retro-compatibilité | complexité de mise en œuvre |
|---|---|---|---|---|---|
| Contraintes structurelles | | | | | |
| [DLLKA05] | oui | non | élevé | non | moyenne |
| [SCDR04] | oui | non | élevé | non | moyenne |
| [CKS+06] | non | oui | élevé | non | moyenne |
| Contrainte calculatoire | | | | | |
| [CDG+02] | oui | oui | élevé | non | élevée |
| [REMP07] | oui | oui | élevé | non | élevée |
| Contrainte sociale | | | | | |
| [YKGF06] | oui | non | élevé | non | moyenne |
| Certification centralisée | | | | | |
| [STY03] | non | oui | faible | non | moyenne |
| [FMR+09] | non | oui | faible | non | faible |
| Certification distribuée | | | | | |
| [DH06] | oui | oui | élevé | non | faible |
| [LMT08b] | oui | oui | élevé | non | élevée |

TABLE 2.1 – Comparaison des méthodes de protection contre l'attaque Sybil

Le chapitre suivant présente le réseau P2P KAD et les travaux relatifs. KAD est à la fois l'objet de notre étude en tant que DHT et l'environnement de nos expériences en tant que réseau P2P largement déployé.

| Protection | solution distribuée | détection fausses entrées | détection corruption index | résistance aux Sybils | surcoût | complexité | retro-compatibilité |
|---|---|---|---|---|---|---|---|
| [LNR05] | non | oui | non | non | faible | faible | oui |
| [DHX09] | oui | oui | oui | non | élevé | moyenne | non |
| [SRRS09] | oui | oui | oui | non | élevé | élevée | non |
| [CA07] | oui | partielle | partielle | non | moyen | moyenne | oui |
| [LMV08] | oui | non | partielle | non | faible | faible | oui |
| [LMSW10] | oui | oui | partielle | non | faible | faible | oui |

TABLE 2.2 – Comparaison des méthodes de protection contre la pollution

# Chapitre 3

# Le réseau pair à pair KAD

**Sommaire**



## Introduction

Nous utilisons le réseau pair à pair KAD comme environnement expérimental pour valider chacune des contributions de la thèse. Plusieurs raisons ont motivé notre choix pour ce réseau. Tout d'abord, KAD est un réseau P2P structuré basé sur le protocole de routage Kademlia [MM02]. L'étude des différentes architectures P2P réalisée dans le premier chapitre a montré la supériorité des DHT pour mettre en œuvre un réseau P2P, et parmi les différentes architectures, Kademlia est la plus efficace. Le fait que KAD repose sur Kademlia permet d'une part de travailler à partir d'une base théorique bien établie et d'autre part de pouvoir aisément généraliser les résultats obtenus sur KAD à d'autres DHTs. Ensuite, le réseau P2P KAD est un des réseaux P2P les plus largement déployés à ce jour. Il propose un service complet de partage de fichiers à des millions d'utilisateurs, incluant l'indexation distribuée et la recherche des fichiers. De ce fait, KAD est naturellement confronté aux problèmes affectant les réseaux réels tels que les problèmes de sécurité ou encore la contrainte de maintenir la rétro-compatibilité entre les clients au fil des versions. En outre, les solutions proposées et validées sur KAD peuvent profiter directement à des millions d'utilisateurs, ce qui est de surcroit motivant. Enfin, le code source des deux principaux clients (eMule [13] et aMule [14]) implantant KAD est libre ce qui en facilite la compréhension et la modification. L'accès au code source permet de comprendre les spécifications du protocole KAD, d'instrumenter et de modifier les clients. De nombreuses modifications du client officiel proposées par la communauté existent à ce titre. Par ailleurs, le développement des principaux clients est toujours actif au travers de mises à jour au moins bi-annuelles.

13. http ://www.emule-project.net/
14. http ://www.amule.org/

| Type | Nomination | Objet | Identifiant (notation hexadécimale) |
|---|---|---|---|
| pair | KADID | - | `32FFF76959F6A7095347FB338B304330` |
| fichier | FILEID | avatar.avi | `D37C98517E79DDC1688E27D1FE849BE5` |
| mot-clé | KEYWORDID | avatar | `A35BC8A4D252ADB3A99A46A28B275DFB` |

TABLE 3.1 – Exemples d'identifiants utilisés dans KAD

Nous présentons dans ce chapitre le fonctionnement de KAD dont la compréhension est nécessaire pour la bonne lecture de ce manuscrit. Nous présentons également les différentes études académiques déjà réalisées sur ce réseau P2P.

## 3.1 Mise en œuvre de la DHT

Nous avons déjà présenté le fonctionnement générique de la DHT Kademlia dans la section 1.1.2.2. Nous considérons ici comment celle-ci est implantée pour constituer le réseau KAD et relevons notamment les changements apportés à la spécification d'origine. L'ensemble du fonctionnement de KAD a été décrit très précisément dans le rapport de Master de René Brunner [Bru06] qui constitue de loin la source la mieux documentée sur le protocole de KAD.

### 3.1.1 Espace d'adressage

La première différence avec Kademlia réside dans l'espace d'adressage utilisé. Kademlia est en effet décrit comme utilisant un espace d'adressage de 160 bits alors que les identifiants de KAD ont 128 bits. Trois types d'entités partagent l'espace d'adressage à savoir : les pairs, les fichiers et les mots-clés. Chaque nœud de KAD possède un identifiant définissant sa position dans la table de hachage distribuée. Tout comme dans Kademlia, la distance entre deux identifiants est donnée par la métrique XOR. L'identifiant d'un pair est choisi aléatoirement par ce dernier à la première exécution du client puis est gardé à travers les différentes sessions. Le service de partage de fichiers réalisé par KAD nécessite d'enregistrer des informations sur des fichiers et des mots-clés dans la DHT. Chaque fichier possède ainsi un identifiant obtenu par hachage de son contenu par la fonction MD4, contrairement à Kademlia qui préconise l'utilisation de la fonction de hachage SHA-1. Les tailles d'identifiants différentes entre KAD et Kademlia proviennent de ce changement de fonction. De même, l'identifiant d'un mot-clé est obtenu par hachage des caractères ASCII suivant la même fonction. Ces trois types d'identifiants sont régulièrement appelés « KADID », cependant pour éviter toute confusion dans la suite de ce manuscrit, « KADID » fera référence à l'identifiant d'un pair et nous utiliserons les termes « FILEID » et « KEYWORDID » pour faire respectivement référence à l'identifiant d'un fichier et d'un mot-clé. Le tableau 3.1 présente un exemple de chaque identifiant.

### 3.1.2 Connexion au réseau P2P

#### 3.1.2.1 Phase d'amorçage

La phase d'amorçage consiste à découvrir progressivement les autres pairs connectés au réseau. KAD réalise cette étape de manière complètement distribuée. Pour rejoindre le réseau, un pair peut soit renseigner directement l'adresse IP et le port d'un pair déjà connecté au réseau, soit obtenir et utiliser un fichier contenant une liste de pairs potentiellement connectés. Le pair

entrant envoie alors des requêtes au(x) pair(s) connecté(s) afin de recevoir de nouveaux contacts qui sont interrogés à leur tour jusqu'à ce que le nombre de contacts obtenus peuple les premiers niveaux de la table de routage. Les requêtes d'amorçage utilisées sont des requêtes de bootstrap (`KADEMLIA2_BOOTSTRAP_REQ`), lorsqu'un pair reçoit une requête de ce type, il envoie une réponse (`KADEMLIA2_BOOTSTRAP_RES`) comportant les informations (KADID, adresse IP, port) de 20 pairs sélectionnés aléatoirement dans sa table de routage.

#### 3.1.2.2 Connectivité

Seuls les pairs directement contactables peuvent recevoir ces requêtes d'amorçage. En effet, deux équipements peuvent limiter la connectivité des pairs dans KAD à savoir les pare-feux et les routeurs effectuant une translation d'adresse. Afin d'être parfaitement fonctionnel, un client KAD doit pouvoir recevoir des paquets entrant sur un port UDP et TCP préalablement définis. Le port UDP est utilisé pour recevoir tous les messages de signalisation liés au protocole KAD. Le port TCP, quant à lui, est utilisé pour recevoir les demandes de connexion liées aux transferts de fichiers. Le pare-feu doit ainsi être configuré afin de laisser passer les paquets entrant sur les deux ports nécessaires, et un routeur réalisant une translation d'adresse doit être configuré afin de rediriger les ports utilisés par le client vers la machine concernée. Les pairs à la connectivité limitée peuvent toutefois utiliser l'application et partager leurs fichiers. Pour cela, ils choisissent un autre pair dans le réseau, appelé « buddy », dont la connectivité n'est pas limitée et établissent une connexion TCP permanente avec ce dernier. Ce pair sert alors de relais en recevant les requêtes entrantes pour le compte du pair dont il a la charge. Toutefois, un pair non directement contactable ne peut participer à la DHT (ni pour l'indexation, ni pour le routage), cette charge est uniquement dévolue aux pairs pouvant recevoir directement des requêtes. Ces problèmes de connectivité des pairs ont fait l'objet d'une étude précise dans le cadre de la DHT de BitTorrent [JOK09] et dont les résultats sont généralisables à KAD.

### 3.1.3 Routage

#### 3.1.3.1 Table de routage

La table de routage de KAD reprend le principe de Kademlia. La table de routage de chaque pair est constituée de différents groupes de $k$ contacts (appellés « k-bucket ») organisés en arbre de manière à ce que chaque niveau $i$ détienne un groupe de pairs dont la distance est comprise entre $2^{128-i}$ et $2^{128-(i+1)}$ par rapport au KADID du pair courant. Dans Kademlia, chaque k-bucket regroupe des contacts partageant exactement un certain nombre de bits de poids fort avec le pair courant. Par conséquent, un groupe de k-contacts couvre une zone d'autant plus petite qu'il est proche du pair. Cette organisation permet d'obtenir un routage en O(log n). Un k-bucket de niveau $i$ peut se diviser en deux si celui-ci est rempli et se trouve être le plus proche du pair courant. Parmi les deux k-buckets résultant de la scission, le premier ne pourra plus se subdiviser et ses contacts partagent exactement les $i-1$ bits de poids fort du pair courant, le second pourra encore se subdiviser et ses contacts partagent au moins $i$ bits de poids fort avec le pair courant. Dans KAD, la valeur définie pour $k$ est de 10 contacts.

Cependant, l'étude du code source de KAD nous a permis d'identifier plusieurs différences d'implantation. La plus évidente concerne les niveaux supérieurs de la table de routage. En effet, la table de routage de KAD n'est pas l'arbre binaire parfait présenté dans Kademlia comme le montre la figure 3.1. Tout d'abord, toutes les feuilles (k-bucket) peuvent se subdiviser jusqu'au niveau 4 sans restriction. Quand Kademlia dispose d'un seul k-bucket pour couvrir la moitié de la DHT opposée au pair courant selon la métrique XOR (ce qui correspond à la première division

de l'arbre sur le schéma 3.1) , KAD en dispose de 8, lui permettant de trouver de meilleurs contacts vers les destinations les plus éloignées. Plus généralement, les 16 premiers k-buckets (160 contacts) représentés sur la figure 3.1 couvrent la majeure partie de la table de routage et sont utilisés pour effectuer le premier saut lors de la localisation d'un identifiant.

Ensuite, KAD introduit une notion supplémentaire pour peupler sa table de routage. En plus de la notion de niveau commune à Kademlia, KAD utilise la notion d' « index ». L'index est une distance XOR calculée en considérant des bits supplémentaires lorqu'un k-bucket de Kademlia doit être créé. Kademlia place ainsi dans le $n^{eme}$ k-bucket les contacts dont le bit $n$ diffère avec le pair courant alors que KAD crée plusieurs k-buckets pour organiser ces contacts comme qu'illustré par la figure 3.2. Ainsi, pour un niveau $n$ donné, les pairs dont l'index est inférieur à 5 (ce qui signifie qu'étant donné les premiers 4 bits suivant le bit $n$ courant : KADID XOR ContactID <= 5) peuvent continuer à se diviser pour former des k-buckets.

KAD stocke donc cinq 10-bucket par niveau au delà du niveau 4. Le nombre de contacts maximum dans la table de routage est donc de $(11 + 5 * (128 - 4) + 1) * 10 = 6320$ contacts alors qu'avec des 10-buckets, Kademlia comporterait au maximum $128 * 10 = 1280$ contacts. En pratique, ce nombre n'est jamais atteint car la taille réelle du réseau P2P limite la profondeur de la table de routage, très peu de contacts pouvant satisfaire les contraintes de proximité des niveaux inférieurs dont la division est alors improbable. La table de routage de KAD est donc sensiblement différente, et comporte plus de contacts que la table de routage de Kademlia mais, en contrepartie, peut router plus efficacement et est moins sensible au va-et-vient des pairs.

Pour limiter les effets du va-et-vient des pairs sur la cohérence de la table de routage, KAD utilise des requêtes permettant de vérifier si un contact est toujours connecté. La requête `KADEMLIA2_HELLO_REQ` et la réponse associée `KADEMLIA2_HELLO_RES` sont utilisées à cet effet. Un pair est ainsi capable de mesurer la durée de vie d'un contact à l'intérieur de sa propre session. L'algorithme de mise à jour de la table de routage favorise les contacts connectés depuis longtemps pour améliorer la stabilité du réseau.

#### 3.1.3.2 Localisation des ressources

Étant donné un identifiant, le routage consiste à localiser le pair lui même, s'il s'agit d'un KADID, ou les pairs les plus proches de l'identifiant, s'il s'agit de mots-clés ou fichiers. Le routage de KAD est basé sur la métrique XOR. La première étape consiste à rechercher dans la table de routage du pair courant les 3 contacts les plus proches de la cible. Pour chaque contact est alors émise une requête de localisation (`KADEMLIA2_REQ`) requèrant des contacts plus proches. Cette requête spécifie le KADID cible et le nombre $n$ de contacts que l'on souhaite obtenir en réponse ($0 \leq n \leq 31$), par défaut 4 contacts sont requis. A la réception de la requête, un pair envoie la réponse associée (`KADEMLIA2_RES`) comportant les informations (KADID, adresse IP, port) des $n$ contacts de sa table de routage les plus proches du KADID cible spécifié. Parmi les contacts reçus, les trois meilleurs sont à nouveau interrogés par le même procédé. Le routage de KAD se fait donc de manière itérative avec plusieurs requêtes envoyées en parallèle comme présenté par la figure 3.3, ce qui assure la fiabilité du routage en cas d'intermédiaires malveillants ou défaillants. Le parallélisme des requêtes n'est pas strict dans le sens où dès qu'un nouveau contact réduisant la distance à la cible est trouvé parmi les réponses, celui-ci est interrogé. KAD n'attend donc pas la réception de toutes les réponses émises à l'étape $n$ pour commencer l'étape $n + 1$. Tant que des contacts plus proches sont retournés, le routage continue de progresser. La condition d'arrêt est la non réception de contacts plus proches pendant les 3 dernières secondes, signifiant que les derniers pairs interrogés ne sont pas en mesure de fournir d'autres pairs se rapprochant de la cible.

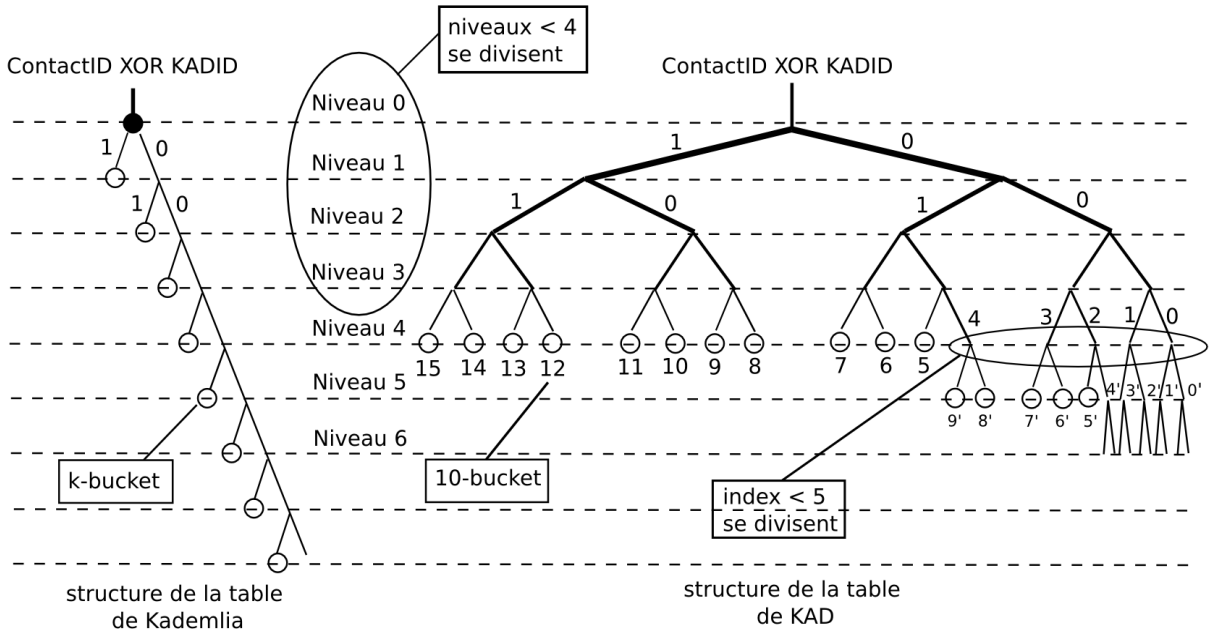


FIGURE 3.1 – Tables de routage de Kademlia et de KAD comparées

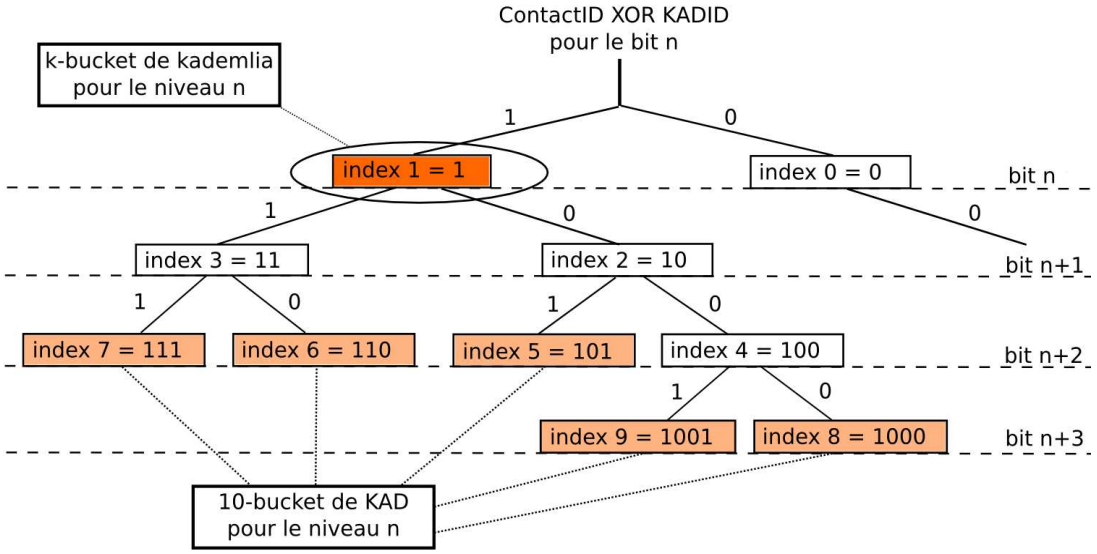


FIGURE 3.2 – Calcul des index pour un niveau de la table de routage de KAD

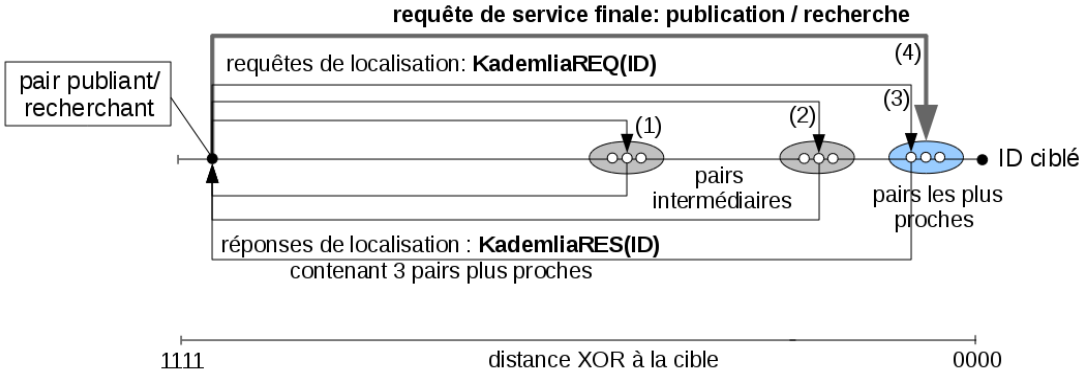


FIGURE 3.3 – Routage itératif dans KAD

### 3.1.4 Mécanisme de double indexation

En tant que support de partage de fichiers, la fonction principale de la DHT de KAD est d'offrir un service de recherche pouvant trouver, étant donné un ensemble de mots-clés, les fichiers s'y rapportant, mais également, étant donné un fichier, les sources de celui-ci. Pour ce faire, KAD utilise la DHT pour mettre en œuvre un mécanisme d'indexation à deux niveaux permettant d'indexer des mots clés et des fichiers tel que présenté par la figure 3.4. Chaque mot-clé composant le nom d'un fichier est ainsi associé à ce dernier. Dans l'exemple, les mots-clés « matrix » et « revolution » sont ainsi associés au fichier vidéo. La publication permettant ce lien a pour clé l'identifiant du mot-clé (KEYWORDID=11000 ou 00010) et la valeur indexée est l'identifiant du fichier (FILEID=00111). De même, un fichier est associé au pair le partageant, la clé est alors l'identifiant du fichier (FILEID=00111) et la valeur stockée, l'identifiant du pair publiant (KADID=11111). Les messages échangés entre pairs sont représentés par les flèches droites et l'indexation créée dans la DHT par les flèches courbées.

#### 3.1.4.1 Partage d'un fichier

Lorsqu'un fichier est partagé, son contenu ainsi que chacun des mots clés du titre sont hachés par une fonction MD4. Les identifiants ainsi générés sont publiés sur le réseau comme présenté par la figure 3.5, ceux-ci sont volontairement simplifiés par une valeur décimale. Les pairs potentiellement responsables de l'indexation d'un fichier ou d'un mot clé sont ceux dont le KADID est assez proche de l'identifiant publié. Cette distance, appelée « zone de tolérance », est définie par les 8 premiers bits de l'identifiant qui doivent être communs. Le mécanisme d'indexation à deux niveaux permet ensuite de retrouver un fichier particulier étant donné un ensemble de mots clés. Pour publier un fichier, deux types de requêtes sont nécessaires :

- les requêtes `KADEMLIA2_PUBLISH_KEY_REQ` sont envoyées vers l'identifiant des mots clés et associent un mot clé avec un fichier ;
- les requêtes `KADEMLIA2_PUBLISH_SOURCE_REQ` sont envoyées vers l'identifiant du fichier et associent un fichier avec une source (un pair le partageant).

Ainsi, la réalisation de services (publication ou recherche) se fait en deux étapes. La première consiste à chercher des pairs dans la zone de tolérance souhaitée. Ceci est réalisé par la fonction de routage précédemment décrite. Une fois la zone de tolérance atteinte, des requêtes spécifiques au service demandé sont envoyées, comme par exemple les requêtes de publication décrites ci-dessus. Afin de maintenir l'information malgré le va-et-vient des pairs, deux mécanismes sont

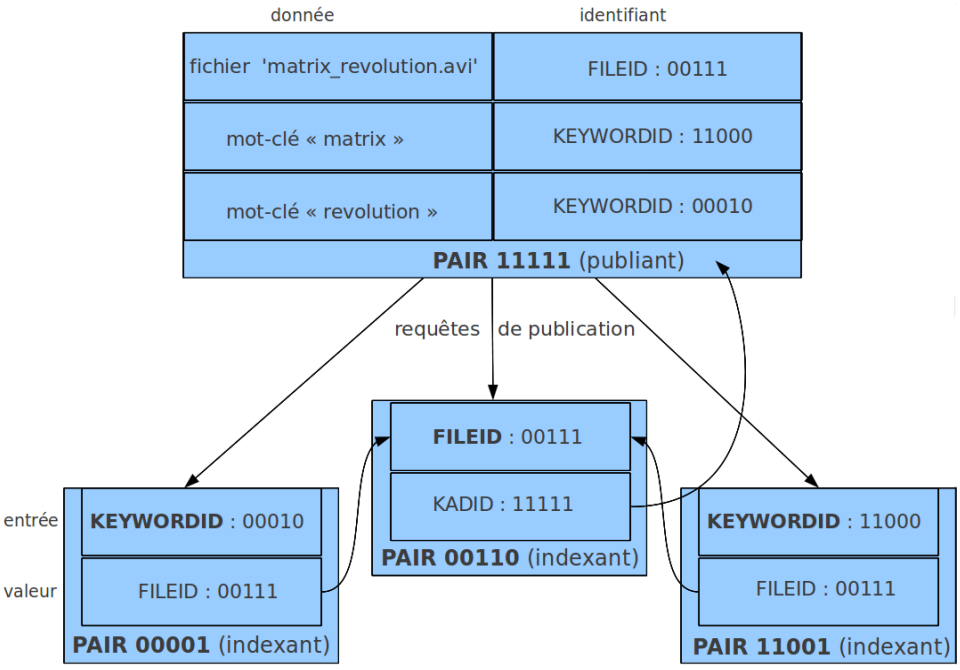


FIGURE 3.4 – Liens établis par l'indexation à deux niveaux, inspiré de [Bru06]

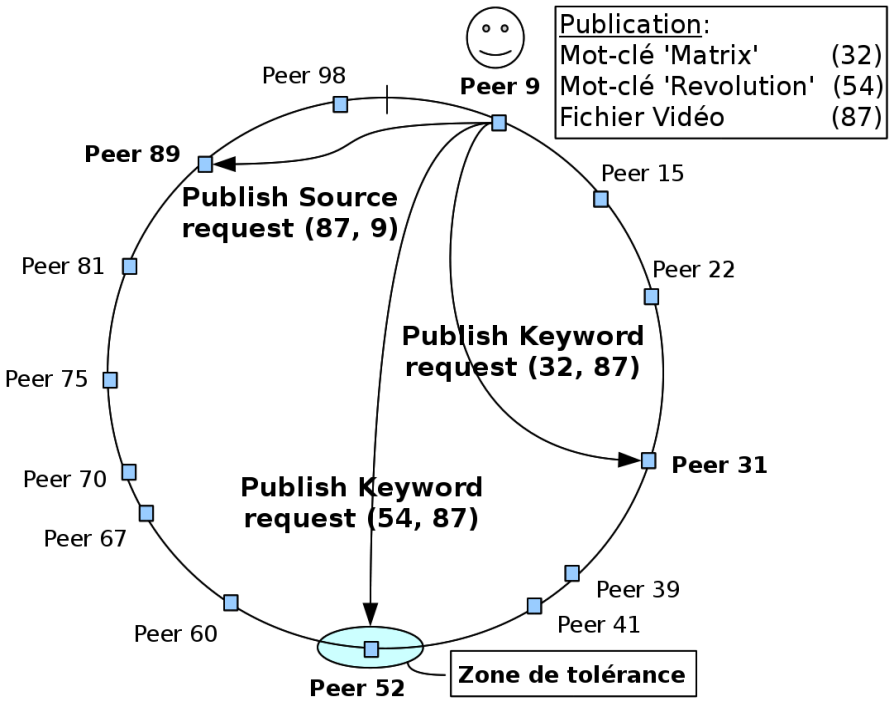


FIGURE 3.5 – Publication d'un fichier sur KAD

utilisés. D'une part, chaque donnée publiée est répliquée sur 10 pairs. Tant que 10 pairs n'ont pas acquitté les requêtes de publication, le processus continue. D'autre part, la publication des mots-clés est rafraîchie périodiquement toutes les 24 heures et celle des fichiers toutes les 5 heures. La différence entre les fréquences de rafraîchissement est due au fait qu'étant donné un fichier, tous les pairs qui le partagent vont réaliser la publication des mêmes mots-clés et ainsi émettre des informations similaires alors que la publication d'un fichier est spécifique à chaque pair qui se déclare comme source de celui-ci.

Afin d'économiser des ressources, une publication de mot-clé peut contenir jusqu'à 50 fichiers. Par exemple, il y a de fortes chances de retrouver le nom de l'interprète parmi le titre des musiques d'un album. Le nom ne sera ainsi publié qu'une seule fois avec l'ensemble des fichiers de l'album. Par ailleurs, pour chaque fichier, des informations supplémentaires sont publiées telles que le nom complet du fichier et sa taille.

#### 3.1.4.2 Téléchargement d'un fichier

Lorsqu'un utilisateur recherche un fichier, il renseigne tout d'abord un mot-clé dont le client KAD calcule l'identifiant par la fonction MD4. Cet identifiant est utilisé pour interroger les pairs de la DHT indexant ce mot-clé grâce aux requêtes `KADEMLIA2_SEARCH_KEY_REQ`, qui retournent alors la liste des fichiers publiés avec ce mot clé. En cas de succès, l'émetteur reçoit des informations concernant les différents fichiers correspondant aux critères de recherche. La sélection d'un fichier parmi ceux trouvés déclenche alors la recherche des sources. Celle-ci est réalisée en émettant des requêtes `KADEMLIA2_SEARCH_SOURCE_REQ` vers l'identifiant du fichier choisi. L'émetteur reçoit alors une liste de pairs pouvant être contactés pour télécharger tout ou partie du fichier et initie une connexion TCP vers chacun d'eux. Nous ne détaillerons pas la procédure de téléchargement du fichier une fois les pairs mis en relation car celle-ci sort du contexte de notre étude.

## 3.2 Expérimentations sur le réseau KAD

Le réseau P2P KAD, étant une des rares DHT déployée à grande échelle, a fait l'objet de nombreuses études académiques. Bien que mis en service depuis 2003, la majeure partie des publications sur KAD sont plus récentes et ce réseau P2P a gagné l'intérêt de la communauté scientifique à partir de 2007. Contrairement à Overnet dont le code source des clients n'était pas disponible, les clients libres de KAD ont facilité l'investigation du réseau en évitant le travail de rétro-ingénierie. Plusieurs types d'études ont été réalisées. La majeure partie d'entre elles s'intéresse aux propriétés de la DHT de KAD et notamment à ses performances. D'autres études se focalisent davantage sur l'étude des pairs du réseau, ou encore sur une des failles de sécurité affectant la DHT. A ce titre, les travaux de thèse de Moritz Steiner [Ste08] présentent des études sur de nombreux aspects du réseau P2P KAD.

### 3.2.1 Supervision du réseau KAD

#### Mesures de la DHT

Les auteurs de [ZDL07] ont réalisé une des premières études complètes de la DHT de KAD en réalisant deux types de mesures : des mesures locales depuis un client eMule modifié connecté à KAD dont ils relèvent les paquets échangés et les modifications de la table de routage, et des mesures globales grâce à un explorateur composé de clients aMule allégés, exécutés sur PlanetLab, et explorant chacun une fraction de la DHT. Plusieurs résultats ont émergé des mesures locales.

Tout d'abord, l'activité de la table de routage est très importante lorsqu'un client rejoint le réseau puis, le nombre d'insertion/suppression par minute diminue et la taille de la table de routage atteint un nombre stable de 1400 contacts après quelques heures. Ceci est dû au fait que KAD privilégie les contacts stables (i.e. qui sont connectés depuis longtemps) mais cette information ne peut être obtenue que si le client lui-même est connecté depuis assez longtemps au réseau. Ensuite, la majeure partie du trafic généré par un client est due aux requêtes de publication (mot-clé ou fichier) qui sont six fois plus nombreuses que les recherches. Enfin, lorsqu'un contact est recherché dans la table de routage, le nombre moyen de bits communs entre le plus proche contact et la cible est de 10. Ceci s'explique par la structure de la table de routage des anciens clients qui comportait 20 pairs par K-Bucket et pouvait se diviser jusqu'au niveau 5. 640 contacts ($2^5 * 20$) étaient ainsi uniformément répartis sur la DHT alors que la table de routage actuelle présentée dans la figure 3.1 en comporte 160 ($2^4 * 10$). Cependant, ces observations sont obsolètes aujourd'hui, d'une part à cause des nombreuses améliorations apportées aux clients depuis 2007 et d'autre part, à cause de l'augmentation de la taille du réseau.

Plusieurs travaux vont plus loin que les mesures de la DHT et proposent des améliorations du protocole KAD paliant les différentes faiblesses constatées. Ainsi, Stutzbach et al. [SR06] étudient l'efficacité du processus de routage de KAD (en prenant en compte la structure de la table de routage, l'obsolescence des contacts stockés ainsi que l'algorithme d'émission de requêtes parallèles) à travers un modèle analytique et une validation par des mesures réelles. Les auteurs mettent ainsi en évidence l'importance des k-buckets qui permettent d'obtenir aléatoirement quelques bits supplémentaires améliorés à chaque étape tout en offrant davantage de robustesse au va et vient des pairs. Ils valident aussi théoriquement la justesse de certains paramètres d'implantation d'eMule (10 contacts par bucket, 3 requêtes parallèles) mais soulignent que l'algorithme de publication peut être amélioré en limitant le nombre de réplicats pour en privilégier de meilleurs. L'article [SCB09] propose une étude similaire en prenant en compte d'autres paramètres tels que le nombre de contacts demandés à chaque requête de localisation ou encore la condition d'arrêt du processus de localisation (3 secondes sans nouveau meilleur contact). En mesurant les paramètres réels du réseau tels que le taux de contacts morts (0.31), le nombre de sauts par recherche (entre 3 et 4) et le délai d'aller-retour des messages (inférieur à 1 seconde), les auteurs proposent une nouvelle stratégie capable de réduire la latence de la recherche sans surcoût. Pour cela, ils suggèrent de réduire le temps de stabilisation nécessaire (à 0.5 seconde) ce qui réduit la durée de réalisation du service, en émettant des requêtes de service dès qu'un pair dans la zone de tolérance est trouvé. Cependant, pour que les informations publiées restent proches de leur clé, cette amélioration ne peut être appliquée qu'au processus de recherche.

Les mêmes auteurs étudient également dans [SEENB07] la charge induite par les requêtes de publication sur les pairs grâce à l'injection de sondes dans le réseau. Ils mettent ainsi en évidence que les mots-clés les plus publiés (jusqu'à un ordre de grandeur) sont des mots de liaison (stopwords) tels que « the » ou « and », qui ne sont pas discriminants lors de la recherche de contenus. La majeure partie de la charge de KAD étant induite par la publication de mots-clés, exclure ces mots-clés inutiles permet de réduire la charge sans désavantage notable. L'article [CB07] s'intéresse également au mécanisme d'indexation de KAD. Il applique en particulier la théorie de la fiabilité d'un système d'information pour modéliser comment le va et vient des pairs impacte la fiabilité du stockage dans la DHT et propose une nouvelle politique de publication offrant une fiabilité de stockage équivalente à moindre coût. La nouvelle politique est basée sur une re-publication désynchronisée des réplicats et sur l'inspection périodique des nœuds déjà responsables de la clé, la fréquence des vérifications étant inversement proportionnelle à la disponibilité passée du nœud.

Enfin, deux articles soulignent certains dysfonctionnements de KAD. L'article [KCTHK09] mesure que seuls 18% des réplicats sont contactés lors d'une recherche suivant immédiatement une publication. La probabilité qu'un réplicat soit trouvé dépend directement de sa distance à la cible. Ainsi, sur 10 réplicats, les deux réplicats les plus proches sont retrouvés dans 90% des cas mais ce taux décroît très vite à 20% à partir du cinquième. Ces mauvaises performances de la recherche associées au va et vient des pairs rend celle-ci très aléatoire pour les informations rares, tout en gâchant de nombreuses ressources. Cependant, des mesures montrent que les tables de routage de voisins proches d'une clé sont très cohérentes. Le problème vient du processus de routage qui excelle à trouver les 2-3 nœuds les plus proches d'une clé, ces derniers étant hautement répétés dans les réponses reçues lors du dernier saut et de ce fait, les autres nœuds constituant les 10 réplicats sont sensiblement plus loin de la clé et donc difficilement retrouvables. Qui plus est, les contacts les plus proches retournés sont souvent morts, car, dans KAD, les contacts des niveaux inférieurs de la table de routage sont moins fréquemment vérifiés. Les auteurs proposent une nouvelle méthode de localisation rétro-compatible et obtiennent 90% de réplicats communs entre la publication et la recherche consécutive d'une clé ce qui améliore le taux de succès d'une recherche de 65% à 95 % après 20 heures. Pour cela, le pair le plus proche trouvé n'est pas interrogé normalement (2 contacts demandés) mais avec 20 contacts. Cette modification assure une recherche robuste au prix d'un surcoût de messages (50%) durant la phase de recherche et d'une latence plus importante. Pour réduire cette latence, les auteurs proposent tout comme [SCB09] de confondre la phase de localisation et de recherche en ajoutant un bit supplémentaire signalant si le pair répondant à une requête de routage indexe lui-même cette clé.

Yu et al. s'intéressent dans [YL09] à la disponibilité et à la stabilité des contacts présents dans les tables de routage. Les tables de contacts des pairs sont parcourues grâce à un explorateur. Ils constatent que les pairs sont bien distribués à l'échelle mondiale dans plus de 100 pays dont les principaux sont l'Espagne, la Chine, l'Italie et la France. Les auteurs relèvent que seuls 65% des pairs obtenus sont contactables par le protocole KAD ce qui est majoritairement dû à des problèmes de connectivité entrante (pare-feu, NAT). Ils notent également que de nombreux identifiants sont répétés dans le réseau alors qu'ils devraient être uniques. Le cas des répétitions est étudié en détail dans l'article [YFX+09]. Il est montré que les répétitions d'identifiants représentent 20% des pairs et 4.5% des pairs actifs, la majeure partie des contacts inactifs sont en réalité déconnectés et quelques uns ne répondent volontairement pas aux requêtes. D'après l'étude, la majorité des pairs actifs dont l'ID est répété semble se comporter normalement en suivant le protocole de KAD. Une attaque pourrait utiliser la répétition des identifiants en ciblant les pairs responsables d'un contenu particulier pour les rendre plus difficiles à localiser. Cependant, les auteurs n'étudient ni la réalité de cette attaque en contactant les pairs, ni la corrélation potentielle entre les identifiants répétés et les contenus réels stockés dans la DHT qui aurait pu mettre en évidence les attaques.

### Mesures des pairs

D'autres études se sont intéressées aux caractéristiques des pairs de KAD. Ainsi les articles [SBEN07], [SENB07b] et [SENB09], présentent l'analyse des données d'un explorateur effectuant un recensement périodique des différents pairs du réseau. Ces explorations périodiques du réseau ont d'abord été réalisées pendant deux semaines, puis, l'exploration a été limitée à une seule zone sur une durée de six mois. Il apparaît que KAD est principalement utilisé en Europe (Espagne, France, Italie, Allemagne) et en Chine pour un nombre total d'utilisateurs relevé entre 3 et 4 millions. L'activité en Europe est cependant plus importante ce qui permet de constater une légère augmentation du nombre de pairs (+10%) en soirée (GMT+1). Les auteurs consta-

tent le va et vient fréquent des pairs et montrent que les nombre de départs ou d'arrivées entre deux explorations suit la loi binomiale négative. La durée d'une session suit quant à elle une distribution de Weibull, ce qui permet d'estimer l'espérance qu'un pair reste dans le réseau étant donné son temps de session. Plus un pair est connecté depuis longtemps, plus la probabilité qu'il reste connecté augmente, ce qui valide l'approche de KAD classant les contacts d'après leur âge perçu. L'exploration met également en évidence certains pairs changeant leur KADID entre deux sessions (10% des pairs sur un mois), notamment en Chine. Cet aléa rend plus difficile la caractérisation des utilisateurs car en plus de l'adresse IP, le KADID peut facilement changer. Dans [SENB07b] les auteurs constatent un taux élevé de départs permanents après une session unique (40%) ou quelques jours d'utilisation (70%). Une version plus étendue de ces travaux [SENB09] décrit l'algorithme de l'explorateur et une caractérisation des différences de comportements entre les pairs chinois et européens. Ces derniers présentent notamment une disponibilité journalière plus importante (40% sont connectés plus de 5 heures) mais celle-ci peut varier énormément dans le temps pour un même pair.

L'article [MRGS09], a déjà été présenté en section 1.2.1.5 pour son approche de supervision par l'injection de sondes dans le réseau KAD. Après une première phase d'exploration, chaque sonde est insérée à proximité d'un pair et connue uniquement de ce dernier. Chaque pair du réseau a ainsi sa sonde associée qui reçoit une copie des requêtes de service envoyées au pair supervisé. Cette visibilité limitée permet d'insérer un grand nombre de sondes sans perturber le réseau. Cependant, la précision de la supervision du trafic dépend de celle de l'explorateur découvrant les pairs et se montre limitée au delà d'une zone de 6 bits (32000 pairs), l'exploration prenant alors trop de temps pour fournir des données fiables. Parmi les résultats des données ainsi recueillies, il apparaît que 60% des fichiers et 95% des mots-clés ne sont jamais recherchés.

Enfin, [LMSW09] présente une étude comparative entre KAD et eDonkey s'intéressant à la distribution spatiale et temporelle de l'activité des pairs. Afin de caractériser les pairs, les auteurs s'intéressent uniquement aux requêtes de recherche par mot-clé, celles-ci étant émises directement par les pairs, d'une part en collectant toutes les recherches à destination d'un serveur eDonkey instrumenté, d'autre part en plaçant des pairs instrumentés autour de mots-clés populaires dans KAD ce qui permet d'intercepter les requêtes liées. Les méthodes de mesure étant différentes et la représentativité des sondes dans leur réseau respectif étant faible, il est difficile d'obtenir des résultats absolus d'après cette étude. Il semble néanmoins apparaître que le réseau KAD est relativement plus populaire en Europe.

Il ressort de ces différentes études que les pairs et la DHT ont largement été étudiés en tant que tels, mais non leurs relations aux contenus partagés dans le réseau. Plus généralement, peu de travaux parmi ceux précédemment réalisés sur KAD s'intéressent à l'activité des accès aux contenus stockés au sein du réseau KAD.

### 3.2.2 Vulnérabilités du réseau KAD

#### Attaque Sybil

L'attaque Sybil, telle que l'a décrite Douceur [Dou02], consiste à créer un grand nombre de faux pairs appelés les "Sybils" pour augmenter l'influence d'une entité sur la DHT, l'attaque Sybil étant d'autant plus efficace que le nombre de Sybils créés est important. L'article [CDG+02] ajoutait que placer les Sybils de manière précise au sein de la DHT permet de prendre le contrôle de certaines clés. Cependant, la vulnérabilité des DHT à cette attaque restait théorique. Steiner et al [SENB07a] furent les premiers à réaliser une attaque Sybil réelle sur KAD permettant de contrôler une partie du réseau et ont ainsi montré que KAD pouvait être gravement

affecté par cette attaque avec peu de ressources. L'attaque nécessite deux étapes. Une première étape d'exploration parcourt le réseau P2P en envoyant de manière systématique de nombreuses requêtes de découverte de pairs, afin de connaître le plus de contacts possibles. Quand tous les pairs du réseau ou d'une zone particulière ont été découverts, la seconde étape consiste à polluer leur table de routage en annonçant les Sybils comme de nouveaux pairs par l'envoi de requêtes Hello. Steiner et al ont ainsi injecté $2^{16}$ Sybils à partir de la même machine physique dans une zone couvrant 1/256ème de KAD. Dès qu'un Sybil reçoit une requête de routage, il détourne le routage en annonçant d'autres Sybils plus proches de la cible afin d'attirer la requête de service finale. Les Sybils ainsi insérés peuvent intercepter la quasi-totalité des requêtes de publication et de recherche à l'intérieur de cette zone et sont ainsi capables de superviser ou manipuler l'indexation des fichiers. Les Sybils ont ainsi relevé 1.5 millions d'identifiants de fichiers et 42000 de mots-clés. L'activité de la zone génère 10000 requêtes de publication par minute pour 1000 recherches. Au delà de la supervision, une application possible expérimentée dans [SENB07a] est une attaque éclipse faisant disparaître certains mots clés du réseau en feignant d'indexer toutes les requêtes de publication pour ces mots clés et en niant ensuite les requêtes de recherche de contenu. Les auteurs ont également réalisé une attaque localisée sur les mots-clés « the » et « dreirad » en plaçant des Sybils plus proches de leur identifiant que les pairs légitimes et ont ainsi montré que 32 Sybils suffisaient à éclipser le mot-clé et 8 Sybils à le superviser.

**Déni de service**

Les failles de sécurité de KAD peuvent conduire à des attaques de déni de service, contre le réseau lui même d'une part, ou contre des entités extérieures d'autre part. Les articles [KLR09] et [WTCT+08] décrivent deux attaques différentes empêchant l'accès aux services de KAD. Dans [KLR09], les auteurs constatent que les contre-mesures insérées dans les derniers clients KAD limitant le nombre d'occurrences d'une même adresse IP dans une table de routage empêchent l'insertion massive de Sybils. Ils exploitent alors le mécanisme de localisation des ressources de KAD pour réaliser une attaque de type éclipse sur les contenus. Pour cela, seul un Sybil dont l'identifiant est proche de la ressource ciblée est inséré dans les tables de routage des pairs entourant la cible (18 bits communs), et qui sont préalablement découverts par une exploration ciblée. Quand ce Sybil est interrogé par une requête de localisation, il répond en générant l'identifiant d'autres Sybils plus proches, qui sont à leur tour interrogés. Lorsque 10 Sybils ont été interrogés par un même pair, ils sont en mesure de capturer les réplicats du processus de publication. Une seconde stratégie permet un déni de service plus général affectant le processus de localisation du réseau qui est utilisé pour tous les services (publication, recherche, ...). L'attaque génère pour chaque requête de localisation interceptée un Sybil toujours un peu plus proche de la cible. De ce fait, le processus de routage continue indéfiniment car des contacts plus proches sont toujours trouvés, et la condition de 3 secondes sans nouveau contact stoppant la phase de localisation n'est jamais validée. Le processus est stoppé quand le délai maximal alloué au service est atteint, sans qu'une requête de service n'ait été envoyée. Le point critique de cette attaque est la visibilité du premier Sybil qui doit être trouvé pour que l'attaque fonctionne. Dans l'article [WTCT+08], les auteurs exploitent une autre vulnérabilité dans l'enregistrement des contacts dans la table de routage de KAD qui autorise un pair à usurper l'identité d'un autre. Ce manque de vérification sur les identités des pairs permet à un pair malveillant de corrompre les références de contacts dans une table de routage en écrasant les informations (adresse IP, port) par celles de leur choix, par la simple annonce d'une requête *Hello_ REQ* spécifiant le KADID ciblé. Associé à un explorateur, cette faiblesse exploitée à grande échelle permet de partitionner le réseau en réorganisant les liens entre les pairs. L'application de cette attaque réalisée à grande échelle

montre qu'elle est efficace et consomme peu de bande passante.

Le réseau KAD peut également être utilisé pour lancer une attaque vers des services extérieurs, notamment des dénis de service distribués (DDoS). L'article [ZDL07] a ainsi découvert un problème de sécurité en analysant les pairs découverts lors d'une exploration. L'explorateur a ainsi mis en évidence de nombreux contacts utilisant le port UDP 53, normalement utilisé pour le service DNS. Ces contacts font naturellement l'objet de requêtes de type `KADEMLIA2_HELLO_REQ` pour vérifier leur état ; or, le fonctionnement d'un serveur DNS fait que celui-ci renvoie le premier octet d'un message incompris (venant ici de KAD) mais ce retour est interprété comme un message Hello légitime (en fait le sien partiellement renvoyé) par le client initial. Celui-ci envoie alors un message `KADEMLIA2_HELLO_RES`, lui aussi partiellement renvoyé par le serveur DNS et acquittant ainsi la requête initiale du pair. Par ce procédé, les faux contacts pointant vers un serveur DNS sont maintenus en vie indéfiniment et se propagent dans le réseau. Cette faille est utilisée pour lancer une attaque de déni de service distribué sur des serveurs DNS. Elle a depuis été corrigée, les clients KAD interdisant toute communication vers le port DNS et contrôlant davantage les échanges de messages. D'autres stratégies de déni de service sont présentées dans [LC08]. La première, appelée attaque asymétrique, consiste à usurper l'adresse IP de la victime tout en émettant des requêtes de type `BOOTSTRAP_REQ` de taille 2KB dans le réseau. Les réponses `BOOTSTRAP_RES` générées contenant les contacts étant 260 fois plus volumineuses, elles permettent de saturer le lien de la victime. La seconde, appelée attaque par réflexion des tables de routage, consiste à annoncer de nombreux faux pairs dans le réseau avec l'adresse IP de la victime pour y rediriger le trafic P2P. La dernière méthode, appelée attaque par l'indexation, consiste à annoncer la victime comme source potentielle de fichiers en émettant de fausses `PUBLISH_SOURCE_REQ`. Les auteurs comparent l'efficacité de chaque approche. Les deux premières méthodes ne sont cependant plus possibles après l'introduction récente de protections dans les clients KAD, comme nous le verrons dans le chapitre 4. L'efficacité de l'attaque par indexation de la victime dépend de la popularité des fichiers corrompus. Il apparaît également que l'horaire des attaques influence grandement leur efficacité, celles-ci étant tributaires de l'activité du réseau pour générer le trafic. Sun et al. présentent dans [STR07] un autre détournement de KAD pour réaliser une attaque DDoS échappant aux nouvelles protections. Ils insèrent pour cela l'adresse IP de la victime dans les réponses des requêtes de routage et proposent plusieurs optimisations permettant de multiplier jusqu'à 40 fois le trafic d'attaque. Le protocole de KAD doit donc encore être amélioré contre ces exploitations malveillantes. Les auteurs proposent dans l'article [STR10] des règles allant dans ce sens, évitant le détournement des DHT pour le déni de service distribué, et en particulier celui de KAD grâce à une collaboration avec les développeurs qui incorporent ces règles dans les prochains clients.

Pour finir, extrêmement peu d'études ont été réalisées sur KAD concernant la pollution du réseau. Ainsi, [LMSW10] fut présenté en section 2.1.2.2. Il y est décrit deux méthodes pour polluer le réseau, l'une basée sur l'insertion de nœuds à proximité des ressources, l'autre par l'émission de nombreuses requêtes visant à remplir l'index des pairs. La pollution est ensuite appliquée pour réaliser un DDoS que les auteurs expérimentent avec succès sur le réseau. L'article [SRRS09], présenté en section 2.1.4, propose de limiter la pollution par un ensemble de vérifications réalisées par les nœuds responsables d'une ressource, ainsi que par un filtrage collaboratif des fichiers pollués par vote des pairs. La solution nécessite l'échange de nombreux messages tout en restant vulnérable à l'attaque Sybil ce qui la rend peu efficace.

## Conclusion

Nous avons montré dans ce chapitre comment KAD a mis en œuvre avec succès et à grande échelle une table de hachage distribuée pour réaliser une application de partage de fichiers. Depuis quelques années, le réseau a suscité l'intérêt de la communauté scientifique qui en a étudié les performances, la sécurité ou encore le comportement des pairs. Cependant, certains aspects de KAD demeurent méconnus. En particulier, aucune observation ni quantification d'attaques réelles n'a été réalisée alors que KAD a été prouvé vulnérable à de nombreuses attaques (pollution, attaque Sybil, ...). De même, la diffusion des contenus malveillants au sein de ce réseau n'a pas été étudiée.

Dans la suite de ce manuscrit, KAD se présente comme le principal support de nos contributions et bénéficie à ce titre directement des nouvelles connaissances acquises sur ce réseau dans le cadre de cette thèse et des solutions proposées pour en améliorer la sécurité des contenus. Dans la seconde partie de cette thèse, après avoir mis en évidence les limites des protections insérées dans KAD contre les attaques Sybil, nous proposons une architecture permettant de superviser précisément les accès aux contenus en capturant l'ensemble des requêtes émises par les pairs à partir d'une recherche de mots-clés. Nous appliquons notre solution à l'étude d'un type de contenu illégal diffusé au sein de KAD, à savoir des contenus à caractère pédophile. Dans la dernière partie de ce manuscrit, nous constatons l'ampleur des attaques affectant KAD. Nous montrons ainsi que le réseau est largement affecté par une nouvelle forme de pollution particulièrement néfaste, mais également que de nombreux pairs suspects sont insérés dans la DHT à proximité des contenus indexés. Pour remédier à ce problème, nous proposons une solution permettant de détecter et de lutter efficacement contre les insertions ciblées nœuds qui est parfaitement compatible avec le réseau actuel.

Deuxième partie

# Supervision des contenus et contrôle de leur accès

# Chapitre 4

# Évaluation des mécanismes de défense contre l'attaque Sybil

**Sommaire**



## Introduction

L'état de l'art a montré que la supervision de grands réseaux pair-à-pair entièrement distribués est une tâche très complexe quand on s'intéresse aux contenus, notamment à cause des problèmes de pollution. Nous proposons, dans cette partie, une nouvelle architecture de supervision alliant des sondes distribuées et des pots de miel nous permettant d'étudier très précisément l'activité des contenus partagés dans un réseau P2P structuré. Avant de présenter notre architecture de supervision puis son application, respectivement dans les chapitre 5 et 6, nous étudions dans ce chapitre les mécanismes de protection contre l'attaque Sybil insérés récemment dans KAD.

En effet, même si l'élaboration de défenses contre l'attaque Sybil a donné lieu à de nombreuses études, la majorité des solutions proposées sont difficiles à mettre en œuvre ou ne sont pas adaptées au paradigme pair à pair. Leur application est encore plus délicate lorsqu'on considère les contraintes d'un grand réseau P2P déjà existant tel que KAD, qui ne dispose pas d'infrastructure de gestion des pairs et qui requiert une rétro-compatibilité entre les différents clients. En l'absence de protection, il a été prouvé que KAD est très vulnérable à diverses attaques que nous rappelons brièvement dans le paragraphe suivant. Dans ce contexte, il est important d'améliorer la sécurité du réseau P2P KAD et les contraintes du réseau rendent cette tâche difficile.

| Clients / Protection | Suivi de paquets | Limitation IP | Vérification IP |
|---|---|---|---|
| eMule 0.48a / aMule 2.1.3 | Non | Non | Non |
| eMule 0.49a / aMule 2.2.1 | Oui | Oui | Non |
| eMule 0.49b / aMule 2.2.2 | Oui | Oui | Oui |

TABLE 4.1 – Protections actives en fonction de la version du client

### Historique des attaques

Avant la mise en œuvre des protections, Steiner et al [SENB07a] avaient montré la vulnérabilité de KAD à l'attaque Sybil en réussissant à contrôler une partie de la DHT grâce à l'injection de nombreux faux pairs. Pour cela, un explorateur se chargeait dans un premier temps de collecter les informations sur les pairs du réseau (ou d'une zone plus restreinte) en interrogeant chacun d'entre eux avec de nombreuses requêtes de localisation (`Kademlia_REQ`) permettant de découvrir la table de routage de chaque pair. Dans un second temps, le coeur de l'attaque Sybil consistait à contacter chaque pair découvert afin de polluer sa table de routage avec des Sybils. Précédemment triviale, cette étape est désormais protégée par les mécanismes étudiés dans ce chapitre.

La seconde attaque majeure dont les protections cherchent à prémunir le réseau est décrite dans l'article [WTCT+08]. Cette attaque ne se limite pas à la simple injection de Sybils mais corrompt les références de pairs normaux dans les tables de routage en écrasant leurs informations. Les attaquants sont ainsi capables d'écraser une entrée légitime (KADID/adresse IP) par leur propre référence avec un seul message `Hello_REQ` ré-annonçant le KADID ciblé avec de nouveaux paramètres (adresse IP, ports), pointant ainsi la faiblesse de la gestion des identités dans KAD. Cette vulnérabilité peut être exploitée de façon encore plus pernicieuse. Afin d'économiser des ressources, l'attaquant peut corrompre les références de contacts avec les informations d'autres pairs du réseau. Ceci rend l'entrée écrasée inutilisable car le pair annoncé ne possède pas réellement l'identifiant écrasé, et ne peut donc pas bien gérer les requêtes qui lui sont transmises. Cette attaque, couplée à un explorateur, permet à un attaquant de partitionner le réseau en corrompant méthodiquement les tables de routage.

Ces études ont montré que KAD pouvait être abusé de différentes manières, entraînant d'importants dysfonctionnements dans le système tout en nécessitant peu de ressources.

### Historique des protections

Ces problèmes de sécurité n'ont été pris en considération que récemment dans le réseau P2P KAD. L'importance des faiblesses de KAD et des problèmes de sécurité engendrés ont en effet forcé les développeurs à réagir en implantant de nouvelles protections dans les dernières versions des clients. Ceux-ci intègrent désormais des mécanismes de protection basés sur une détection locale des comportements suspects afin de limiter les effets d'une attaque Sybil. Même si les anciens clients sont toujours vulnérables, la mise à jour progressive des clients composant le réseau devrait rendre les attaques moins efficaces et constituer une base saine pour son évolution future. Les clients KAD implantent différents niveaux de protection en fonction de leur version comme indiqué dans le tableau 4.1. Pour des versions similaires, eMule et aMule sont basés sur le même code source et se comportent de la même manière.

Les dernières versions des clients de KAD ont ainsi introduit de nouveaux mécanismes de protection visant à limiter ces attaques. Ces mécanismes n'ont cependant jamais été décrits ni évalués. Ils répondent directement aux attaques définies dans l'état de l'art et pour lesquelles il

est nécessaire de réévaluer la faisabilité et l'efficacité sur les clients actuels. A travers la compréhension et l'évaluation des mécanismes destinés à limiter les attaques Sybil, notre objectif est ici de valider ces protections et, le cas échéant, d'identifier les faiblesses potentielles permettant malgré tout d'agir sur le réseau afin de mettre à jour les connaissances des vulnérabilités de KAD.

Nous décrivons tout d'abord chacune des trois protections identifiées, à savoir : la protection contre l'inondation de requêtes, la limitation des adresses IP et la vérification des identités. Nous évaluons ensuite l'efficacité de chaque mécanisme au travers d'attaques spécifiques permettant d'évaluer la robustesse de différentes versions de clients KAD aux vulnérabilités précédemment identifiées. Enfin, nous présentons une nouvelle attaque distribuée permettant de dépasser les protections existantes et discutons de leurs limites.

## 4.1 Description des protections

Les versions des clients KAD antérieures à eMule 0.48a et aMule 2.1.3 incluses, ne présentent aucune protection contre l'attaque Sybil. Les clients acceptent les messages entrant avec très peu de vérification, notamment, les messages peuvent être reçus aussi rapidement que le client peut les traiter. Concernant la table de routage, il n'y a aucune restriction au sujet du KADID ou de l'adresse IP d'un pair. Un contact peut être ajouté dès lors qu'il correspond à une place libre dans un k-bucket du client courant. Il est ainsi possible de remplir rapidement l'intégralité de la table de routage d'un pair en ayant connaissance du KADID du pair ciblé et de la table structure de la table de routage de KAD.

Nous avons identifié et évalué trois nouveaux mécanismes de protection présents dans les versions suivantes des clients KAD.

### 4.1.1 Suivi de paquets et protection contre l'inondation

La première protection concerne l'ajout d'une mémoire à court terme (12 minutes) dans le client, gardant une trace des derniers messages échangés afin d'en écarter ceux suspects. Plus précisément, cet historique sert deux objectifs. D'une part, il permet de détecter et supprimer les réponses non-sollicitées envoyées par certains pairs alors qu'aucune requête ne leur a été adressée par le client courant. D'autre part, sa principale fonction est de permettre une protection contre l'inondation de messages. Grâce à l'historique et à l'horodatage des messages précédemment reçus, le client peut détecter l'envoi rapide de nombreux messages depuis une même adresse IP source, limitant ainsi l'efficacité d'un explorateur tel que présenté dans [SENB07a].

Pour chaque type de message KAD, un seuil définit la vitesse maximale de réception de ces messages depuis une même source. En dessous de ce seuil, le pair émetteur est considéré comme légitime. Si la vitesse maximale d'émission est dépassée par un pair, les messages en provenance de ce dernier sont alors ignorés par le client. Si le taux des messages dépasse largement (5 fois) la limite, l'émetteur est considéré comme un attaquant et son adresse IP est bannie du client.

### 4.1.2 Limitation des adresses IP

La limitation des adresses IP est le cœur du mécanisme de protection contre l'attaque Sybil. Elle est basée sur l'idée que le nombre de pairs légitimes partageant une même adresse IP publique est relativement limité, contrairement aux Sybils. Ainsi, avant d'ajouter un contact à la table de routage, son adresse IP est désormais comparée aux contacts déjà présents dans celle-ci et le

| Clients | Séquence | Nb de messages | Chiffrement |
|---|---|---|---|
| eMule 0.48a | `Hello_REQ` - `Hello_RES` - `Hello_REQ`(challenge) - `Hello_RES`(challenge) | 4 | Non |
| eMule 0.49a | `Hello_REQ` - `Hello_RES` - `PING` - `PONG` | 4 | Oui |
| eMule 0.49b | `Hello_REQ` - `Hello_RES` - `Hello_RES_ACK` | 3 | Oui |

TABLE 4.2 – Différentes vérifications de l'adresse IP en fonction des clients

contact est ignoré le cas échéant. Cette contrainte n'autorise qu'une instance de la même adresse IP dans une table de routage, autrement dit, un seul KADID par adresse IP.

La limitation tient également compte de la proximité des adresses IP présentes dans la table de routage et interdit plus de 10 adresses venant d'un même sous réseau de taille /24 ou plus petit. Au plus 10 pairs d'un même sous réseau peuvent donc être simultanément dans une même table de routage. Cette contrainte vise à empêcher une unique entité disposant d'une plage d'adresses IP de lancer une attaque. Enfin, une dernière contrainte empêche deux pairs d'un même sous-réseau d'être dans un même k-bucket dans la table de routage d'un client. Cela oblige donc les pairs dont l'adresse IP est proche d'avoir des identifiants (KADID) éloignés les uns des autres sur la DHT. Cette contrainte vise à empêcher une attaque ciblée sur un identifiant dans le cas où une entité disposant d'un sous-réseau positionnerait les Sybils proches de l'information à contrôler. Ces limitations rendent l'attaque Sybil décrite précédemment extrêmement coûteuse, puisque chaque Sybil doit dorénavant afficher une adresse IP publique différente.

Il convient de remarquer que les utilisateurs utilisant la translation d'adresse sont peu affectés par ces contraintes et peuvent toujours participer au réseau. En effet, même si deux pairs partageant une adresse IP ne peuvent pas être référencés par un même contact, une table de routage ne contient que quelques centaines de contacts alors que le réseau en compte des millions, ce qui rend la probabilité d'une telle collision négligeable.

### 4.1.3 Vérification de l'identité des pairs

Les derniers clients KAD ont introduit des éléments permettant une gestion plus rigoureuse de l'identité des pairs afin de limiter l'usurpation d'identité (écrasement de l'adresse IP ou du KADID) entre pairs. Ainsi, avant qu'un contact ne puisse être ajouté ou modifié dans la table de routage, KAD vérifie son adresse IP par un échange de messages (three-way handshake) assurant que le pair dispose bien de l'adresse IP annoncée. Les versions plus anciennes de KAD sont vérifiées par un processus similaire grâce à d'autres primitives du protocole comme indiqué dans le tableau 4.2. A la suite d'une annonce, les clients les plus anciens sont vérifiés par un second échange `Hello_REQ` - `Hello_RES` incluant, en tant que défi, un KADID aléatoire devant être retourné. Les clients plus récents disposent des primitives PING et PONG permettant de réaliser ce test en y ajoutant une identification chiffrée. Enfin, la dernière version réalise la vérification directement à la suite de l'annonce grâce à la nouvelle primitive `Hello_RES_ACK` ce qui la rend plus efficace.

De plus, un pair ne peut plus désormais mettre à jour ses paramètres (adresse IP, port) dans la table de routage d'un autre sans détenir la clé privée correspondant à la clé publique présentée initialement. Ceci rend impossible la modification des paramètres d'un pair par une tierce partie malveillante. Enfin, les pairs qui échouent lors de la vérification sont marqués comme « non-vérifiés » dans la table de routage et ne sont plus utilisés pour localiser les ressources. Ces protections empêchent les attaques basées sur l'usurpation d'adresse IP et l'écrasement des

contacts dans la table de routage. C'est une réponse aux vulnérabilités soulignées dans l'article [WTCT+08] et une des solutions envisagées par les auteurs eux-mêmes.

## 4.2 Évaluation des protections

### 4.2.1 Méthode

L'insertion de Sybils dans les tables de routage est la base pour réaliser les différentes attaques sur le réseau. C'est pourquoi nous mesurons, pour un client donné, la permissivité de sa table de routage en fonction des diverses protections activées et discutons de l'utilité des Sybils quand ceux-ci sont capables de passer les protections. Nous avons pour cela développé un logiciel d'attaque exploitant les faiblesses précédemment identifiées afin de tester les différents mécanismes. Nous commençons par l'infection d'un client non protégé pour ensuite ajouter une à une les nouvelles protections, tout en modifiant notre attaque en conséquence.

Notre évaluation des protections se limite à un unique client et ne considère pas l'ensemble du réseau. Ceci est motivé par le fait que les mécanismes de protection appliquent des contraintes locales et ne requièrent aucune interaction entre les pairs pour fonctionner. Par conséquent, les résultats obtenus en évaluant un seul client sont suffisants et la sécurité globale du réseau peut en être déduite.

Notre outil annonce les Sybils au client-test instrumenté en envoyant de nombreux messages `Hello_REQ` avec des KADIDs créés spécifiquement. La difficulté majeure de cette attaque est d'annoncer le bon identifiant pour chaque Sybil en fonction du pair ciblé de manière à correspondre parfaitement aux besoins de sa table de routage et ainsi maximiser la pollution. La génération des identifiants des Sybils a donc requis la compréhension précise de la structure et des algorithmes utilisés pour maintenir la table de routage de KAD car celle-ci, bien qu'héritée de Kademlia, en diffère sensiblement comme nous l'avons vu en section 3.1.3.1.

Grâce à la connaissance de la table de routage et connaissant l'identifiant du pair ciblé, nous pouvons calculer l'identifiant des Sybils de telle sorte qu'il corresponde à un k-bucket libre dans la table de routage de la victime. Les identifiants sont ainsi générés par des masques : KADID(cible) XOR masque[$i$] = KADID(Sybil[$i$]).

Comme nous injectons des contacts dont l'identifiant est forgé, ils peuvent atteindre des k-buckets très profonds dans la table de routage (avec des identifiants proches du pair ciblé) ce qui explique que la taille de la table de routage attaquée soit supérieure à celle constatée durant une exécution normale. Du point de vue de la cible, l'annonce de Sybils très proches est considérée comme si la DHT était très peuplée.

### 4.2.2 Résultats

#### 4.2.2.1 Sans protection

La première attaque réalisée considère le client aMule 2.1.3 qui est démuni de protection et nous sert de référence pour les évaluations suivantes des nouveaux mécanismes. Cette version non protégée permet de réaliser une attaque Sybil simple et très efficace. L'identifiant des Sybils est construit comme décrit précédemment, les Sybils affichent tous la même adresse IP et sont annoncés rapidement (4 Sybils par seconde). De plus, cette version du client réalise une mise à jour des contacts approximative qui s'avère être profitable à l'attaque.

En effet, afin de maintenir la table de routage à jour, chaque contact de KAD est testé périodiquement en émettant une requête `Hello_REQ` puis son statut est validé à la réception de la

réponse `Hello_RES`. Cette version implante une optimisation qui devient une faille de conception lorsque l'on considère l'attaque Sybil. Ainsi, chaque `Hello_RES` reçue depuis une adresse IP (quelque soit son KADID) acquitte tous les contacts de la table de routage ayant cette adresse IP. De plus, un pair inconnu peut s'annoncer directement en émettant la réponse `Hello_RES` (au lieu de la requête), et être malgré tout ajouté à la table de routage. Par conséquent, notre attaque envoie des `Hello_RES` qui permettent à chaque Sybil de s'annoncer tout en acquittant tous les précédents déjà placés. Lorsque l'attaque est terminée, un seul et unique message `Hello_RES` est ensuite suffisant pour maintenir les centaines de Sybils en vie.

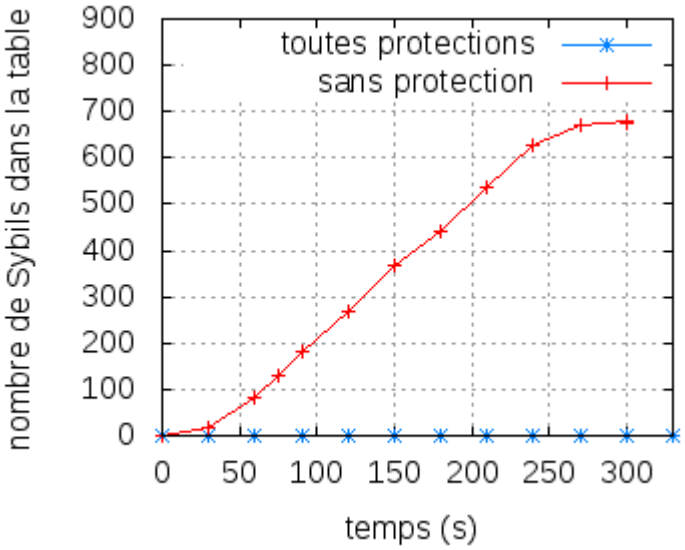


FIGURE 4.1 – Propagation des Sybils dans la table de routage

La courbe 4.1 montre l'évolution du nombre de Sybils dans la table de routage de deux clients subissant cette attaque : une ancienne version vulnérable et la dernière version incluant tous les mécanismes de protection. La table de routage de la version vulnérable est largement corrompue après l'attaque alors que les Sybils sont absents de la table de routage du client protégé, ce qui est bien le résultat attendu. Après l'émission des Sybils pendant 300s, la table de routage infectée compte ainsi 70% de Sybils (689 Sybils sur 953 contacts). La partie de la table qui n'est pas polluée est constituée des 200 contacts légitimes sauvegardés lors de la précédente exécution et re-chargés pour rejoindre le réseau. La courbe 4.2 montre la vitesse de remplissage de la table de routage selon que le client soit ou non attaqué. Ici le nombre global de contacts est considéré indépendemment de leur qualité de Sybil ou de pair légitime. Malgré une vitesse modérée d'émission des Sybils, la table de routage croît anormalement lors de l'attaque. Nous évaluons ensuite plus en détail ces mécanismes en les confrontant à des attaques plus élaborées.

#### 4.2.2.2 Suivi des messages et protection contre l'inondation

La seconde expérience a pour objectif d'évaluer la protection contre l'inondation de messages. Nous avons ainsi lancé l'attaque sur le client eMule 0.49a en désactivant la limitation des adresses IP. Pour être fonctionnelle, l'attaque a dû être modifiée de deux manières. D'une part, le suivi des messages ne traite aucune réponse si le client courant n'a pas envoyé la requête correspondante. La faiblesse autorisant l'annonce et le maintien des Sybils par un unique message `Hello_RES`

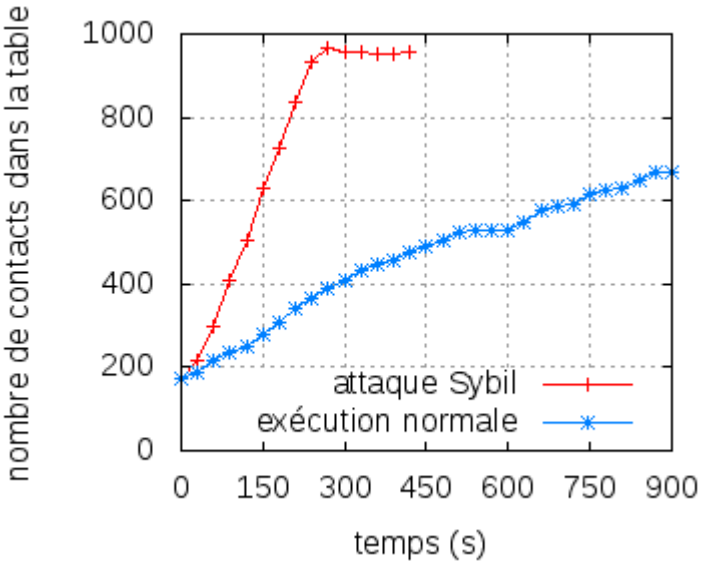


FIGURE 4.2 – Nombre de contacts dans la table de routage en fonction du temps

n'est donc plus possible. Nous avons modifié l'attaque en conséquence en répondant aux requêtes `Hello_REQ` émises par le client victime pour tester les Sybils. Cette tâche n'est cependant pas triviale. En effet, un message `Hello_REQ` ne spécifie pas le KADID du pair destinataire mais celui-ci doit être retourné dans la réponse `Hello_RES`. Un client normal ne gère que son propre KADID, il n'y a donc jamais d'ambiguïté pour répondre. Cependant, notre outil d'attaque gère de multiples KADIDs pour les Sybils et ne sait pas a priori quel KADID est concerné par la requête `Hello_REQ` reçue. Pour résoudre ce problème, chaque Sybil doit communiquer grâce à un port UDP différent qui permet de retrouver quel KADID est concerné par la requête, selon le port sur lequel elle arrive. Cette modification permet d'assurer la pérennité de l'attaque, sans quoi les Sybils insérés sont supprimés après deux heures ou dès qu'un contact plus récent entre dans le k-bucket.

D'autre part, pour ne pas déclencher le mécanisme protégeant un client contre une émission anormale de messages, la fréquence d'émission des messages `Hello_REQ` doit être baissée à deux par minute pour rester sous le seuil de détection. Nos expériences attestent du bon fonctionnement de ce mécanisme. Le taux d'émission avant le rejet des paquets `Hello_REQ` est défini à trois par minute, et au delà de 15 paquets par minute, l'adresse IP émettrice est bannie du client, ce que nous avons pu constater. En se conformant à ces contraintes, l'attaque est fortement ralentie comme le montre la courbe 4.3. En effet, l'injection du même nombre de Sybils que précédemment (courbe 4.1) prend plus de huit heures alors qu'elle prenait quelques minutes sans cette protection. L'attaque demeure pour autant possible, la table de routage étant polluée au final avec 60% de Sybils (540/873). La protection contre l'inondation protège efficacement la majorité des connexions au réseau KAD qui sont de courte durée. Les connexions longues restent cependant vulnérables.

#### 4.2.2.3 Limitation des adresses IP

Notre précédente attaque a été réalisée en utilisant qu'une seule adresse IP pour l'ensemble des Sybils. En activant la limitation des adresses IP, nous avons en effet constaté que seul le premier

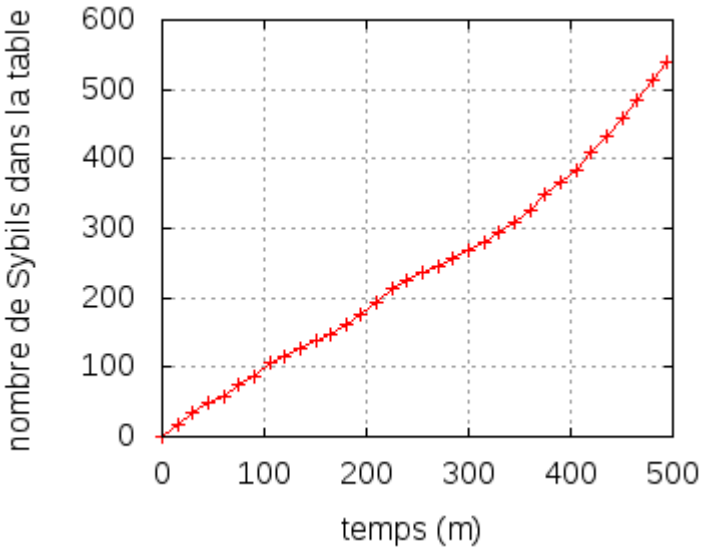


FIGURE 4.3 – Propagation des Sybils dans la table de routage protégée contre l'inondation de messages

Sybil est inséré dans la table de routage, tous les autres KADID présentant la même adresse sont ignorés. Pour contourner ce problème, nous avons modifié l'outil d'attaque de manière à ce que chaque Sybil soit annoncé avec une adresse IP usurpée aléatoirement choisie. La courbe 4.4 montre que cette modification permet la pollution de la table de routage. De plus, nous avons également constaté que la protection contre l'inondation est complètement inefficace avec cette modification. En effet, le suivi des paquets utilisé pour détecter l'envoi abusif de messages mesure le nombre de messages venant d'une même source par rapport à l'adresse IP émettrice. En usurpant leur adresse IP, les messages semblent venir de sources différentes et l'attaque n'est pas détectée. La vitesse de propagation des Sybils usurpant leur adresse IP est donc similaire à celle observée pour un client sans protection (courbe 4.1), le suivi de paquets protégeant contre l'inondation des messages ne pouvant appréhender une attaque distribuée.

L'usurpation d'adresse IP rend cependant les Sybils beaucoup moins utiles pour l'attaquant. Celui-ci ne peut plus espionner, éclipser ou polluer les entrées de la DHT avec de tels Sybils, étant incapable de recevoir les messages. Il est également beaucoup plus délicat de maintenir en vie les Sybils car leur adresse IP étant usurpée, ils ne peuvent recevoir les `Hello_REQ` de vérification. Pour les maintenir, l'attaquant doit anticiper le test en émettant pour chaque Sybil des messages *Hello_REQ* de manière pro-active vers le pair ciblé avant que celui-ci ne sollicite les Sybils par une preuve d'activité.

Néanmoins, l'usurpation d'adresse IP était utilisée dans l'attaque [WTCT+08] pour écraser les contacts dans les tables de routage et réaliser le déni de service final. A ce niveau, cette attaque demeure donc possible ce qui nous amène à étudier la dernière protection visant à empêcher l'usurpation de KADID et d'adresse IP.

#### 4.2.2.4 Vérification des identités

Nous avons finalement rejoué cette dernière attaque avec l'ensemble des protections activées, incluant la vérification des identités. Les Sybils injectés ne peuvent plus se maintenir dans la

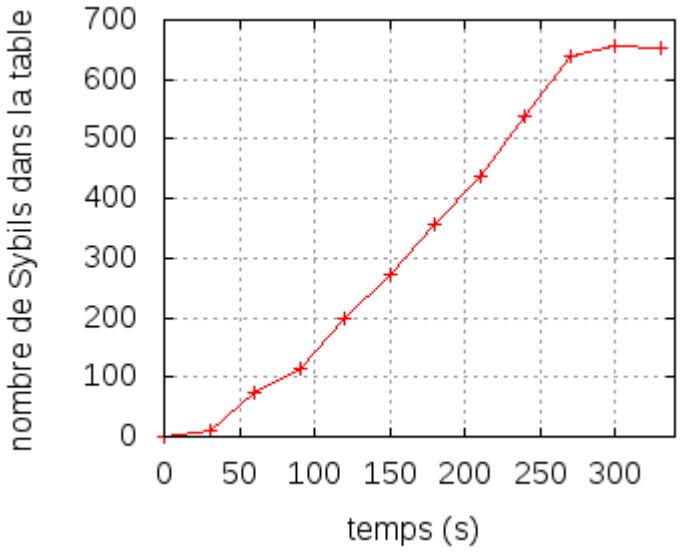


FIGURE 4.4 – Propagation des Sybils usurpant leur adresse IP dans la table de routage protégée contre l'inondation de messages et par la limitation d'adresse IP

table de routage avec une adresse IP usurpée car ils ne peuvent accomplir le test nécessitant une interaction (« three-way handshake »). La courbe 4.6 montre que les Sybils sont dans un premier temps insérés avec un statut « non vérifié », tous comme les autres pairs lors d'une exécution normale 4.5. Ce statut empêche les contacts d'être utilisés pour router des messages ce qui les rend inoffensifs. En revanche, comme le test vérifiant l'identité échoue, les Sybils sont finalement supprimés de la table de routage (courbe 4.6) alors qu'en condition normale les contacts sont progressivement vérifiés jusqu'à l'obtention d'une table de routage entièrement valide après 1500s (courbe 4.5).

De plus, étant donné le statut « non vérifié », les Sybils sont supprimés beaucoup plus rapidement de la table de routage que le délai limite usuel d'une heure. Ne pouvant être maintenus en vie artificiellement, les Sybils usurpant des adresses IP polluent la table pour un temps limité. Par ailleurs, les contacts ne peuvent désormais plus mettre à jour l'information d'un pair sans posséder la clé privée initialement utilisée lors de l'enregistrement dans la table de routage. Ceci empêche l'attaque décrite dans [WTCT+08] qui pouvait alors partitionner le réseau.

Les résultats obtenus sur cette dernière version montrent l'avancée significative que constituent les nouveaux mécanismes contre l'attaque Sybil. Ils empêchent aujourd'hui l'injection massive de Sybils provenant d'une même adresse IP, et de ce fait, augmentent considérablement le coût de l'attaque Sybil telle que pratiquée par [SENB07a]. S'il est devenu très coûteux en adresses IP de réaliser une attaque à large échelle sur le réseau, nous allons montrer que des attaques très ciblées restent en revanche efficaces avec peu de ressources.

### 4.2.3 Limites

Les nouveaux mécanismes de protection permettent de limiter la corruption massive des tables de routage au regard des ressources désormais nécessaires. Cependant, des attaques ciblant précisément une entrée sont possibles et nécessitent beaucoup moins de ressources. Dans [SENB07a], Steiner et al. ont montré que 32 Sybils permettaient de prendre le contrôle de l'indexation du

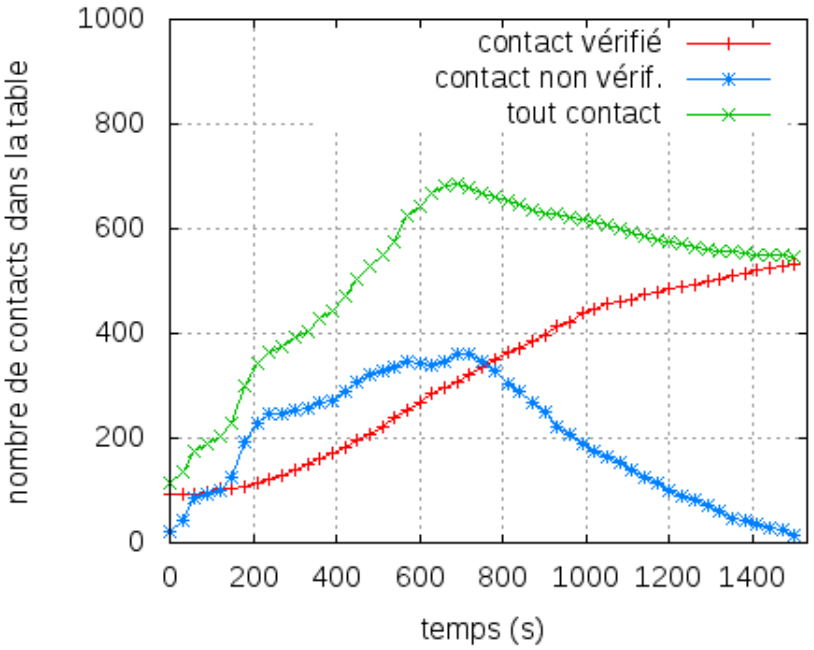


FIGURE 4.5 – Évolution du statut des contacts lors d'une exécution normale

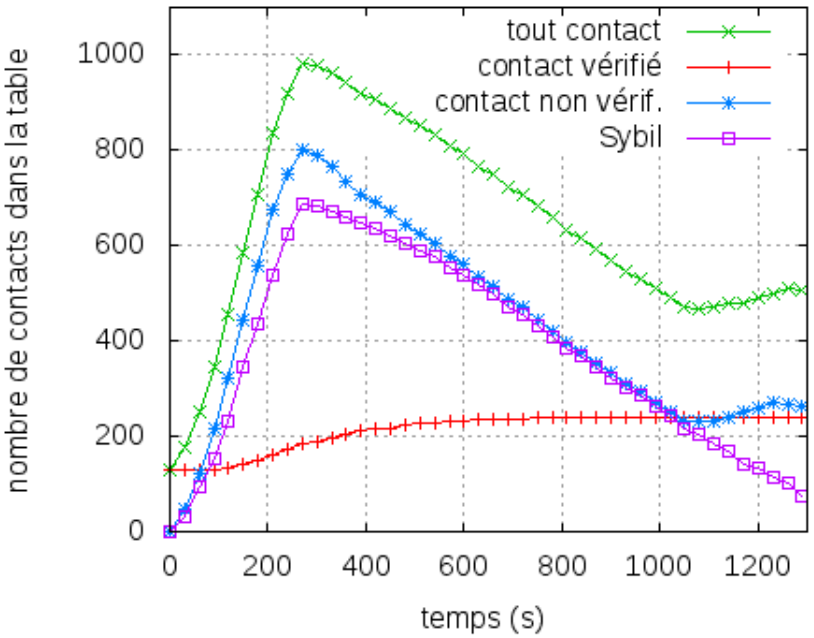


FIGURE 4.6 – Évolution du statut des contacts lors d'une attaque Sybil

mot clé « the ». En revanche, la méthode de propagation alors utilisée (exploration du réseau puis annonce des Sybils à chaque pair) est désormais limitée.

Nous pensons que la principale faiblesse autorisant des attaques ciblées (telle que l'éclipse) sur la DHT est moins l'injection de Sybils dans les tables de routage que la possibilité laissée aux pairs de choisir librement leur idenfiant proche d'une cible, ce qui reste possible malgré les protections actuelles. En choisissant quelques KADIDs extrêmement proches d'un identifiant particulier, c'est à dire beaucoup plus proches que n'importe quel pair choisissant son KADID aléatoirement sur 128bits, et étant donné l'algorithme de routage dans KAD qui consiste à utiliser les 10 nœuds les plus proches de la cible pour stocker une information, il est alors possible d'attirer avec une très forte probabilité les requêtes sur des nœuds que l'on contrôle. Ainsi, sans même nécessiter l'insertion directe des Sybils dans les tables de routage des pairs, leur simple proximité aux resources ciblées associée à l'efficacité de la procédure de localisation de KAD doit permettre leur découverte et l'interception des messages, ce que nous montrons dans le chapitre suivant.

## Conclusion

Afin d'évaluer l'efficacité des nouvelles protections contre l'attaque Sybil implantées dans les versions récentes de clients KAD, nous avons reproduit des vecteurs d'attaque en injectant des Sybils dans la table de routage de clients instrumentés, permettant ainsi d'évaluer l'efficacité de l'attaque selon les mécanismes de protection activés.

Alors que les anciens clients KAD sans protection étaient facilement et largement affectés par une attaque Sybil massive ne nécessitant qu'un seul ordinateur, notre évaluation a montré qu'une détection des comportements suspects, simplement locale au client, permettait de limiter efficacement cette menace. Les solutions implantées par les développeurs de KAD respectent les deux principales contraintes du réseau à savoir l'absence de coût d'infrastructure et la rétro-compatibilité entre clients. Le réseau progressivement composé de clients implantant ces protections devient plus robuste. Ces protections permettent donc, avec un coût négligeable, de combler d'anciennes vulnérabilités et de rendre l'attaque Sybil beaucoup plus difficile à réaliser en augmentant considérablement son coût en ressources. Une attaque ne nécessitant qu'un seul ordinateur il y a quelques mois nécessite aujourd'hui un botnet ou une architecture distribuée beaucoup plus complexe à mettre en œuvre.

Néanmoins, même si ces protections constituent une avancée, nous pensons que des faiblesses de KAD peuvent toujours être exploitées, en particulier, le fait de choisir librement son KADID associé à un algorithme de localisation performant, et laissent KAD vulnérable à des attaques ciblées et distribuées. Bien qu'imparfaites, ces défenses sont absolument génériques et peuvent facilement être appliquées à d'autres réseaux P2P structurés. Ces protections de bon sens devraient constituer la sécurité minimale implantée par toute DHT, à moins d'être immunisée contre l'attaque Sybil dès la conception.

Le chapitre suivant décrit plus précisément les vulnérabilités permettant l'interception des messages sur la DHT et comment notre architecture de supervision les exploite pour superviser précisément la DHT de KAD à un coût raisonnable.

# Chapitre 5

# HAMACK, une architecture de honeypots distribués

**Sommaire**



## Introduction

Nous avons vu que les réseaux P2P sont utilisés pour partager des contenus malveillants. Bien qu'ils offrent par ailleurs de nombreuses possibilités, ils fournissent également un support à ces activités dangereuses diffusant volontairement des contenus hautement indésirables et illégaux. Nous avons également mentionné les difficultés inhérentes à la supervision de grands réseaux dynamiques complètement distribués et notamment le problème crucial de la pollution dont certaines formes amènent des utilisateurs à télécharger un contenu à leur insu. Notre objetif est ici de pouvoir superviser précisément l'activité de tels contenus sur le réseau KAD, en minimisant les faux positifs. Ceci permet d'une part d'en comprendre les principes de diffusion et d'autre part de créer un moyen de supervision fiable dans le cadre de la lutte contre les contenus illégaux, notamment ceux à caractère pédophile.

| Nom de fichier | Nb de Sources | Taille |
|---|---|---|
| "Matrix 2 Reloaded dvdrip.avi" | 362 | 695.21MB |
| "Matrix Revolutions spanish.avi" | 328 | 625.23MB |
| "Game Enter the Matrix DVD.iso" | 298 | 681.19MB |
| "Matrix Reloaded cd1 divx.avi" | 8 | 679.74MB |

TABLE 5.1 – Résultats d'une recherche sur le mot-clé "matrix"

**Stratégie de supervision**

Le fondement de notre approche est de considérer que la supervision des fichiers existants n'est ni fiable du fait de la pollution, ni efficace du fait de la concurrence avec les autres pairs. La pollution ne permet pas de savoir au travers de quels mots-clés un utilisateur accède à un contenu ce qui engendre des faux positifs. Par conséquent, seule la supervision directe des mots-clés entrés par l'utilisateur, ou de fichiers créés spécifiquement par un pot de miel et dont l'indexation n'est pas corrompue peuvent fournir des observations dignes de confiance. L'approche que nous proposons, HAMACK (Honeynet Architecture for Monitoring content ACcess in KAD), part de ce constat et vise à superviser l'accès aux contenus de KAD en attirant dans un premier temps l'ensemble des requêtes émises par les pairs pour un contenu spécifique puis en constatant les accès à des pots de miel distribués dont l'efficacité est assurée par le contrôle du mécanisme d'indexation.

L'efficacité d'un pot de miel dépend de l'attractivité des appâts proposés, dans notre cas, de celle des différents faux fichiers utilisés. Les fichiers annoncés par le pot de miel doivent donc être attractifs auprès des utilisateurs et éviter la concurrence des fichiers réels similaires. Plusieurs informations sont présentées à un utilisateur à l'issue d'une recherche de fichier, afin qu'il puisse choisir celui désiré (voir tableau 5.1). Parmi celles-ci, le nom du fichier et sa taille sont des informations importantes et peuvent être facilement contrôlées lors de la création des appâts, par opposition au nombre de sources estimé dont il est difficile de maîtriser le résultat avec des pots de miel conventionnels. Par ailleurs, cette dernière information est capitale car elle constitue un critère privilégié pour trier les résultats d'une recherche, les fichiers affichant un nombre de sources élevé étant les plus attractifs. Ceux-ci sont en effet populaires ce qui indique, d'une part un temps de téléchargement réduit, et d'autre part un contenu plus fiable. Typiquement, le dernier résultat du tableau 5.1 devrait être ignoré par la grande majorité des utilisateurs. Nous montrerons par nos mesures que le nombre de sources estimé pour les appâts est une information capitale dans la mise en œuvre de pots de miel efficaces.

Une solution possible pour augmenter le nombre de sources estimé pour un fichier est de multiplier les machines participant aux ressources du pot de miel. Cependant, cette solution montre vite ses limites car des fichiers populaires peuvent facilement dépasser le millier de sources. Alors que dans eDonkey le nombre de sources estimé est retourné par le serveur, dans KAD celui-ci est calculé par les pairs en charge d'indexer un mot-clé. Prendre le contrôle du mécanisme d'indexation permettrait donc deux propriétés augmentant l'attractivité du pot de miel : d'une part en autorisant le contrôle du nombre de source estimé pour les fichiers annoncés, et d'autre part, en éclipsant les fichiers réguliers concurrents. Ainsi, superviser l'accès aux différents mots-clés et annoncer des fichiers attractifs dont l'indexation est parfaitement maîtrisée permet à HAMACK de vérifier avec précision l'intérêt d'un pair pour un contenu particulier, depuis la recherche de mot-clé jusqu'au téléchargement final de fichiers appâts. Ces fonctionnalités font l'originalité de notre approche par rapport aux pots de miel classiques et aux attaques éclipse.

Par ailleurs, les mécanismes de protection nous obligent à adopter une approche distribuée.

**Positionnement**

Plusieurs travaux parmi ceux présentés dans l'état de l'art sont proches des nôtres. En premier lieu, l'attaque Sybil ciblée réalisée par Steiner [SENB07a] permettant d'éclipser certaines entrées de la DHT. Si notre architecture utilise une idée similaire (insérer des nœuds proches d'une clé), la mise en œuvre de l'attaque est cependant différente du fait des mécanismes protégeant désormais la table de routage. En effet l'explorateur utilisé pour préparer l'attaque et l'injection massive de Sybils est inutilisable en raison des récentes protections. Nous avons conçu une stratégie originale pour propager les Sybils ne requérant ni explorateur, ni annonce directe ainsi qu'une architecture distribuée plus complexe. Enfin, l'application de l'attaque est différente et sert le cadre plus général d'une supervision avancée des accès aux contenus en manipulant l'indexation de ceux-ci.

Les travaux de Memon [MRGS09], concomitants aux nôtres, présentent une autre méthode de supervision pour KAD compatible avec les protections. Comme notre approche, ces travaux se basent sur la proximité entre la cible et les sondes pour attirer les requêtes. Ici, l'objet de la supervision est chaque pair du réseau se voyant ainsi associer un unique superviseur, dont la proximité le rend éligible aux mêmes requêtes que le pair supervisé. Cependant, cette approche nécessite un explorateur pour découvrir les pairs et ne peut observer une grande portion du réseau avec précision. Elle est donc adaptée à l'étude de trafic mais n'a pas prouvé sa capacité à superviser des contenus spécifiques disséminés sur la DHT, ni à pouvoir contrôler leur indexation pour investiguer plus précisément le comportement des utilisateurs.

Nous décrivons dans ce chapitre le fonctionnement de notre architecture, notamment comment celle-ci interagit avec les algorithmes de KAD pour en exploiter les faiblesses, et estimons de manière analytique son efficacité. Nous présentons ensuite ses fonctionnalités permettant : (1) de superviser de manière transparente l'ensemble des requêtes relatives à un contenu du réseau, (2) d'éclipser les entrées existantes pour celui-ci et (3) d'attirer les requêtes de téléchargement sur les pots de miel en insérant dans le mécanisme d'indexation de faux fichiers semblant très attractifs aux utilisateurs. Nous présentons finalement sa mise en œuvre sur le réseau pair à pair KAD permettant d'en évaluer expérimentalement les performances.

## 5.1 Contrôle du système d'indexation de KAD

### 5.1.1 Description du processus de réalisation de service de KAD

Cette section décrit la procédure de réalisation de service de KAD que nous souhaitons contrôler et montre en quoi la grande efficacité du processus de localisation des ressources constitue une menace pour les données stockées lorsque l'on considère des attaques ciblées sur la DHT. En effet, tous les services de KAD nécessitent en premier lieu la localisation des pairs responsables, puis leur interrogation. Ces deux étapes sont réalisées par l'objet « Search ». Ainsi, chaque service (publication ou recherche de mots-clés ou sources de fichier) génère un objet « Search » autonome chargé de réaliser le service demandé. Il y a autant d'objets gérés en parallèle par l'application qu'il y a de services concomitants, mais tous restent distincts et indépendants.

La procédure de recherche est donc constituée de deux phases distinctes. La première partie consiste à trouver les pairs actifs dans la zone de tolérance de l'information recherchée afin de remplir le tableau des contacts *possibles*. Ce tableau contient tous les contacts trouvés pendant

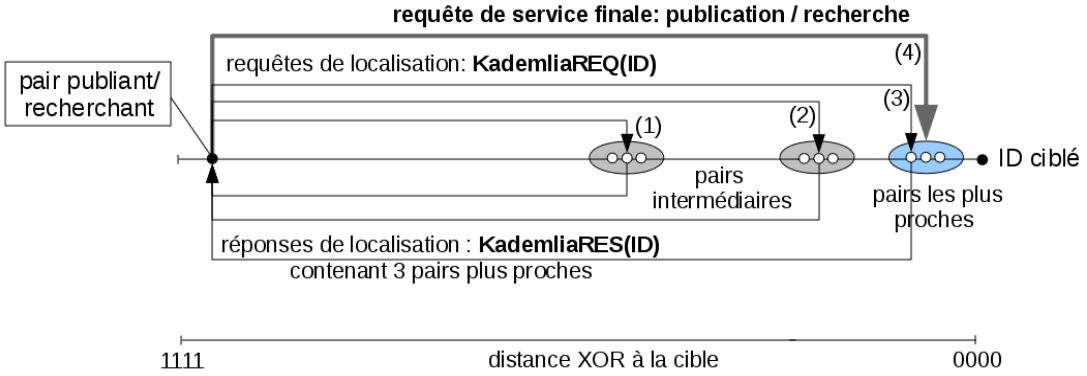


FIGURE 5.1 – Messages échangés durant la procédure de localisation

l'exécution du service et les ordonne par distance par rapport à la cible. Cette partie est réalisée en envoyant plusieurs `KADEMLIA_REQ` en parallèle avec le KADID ciblé pour paramètre. Le routage de KAD se fait de manière itérative avec plusieurs requêtes envoyées en parallèle comme présenté par la figure 5.1 : en premier, les 3 contacts les plus proches de la cible trouvés dans la table de routage sont interrogés pour fournir leurs connaissances encore plus proches, leurs réponses sont attendues et les 3 meilleurs contacts parmi les nouveaux reçus sont à nouveau interrogés. Seuls les contacts ayant répondu lors du processus de localisation, alors marqués comme *responded*, sont ensuite interrogés pour réaliser le service. Tant que des contacts plus proches sont retournés à chaque itération (1) (2), le processus de localisation continue de progresser, sans toutefois commencer la réalisation du service.

Les dernières versions des clients KAD (eMule 0.49c and aMule 2.2.4), vérifient les contacts découverts lors du processus de localisation avec les mêmes protections qu'appliquées à la table de routage et qui ont été présentées dans le chapitre précédent. En particulier, parmi les pairs trouvés, un seul KADID est autorisé par adresse IP et seules deux adresses IP du même sous réseau /24 sont considérées. Ces nouvelles contraintes imposent l'utilisation d'une architecture distribuée pour superviser les services de KAD, comme nous le proposons dans ce chapitre.

La seconde étape débute lorsque la recherche de contacts n'a pas localisé de contacts plus proches de la cible dans les 3 dernières secondes (3), indiquant que les derniers pairs interrogés sont les plus proches. Les meilleurs contacts alors trouvés et ayant répondu à une requête de localisation sont interrogés avec des requêtes spécifiques au service souhaité (4), par exemple `KADEMLIA2_PUBLISH_KEY_REQ` pour publier un mot clé. Les réponses retournées aux demandes de service conditionnent la poursuite ou la fin de celui-ci. L'objet « Search » se termine ainsi si le nombre de réponses positives nécessaires au service est atteint ou, en cas d'échec, par un temps limite. Par exemple, un service de publication est considéré comme terminé lorqu'au moins 10 pairs l'ont acquitté.

### 5.1.2 Stratégie de contrôle

Cette sous-section décrit comment le processus de réalisation des services dans KAD peut être contrôlé en dépit des protections présentées dans le chapitre précédent. La première étape consiste à positionner précisément un certain nombre de nœuds dont nous avons le contrôle par rapport au mot clé ou au fichier faisant l'objet de la supervision. Leur identifiant est ainsi choisi de telle sorte qu'ils soient plus proches de la cible (selon la distance XOR) que n'importe

quel autre pair normal ayant choisi aléatoirement son identifiant. Pour éviter les restrictions des mécanismes de protections, ces pairs sont distribués au niveau de l'Internet. Par ailleurs, ils sont autonomes car ils exécutent une version peu modifiée d'un client KAD régulier. Pour ces raisons, nous appellerons ces pairs contrôlés « sondes » ou « Honeypeers » plutôt que « Sybils ».

Le premier paramètre à considérer est l'identifiant (KADID) ciblé. Pour calculer les 128 bits qui le composent, nous appliquons la fonction de hachage MD4 utilisée par KAD au mot clé ou au fichier à superviser. Une fois le KADID obtenu, celui-ci est passé en paramètre à chaque Honeypeer qui peut alors en dériver son propre KADID en copiant les 96 premiers bits de la cible à l'identique et en choisissant les 32 autres bits aléatoirement. Les 32 bits aléatoires permettent de générer les KADIDs des Honeypeers sans concertation avec un risque de collision minime. D'autre part, 96 bits est une valeur suffisante pour s'assurer qu'aucun autre pair n'est plus proche de la cible que nos sondes. Soit $N$ le nombre moyen de pairs étant au moins aussi proches de la cible que l'une des sondes ; les pairs éligibles doivent choisir de manière aléatoire les premiers 96 bits composant son KADID similaires à la cible ce qui permet $2^{32}$ identifiants sur les $2^{128}$ possibles. En considérant un nombre de pairs majoré à 5 millions, le résultat (5.1) montre qu'avec un préfixe de 96 bits, le nombre moyen de pairs légitimes pouvant être plus proches que les Honeypeers est suffisamment faible pour être considérée comme nul.

$$N = \frac{5 \times 10^6}{2^{96}} = 3,79 \times 10^{-23} \tag{5.1}$$

Alors que les Honeypeers sont les nœuds les plus proches de la référence ciblée, le point critique de l'architecture est d'être capable d'attirer toutes les requêtes émises, étant donné la procédure de réalisation de service de KAD. Lorsqu'une requête de localisation `KADEMLIA_REQ` concernant la référence ciblée arrive sur un Honeypeer, celui-ci répond uniquement avec la liste des autres sondes déployées sur cette cible, assurant ainsi leur découverte. Comme la liste des contacts *possibles* gérée par l'objet « Search » est toujours ordonnée par distance, les Honeypeers sont placés dans les meilleurs positions pour recevoir les requêtes de service une fois la localisation terminée. Ainsi, dès qu'un pair a découvert l'un des Honeypeers pendant la phase de localisation, toutes les requêtes suivantes émises pour terminer la localisation et réaliser le service sont capturées par l'architecture.

Par conséquent, la probabilité d'attirer toutes les requêtes de service émises par un pair pour une référence ciblée peut être approximée par la probabilité de trouver au moins un des Honeypeers durant la procédure de localisation. Nous calculons cette probabilité dans le paragraphe suivant.

### 5.1.3 Modèle probabiliste

Depuis la première étude sur la DHT de KAD [SR06], il a été confirmé par [WTCT+08] et par nos travaux [CCF09a] que la structure de la table de routage de KAD est différente de l'arbre binaire décrit dans Kademlia [MM02]. Ces différences ont été décrites dans le chapitre 3 et illustrées par la figure 3.1. Pour rappel, la table de routage de KAD route plus rapidement car celle-ci contient plus de contacts à chaque niveau. Connaissant la table de routage et la procédure de localisation, nous sommes capables d'estimer la probabilité d'attirer les requêtes de service à destination d'une référence ciblée.

Nous considérons que le pair initiateur et chaque pair intermédiaire disposent d'une table de routage consistante et peuplée conformément à leur place respective dans la DHT. Nous faisons aussi l'hypothèse qu'il n'y a pas de partition dans le réseau P2P et que les Honeypeers sont référencés normalement par les autres. Étant donné l'espace d'adressage de la DHT, nous

considérons que le pair initiateur est à l'opposé de la référence ciblée afin de ne pas réduire artificiellement le nombre de sauts nécessaires pour la localiser.

Soit $H$ le nombre de Honeypeers trouvés à une étape donnée de la procédure de localisation et soit $(1-P_n(H=0))$ ou $P_n(H \geq 1)$ la probabilité de trouver au moins un des Honeypeers au $n^{eme}$ saut de la localisation. Comme les Honeypeers coopèrent lors de la localisation, nous considérons que la découverte d'un Honeypeer mène à la capture de l'ensemble des requêtes suivantes par HAMACK. Soit $NH$ le nombre de Honeypeers utilisés pour superviser une référence et $NP_n$ le nombre de pairs potentiels qui peuvent être interrogés à l'étape $n$ de la localisation. Au début de la procédure, la localisation doit être initiée en contactant les 3 contacts les plus proches de la cible depuis la table de routage, ce qui est modélisé par l'équation 5.2. Ensuite, à chaque étape, trois requêtes sont envoyées vers les contacts les plus proches trouvés à l'étape précédente et chaque réponse associée retourne 4 nouveaux contacts plus proches parmi $NP_n$ ce qui implique que les Honeypeers doivent être trouvés parmi les 12 contacts potentiels trouvés à une étape, ce qui est modélisé par l'équation 5.2. Pour estimer $P_n(H)$, nous pouvons calculer la probabilité de choisir $i$ Honeypeers parmi $NH$ à chaque étape, ce qui correspond à la loi hypergéométrique de paramètre (12, $NH$, $NP_n$+$NH$), puis sommer les différentes valeurs de $i$.

$$P_0(H \geq 1) = \sum_{i=1}^{i=3} \frac{C_{NH}^{i} \times C_{NP_0}^{3-i}}{C_{NP_0+NH}^{3}} \tag{5.2}$$

$$P_n(H \geq 1) = \sum_{i=1}^{i=12} \frac{C_{NH}^{i} \times C_{NP_n}^{12-i}}{C_{NP_n+NH}^{12}} \tag{5.3}$$

Soit $NP$ le nombre total de pairs dans le réseau. En considérant la table de routage de KAD, le pair initiateur dispose au moins d'un « 10-bucket » pour couvrir la portion de 1/16 de la DHT où se trouve la cible. Ainsi, les 3 premiers contacts utilisés pour initier la localisation sont choisis parmi $NP/2^4$ pairs.

$$NP_0 = \frac{NP}{16} = \frac{NP}{2^4} \tag{5.4}$$

A l'intérieur d'un même 10-bucket, les contacts ne sont pas ordonnés mais nous devons traduire le fait de prendre uniquement les 3 meilleurs parmi 10 en terme de probabilité. Nous supposons que les KADIDs à l'intérieur d'un 10-bucket sont uniformément distribués sur les bits non contraints. Ainsi, choisir les 3 meilleurs contacts parmi 10 améliore en moyenne 2 bits supplémentaires vers la cible. Une modélisation plus réaliste est donc :

$$NP_0 = \frac{NP}{2^6} \tag{5.5}$$

Toujours en considérant la table de routage de KAD, les contacts interrogés à l'étape suivante ont une résolution de la zone entourant la cible deux fois meilleure. Plus précisément, l'initiateur, alors loin de la cible, choisit les contacts initiant la localisation dans l'un de ces 10-bucket de plus haut niveau, ayant un index compris entre 5 et 15 sur le schéma 3.1. Ces premiers contacts, une fois sollicités, considèrent leurs buckets ayant un index de 0 pour sélectionner les contacts suivants. Dans le pire cas, l'identifiant de la cible correspond aux 10-buckets avec les index 1'-3"-7"'. Cela signifie que dans le pire cas la localisation progresse de 3 bits par étape grâce à la structure de la table de routage de KAD. Comme cette structure ne varie pas avec la profondeur de la table (une fois les premiers niveaux passés), ce raisonnement peut être répété à chaque étape. Enfin, pour être plus réaliste, nous devons ajouter les 2 bits dûs à la sélection de 3 contacts parmi

| n | $NP_n + NH$ | $P_n(H \geq 1)$ |
|---|---|---|
| 0 | 78145 | 0.00077 |
| 1 | 2461.5 | 0.0931 |
| 2 | 96.5 | 0.928 |
| 3 | 22.4 | 1 |

TABLE 5.2 – Probabilité de trouver au moins un honeypeer à la $n^{eme}$ étape de la localisation

10, ce qui conduit à une amélioration pessimiste de 5 bits par étape dans notre modèle comme l'indique la formule suivante :

$$NP_n = \frac{NP}{2^6 \times 2^{5 \times n}} \tag{5.6}$$

L'application numérique de ces probabilités avec les paramètres ($NP = 5 \times 10^6$,$NH = 20$) est présentée dans le tableau 5.2. Nous pouvons remarquer qu'au moins un Honeypeer sera retourné avec une probabilité de 10% après la première étape, et avec plus de 93% après la seconde. Ces résultats montrent que le routage de KAD est très efficace et assurent que les Honeypeers seront découverts pendant le processus de localisation de la référence ciblée, ce qui valide notre architecture. Des expériences sur le réseau KAD viendront confirmer ce modèle dans les sections suivantes. L'article [KCTHK09] présente également des résultats de mesures effectuées sur le processus de localisation de KAD qui sont conformes à notre modèle. Les auteurs ont ainsi notamment montré expérimentalement que les deux réplicats les plus proches d'un identifiant sont retrouvés dans plus de 90% des cas. Les auteurs concluent d'ailleurs :

– « the kad network works too well in some sense, the level of similarity between routing tables is very high »

## 5.2 Fonctionnalités d'HAMACK

Nous avons validé le fait qu'insérer des sondes à proximité d'une référence de la DHT permettait d'attirer toutes les requêtes de service à son intention. Nous allons maintenant décrire les différentes fonctionnalités que permet ce contrôle de la DHT avec pour objectif la supervision des accès aux contenus illégaux. Ces fonctionnalités sont :

– **Supervision passive** : supervise les requêtes de publication et de recherche pour des mots-clés ou fichiers ;
– **Éclipse de références** : retire les informations concernant des mots-clés ou des fichiers du moteur de recherche ;
– **Annonce d'appâts** : remplace les fichiers attachés à un mot-clé par de fausses références affichant de nombreuses sources ;
– **Insertion de honeypots** : remplace les pairs partageant un fichier par nos honeypots.

### 5.2.1 Supervision passive

La fonctionnalité la plus immédiate et discrète permise par notre architecture est la supervision passive de toutes les requêtes de service émises par les pairs à destination de la référence ciblée. Les Honeypeers enregistrent ainsi les informations des requêtes reçues puis les traitent normalement. Ce comportement est le moins intrusif pour le réseau car les services ne sont impactés en aucune manière par l'architecture de supervision dans le traitement des requêtes interceptées.

Plus précisément, lorsqu'une requête de publication de mot-clé (`KADEMLIA2_PUBLISH_KEY_REQ`) est capturée, un Honeypeer enregistre les informations suivantes contenues dans celle-ci :

- adresse IP de l'émetteur ;
- port de l'émetteur ;
- identifiant du mot-clé ;
- liste des fichiers publiés associés, avec pour chacun d'eux :
  - identifiant du fichier ;
  - liste d'étiquettes décrivant les propriétés du fichier (nom complet, taille) ;

Lorsqu'une requête de publication de fichier (`KADEMLIA2_PUBLISH_SOURCE_REQ`) est capturée, un Honeypeer enregistre les informations suivantes :

- adresse IP de l'émetteur ;
- port de l'émetteur ;
- identifiant du fichier publié ;
- identifiant de la source [15]
- adresse IP anonymisée de la source ;
- port de la source ;

Les sondes capturent également les requêtes de recherche de mot-clé `KADEMLIA2_SEARCH_KEY_REQ`) et de fichier `KADEMLIA2_SEARCH_SRC_REQ`), celles-ci comportent moins d'informations, typiquement les paramètres de l'émetteur et l'identifiant recherché. Cette fonctionnalité de supervision passive nous permet de connaître précisément l'activité du mot-clé ou du fichier étudié. Nous pouvons par exemple découvrir :

- Quels pairs partagent un fichier donné ?
- Quels pairs souhaitent télécharger un fichier donné ?
- Quels sont les nouveaux fichiers publiés pour un mot-clé ?
- Quels pairs cherchent des contenus relatifs à un mot-clé ?

### 5.2.2 Éclipse de références

La seconde fonctionnalité permet d'éclipser facilement des contenus du réseau. Pour éclipser un mot-clé ou un fichier, les clients exécutant les Honeypeers modifient la procédure de traitement des requêtes.

Tout d'abord, il est nécessaire de contourner deux contraintes affectant la capacité d'indexation d'un client. La première définit un nombre maximum de références que peut indexer un pair afin de ne pas donner trop de poids à un unique pair dans la DHT. La seconde définit le nombre maximum de références enregistrées pour un identifiant particulier afin que les contenus populaires (par exemple, le mot-clé « the ») ne soient pas sur-référencés sur un même pair. Si ces limites sont atteintes, le client refuse le service d'indexation et le demandeur doit alors contacter d'autres pairs pouvant ne pas appartenir à notre architecture. Une fois ces contraintes supprimées, les Honeypeers sont en mesure d'acquitter toute demande de publication, quelque soit la popularité des références ciblées.

Comme les Honeypeers attirent la quasi-totalité des requêtes de publication et affirment prendre en charge l'indexation de la référence ciblée, il leur suffit ensuite de nier les requêtes de recherche s'y rapportant pour éclipser le contenu du réseau. De ce fait le contenu, bien que partagé et publié par les pairs, restera inaccessible par le moteur de recherche basé sur la DHT. Les échanges de messages sont décrits par le schéma 5.2. Cette fonctionnalité est intéressante car elle permet de faire disparaître les références malveillantes du réseau, et ce faisant, d'une part

15. l'émetteur et la source annoncée peuvent différer quand l'émetteur ne peut recevoir les requêtes entrantes et doit passer par un intermédiaire

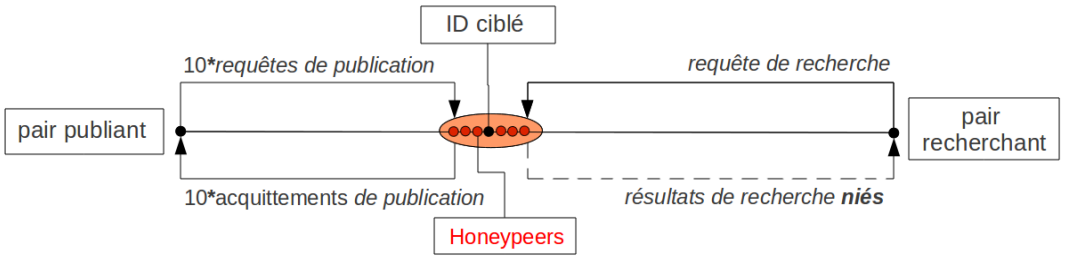


FIGURE 5.2 – HAMACK éclipsant les résultats d'un mot-clé

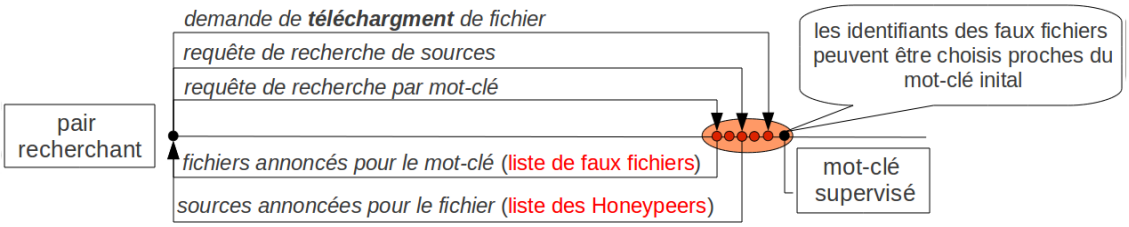


FIGURE 5.3 – HAMACK falsifiant les résultats d'un mot-clé pour promouvoir les honeypots

d'en protéger les utilisateurs en évitant leur accès, et d'autre part d'éviter leur concurrence avec les appâts du honeypot.

### 5.2.3 Annonce de fichiers appâts et honeypots terminaux

Pour qu'un honeypot P2P soit performant, les fichiers annoncés doivent éviter toute concurrence et sembler attractifs en présentant un grand nombre de sources. En effet, le nombre de sources pour un contenu donné est une information essentielle permettant à un utilisateur de trier les résultats d'une recherche. Les fichiers présentant un nombre de sources élevé sont privilégiés car ceux-ci sont populaires, plus fiables, et seront téléchargés plus rapidement.

L'annonce de fichiers appâts et la promotion de honeypots sont les fonctionnalités les plus abouties d'HAMACK car elle permet une supervision très précise de l'accès à un contenu, en interceptant l'ensemble des requêtes émises depuis la recherche de mot-clé jusqu'au téléchargement du faux fichier comme présenté par le schéma 5.3. La supervision des accès des pairs aux faux fichiers proposés par notre architecture se fait en deux étapes.

La première étape est similaire à l'éclipse et consiste à prendre le contrôle de la référence à superviser en attirant toutes les requêtes de publication à destination de celle-ci. Ensuite, au lieu de nier les requêtes de recherche, les Honeypeers répondent à ces dernières en incluant de faux fichiers dont les paramètres (nom, taille, et surtout, nombre de sources...) sont entièrement maîtrisés par l'architecture. La falsification des réponses peut être partielle ou totale si elle est couplée à l'éclipse des autres fichiers, selon la configuration souhaitée. Cette fonctionnalité permet d'afficher un grand nombre de sources pour assurer une bonne visibilité au pot de miel sans recourir à une grande quantité de machines annonçant les appâts.

L'étape finale consiste à capturer la demande de téléchargement lorsqu'un faux fichier parmi la liste proposée est sélectionné. Pour cela, notre architecture doit être capable d'attirer les

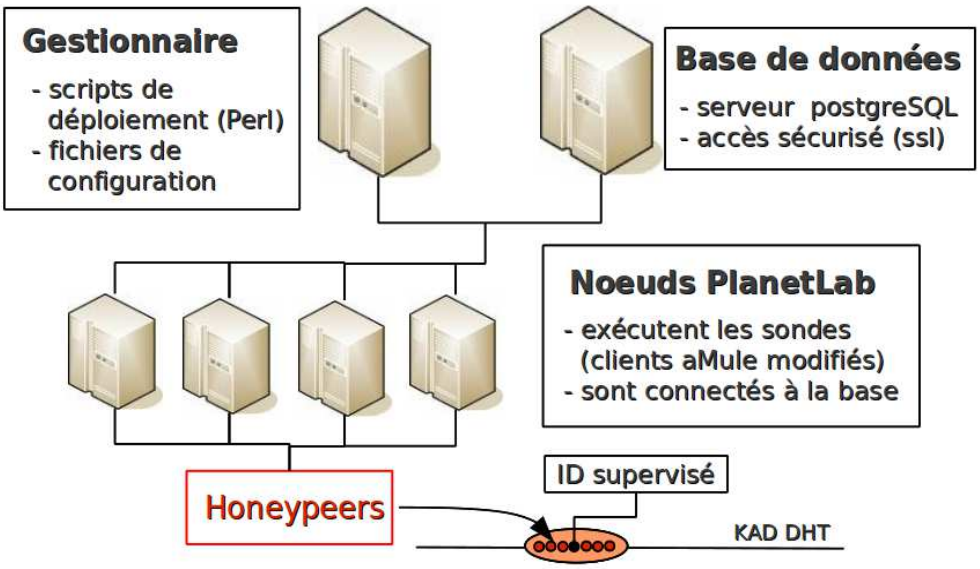


FIGURE 5.4 – Architecture réseau du Honeynet

recherches de sources pour les fichiers proposés et d'y répondre avec les références des Honeypeers, formant ainsi un ensemble de honeypots distribués ou Honeynet. Sans que cela soit nécessaire, une optimisation consiste à générer les KADIDs des fichiers créés pour rester extrêmement proches (96 bits) du mot-clé supervisé. Ainsi, chaque recherche de sources est attirée de la même manière que la recherche de mot-clé initiale et les mêmes Honeypeers peuvent être utilisés.

Enfin, les sources renvoyées dans les réponses sont les Honeypeers eux-mêmes (KADID, IP, port). L'architecture est ainsi assurée de capturer les requêtes de téléchargement. Cette organisation spécifique permet de s'assurer de l'intention d'un utilisateur malveillant. En capturant l'ensemble des requêtes émises, son comportement peut être vérifié depuis le début, par la recherche de mot clé, jusqu'à la demande de téléchargement finale au travers de l'annonce des faux fichiers. La partie suivante décrit l'implantation de notre architecture et les résultats obtenus lors de son déploiement sur le réseau KAD.

## 5.3 Évaluation de l'architecture

### 5.3.1 Implantation

L'architecture de supervision HAMACK a été implantée avec la collaboration de Frédéric Beck et de Juan Pablo Timpanaro. Le schéma 5.4 illustre les différents composants constituant notre architecture. Nous détaillons notamment les deux principaux que sont les clients exécutant les Honeypeers et la base de données enregistrant les informations capturées.

#### 5.3.1.1 Les Honeypeers

La principale difficulté d'implantation de notre architecture vient de la récente protection des clients KAD empêchant l'utilisation d'adresses IP identiques ou proches (même sous-réseau) parmi les contacts découverts par le processus de localisation. Ainsi, pour être fonctionnels,

les Honeypeers déployés sur une même référence doivent présenter des adresses IP publiques différentes et dont seules deux peuvent appartenir à un même sous réseau.

Pour répondre à ces contraintes, notre solution de supervision est distribuée. Nous avons choisi d'exécuter notre Honeynet sur le réseau PlanetLab Europe, comme décrit par le schéma 5.4. Les Honeypeers exécutent donc une version modifiée du client aMule sur des nœuds PlanetLab. Pour être compatible avec une exécution en mode console, nous utilisons la version démon (amuled) du client qui présente les avantages d'être légère (sans interface graphique) et pilotable à distance.

Le code du client a été modifié pour implanter l'ensemble des fonctionnalités décrites dans la section précédente. En particulier, les fonctions traitant les paquets reçus et plusieurs paramètres de KAD ont été modifiés. Les références à superviser, les fonctionnalités souhaitées (supervision passive, pots de miel) ainsi que d'autres paramètres sont lus depuis une base de données au démarrage du client.

Des scripts Perl permettent d'automatiser le déploiement et l'administration de l'architecture sur PlanetLab. Ces scripts sont rassemblés et exécutés depuis l'ordinateur principal gérant le Honeynet. L'écriture des scripts a été facilitée par l'architecture homogène de PlanetLab où chaque Honeypeer exécute le même client dans le même environnement.

#### 5.3.1.2 La base de données

Le second composant de notre architecture est la base de données enregistrant les informations capturées par les Honeypeers. La base de données est implantée par un serveur PostgreSQL dont l'accès est sécurisé par SSL afin d'empêcher toute connexion extérieure aux Honeypeers. La plupart des accès à la base de données se font en écriture (lors de la collecte des requêtes) mais les Honeypeers sont également amenés à consulter quelques informations pour paramétrer leurs fonctions. Lors de leur initialisation, les Honeypeers lisent dans la base de données les informations suivantes : les références à superviser, les autres Honeypeers déjà actifs (en vue de leur collaboration), les fonctions de supervision demandées ou encore la liste des faux fichiers à annoncer en cas de recours aux fichiers appâts.

Les différentes tables ont été conçues pour pouvoir anonymiser facilement les adresses IP récoltées sans toutefois perdre trop d'informations. Lorsqu'une nouvelle adresse IP est observée par le Honeynet, notre base de données lui attribue un indice unique pendant toute l'exécution. A l'issue de la phase d'observation, la correspondance géographique approximative de chaque adresse IP est obtenue grâce à la base de données « GeoLite City [16] ». Enfin, la base de données est anonymisée en supprimant le contenu des adresses IP, ne gardant que leur index et l'information de géolocalisation.

### 5.3.2 Optimisation de l'architecture

Le premier paramètre étudié lors de notre évaluation est l'impact du temps sur l'efficacité de l'architecture afin de connaître le temps minimum d'exécution de l'architecture lui permettant d'être parfaitement fonctionnelle. En effet, les Honeypeers lorsqu'ils rejoignent le réseau, comme n'importe quel autre client, ont besoin d'un certain temps, appelé phase d'initialisation ou d'amorçage, durant lequel ils s'annoncent activement afin d'être connus des autres pairs du réseau. A plus long terme, KAD privilégiant les contacts stables, une longue période d'exécution ininterrompue pourrait également impacter l'efficacité de l'architecture.

Le graphique 5.5 montre, pour chaque intervalle de 6 heures, le nombre de requêtes de publication reçues par HAMACK pendant deux jours. Les requêtes sont capturées après un quart

16. `http://www.maxmind.com/app/geolitecity`

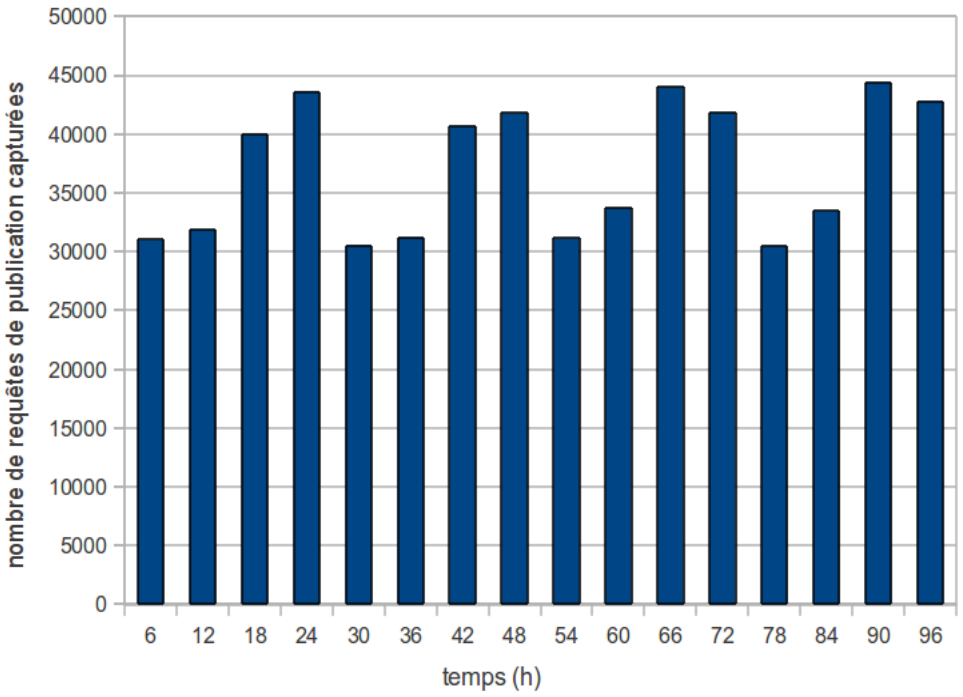


FIGURE 5.5 – Impact du temps d'exécution sur l'efficacité d'HAMACK

d'heure réservé à l'initialisation de l'architecture. Nous pouvons clairement observer que le temps d'exécution n'a pas influence sur l'efficacité de l'architecture. Nous pouvons également remarquer que le nombre de requêtes reçues durant l'après-midi ($\sim$ 40000) entre 12h-18h ou 18h-00h (GMT+1) est plus élevé que durant la matinée ($\sim$ 30000) entre 00h-06h ou 06h-12h. Bien que le nombre de pairs à l'échelle du réseau soit stable dans le temps, la capture des requêtes étant réalisée à travers l'observation de mots-clés définis dans un alphabet et une langue, celle-ci est liée à une zone géographique particulière (ici l'Europe), ce qui explique l'effet journalier. Il est donc nécessaire de considérer une journée complète pour ne pas être soumis aux variations horaires lors de l'évaluation des performances de l'architecture.

Parmi les paramètres pouvant affecter l'efficacité de l'architecture, connaître le nombre de Honeypeers déployés pour superviser une même référence est important afin de limiter le nombre d'instances nécessaires et ainsi limiter la consommation de ressources. Le nombre minimum de Honeypeers devrait être de 10, c'est à dire le facteur de réplication de KAD en dessous duquel on ne peut capturer toutes les requêtes. Cependant, comme plusieurs requêtes de services sont envoyées en parallèle, il est fréquent que plus de 10 requêtes soient émises avant que le service ne soit réalisé. Par conséquent, nous considérons un nombre minimum de 15 Honeypeers nécessaires pour superviser une référence. Le graphique 5.6 montre le nombre de requêtes capturées alors que le nombre de Honeypeers varie entre 15 et 25. Les résultats montrent que l'utilisation de 20 Honeypeers améliore significativement l'efficacité d'HAMACK par rapport à 15. En revanche l'utilisation de 25 Honeypeers ou plus est inutile.

Après avoir étudié les principaux paramètres d'exécution, nous avons évalué trois autres paramètres affectant le comportement des Honeypeers. Nous avons tout d'abord mesuré l'impact de leur collaboration. Celle-ci force les Honeypeers à se connaître explicitement grâce à la base de données et à se promouvoir lorsqu'ils reçoivent une requête de localisation concernant la

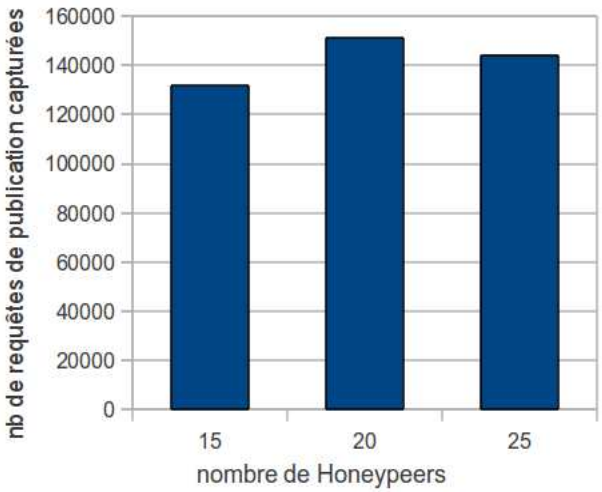


FIGURE 5.6 – Impact du nombre de Honeypeers déployés sur l'efficacité d'HAMACK

référence supervisée. Sans collaboration, les Honeypeers n'ont pas de connaissance a priori les uns des autres et se découvrent et se propagent naturellement par les mécanismes KAD. Nous pouvons constater dans le graphique 5.7 que cette modification a un impact positif sur le nombre de requêtes de publication capturées sur une journée (+23%).

Ensuite nous avons étudié l'influence du nombre de contacts retournés lors d'une demande de localisation. En effet, ce paramètre peut varier selon le type de service préparé. Les dernières versions des clients KAD (eMule 0.49c et aMule 2.2.4) vérifient que le nombre exact de contacts attendus est retourné. Ainsi, ces clients rejettent toute réponse contenant plus que 4 contacts potentiels pour un service de publication ou de recherche. Cependant, ces versions étant très récentes au moment des tests (moins de 2 mois), la plupart des clients n'implantaient pas cette contrainte. Par ailleurs, augmenter le nombre de contacts par réponse de manière à propager l'ensemble des Honeypeers en une seule réponse améliore leur visibilité. Il est donc intéressant d'évaluer la meilleure stratégie. Le graphique 5.8 montre que le Honeynet capture davantage de requêtes de publication (+45%) en respectant les dernières contraintes implantées. Cela signifie qu'une partie significative du réseau P2P met à jour son client dès que possible et, par conséquent, toute nouvelle contrainte implantée dans les clients officiels doit être immédiatement considérée.

Enfin, nous avons évalué le fait d'annoncer activement les Honeypeers dans le réseau en émettant périodiquement des requêtes de localisation vers des identifiants aléatoires de la DHT. Le graphique 5.9 montre que cette modification est inutile et n'apporte aucun gain à l'architecture, les algorithmes de routage et de maintenance des tables mis en œuvre dans KAD étant déjà suffisamment performants pour que les Honeypeers soient bien référencés.

### 5.3.3 Évaluation de l'attractivité des Honeypeers

Évaluer l'efficacité globale de l'architecture est très délicat. Une approche naïve consisterait à instrumenter chaque pair du réseau afin de connaître la proportion effective de requêtes capturées par HAMACK. Nous avons néanmoins conçu deux expériences permettant de fournir un indicateur sur son efficacité. Nous avons déployé l'architecture composée de 20 Honeypeers sur un mot-clé du réseau KAD, activant la collaboration et retournant 4 contacts par requête de localisation.

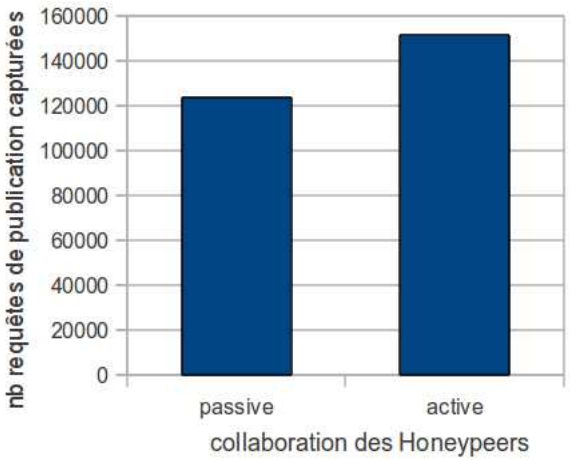


FIGURE 5.7 – Impact de la collaboration entre Honeypeers

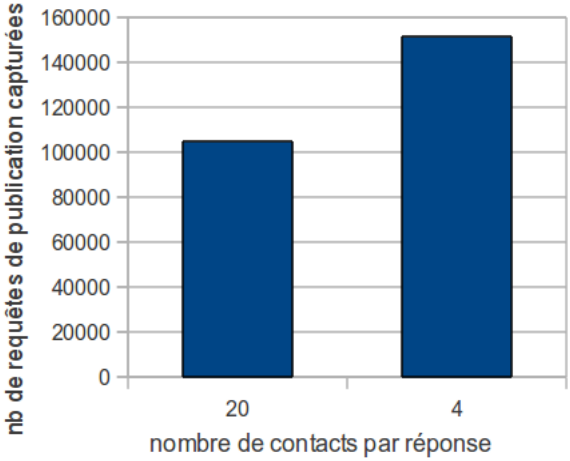


FIGURE 5.8 – Impact du nombre de contacts annoncés dans les réponses

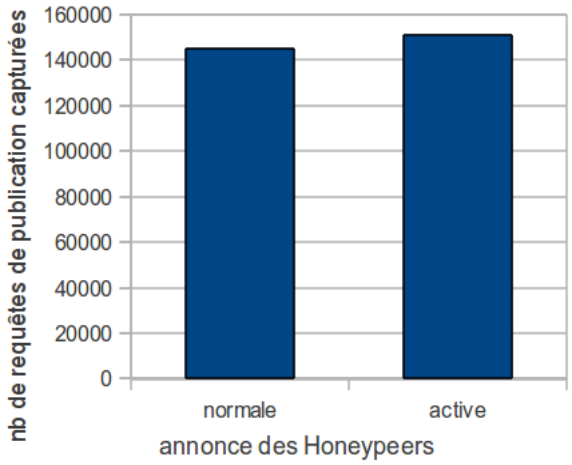


FIGURE 5.9 – Impact de l'annonce active des Honeypeers

La première expérience réalisée a pour objectif de vérifier que l'architecture parvient bien à attirer toutes les requêtes pour l'identifiant supervisé. Pour cela, nous avons publié un mot-clé supervisé par HAMACK depuis 8 pairs différents que nous avons instrumentés. Pour éviter toute mesure biaisée par des conditions initiales similaires, chacune des 8 sources a une place aléatoire dans la DHT et a rejoint le réseau depuis une liste différente de contacts. Étant donnée cette configuration, nous comparons, pour chaque publication, le nombre de requêtes émises par les différentes sources et celui constaté par HAMACK pour chacune de celles-ci. Le graphique 5.10 montre que toutes les requêtes émises sont capturées par les Honeypeers, comme le laissait supposer l'analyse de la procédure de recherche de KAD. Le nombre de requêtes émises (entre 10 et 12) correspond au facteur de réplication (10) nécessaire pour publier un fichier, plus quelques requêtes supplémentaires émises avant la fin du service.

La seconde expérience consiste à compter le nombre de requêtes de publication capturées par HAMACK venant d'un même pair et pendant une courte période de temps. En effet, chaque donnée indexée dans KAD est prise en charge par 10 pairs constituant autant de réplicats. Sans au moins dix réponses affirmatives, la publication n'est pas satisfaite. Ainsi, chaque fois que l'architecture reçoit moins de 10 requêtes répliquées durant l'intervalle de temps d'une publication, des requêtes ont potentiellement été manquées. Le graphique 5.11 compte le nombre de publication pour chaque valeur de réplication observée. Les résultats sont très satisfaisants. En effet, peu de publications sont observées avec moins de 8 réplicats (8.9%) et comme attendu, la grande majorité des réplicats capturés se situe autour de 10.

Finalement, nous avons mesuré la distribution des requêtes capturées sur l'ensemble des 20 Honeypeers supervisant une même référence pendant une journée. En ordonnant les Honeypeers par rapport à leur distance XOR à la cible, le graphique 5.12 montre que la distribution de la charge entre les Honeypeers est liée à cette distance : plus un Honeypeer est proche de la cible, plus il reçoit de requêtes. Bien que les Honeypeers soient plus proches que n'importe quel autre pair de la cible du fait des 96 bits communs avec la cible, et malgré leur collaboration, ils restent néanmoins en concurrence pour attirer les réplicats. Ainsi, la liberté de choisir les 32 bits restant est suffisante pour procurer un gain d'attractivité significatif aux Honeypeers qui se rapprochent de la cible, ce qui montre que l'algorithme de KAD parvient toujours à localiser les pairs les plus proches même dans ces conditions extrêmes. Ayant la connaissance des identifiants

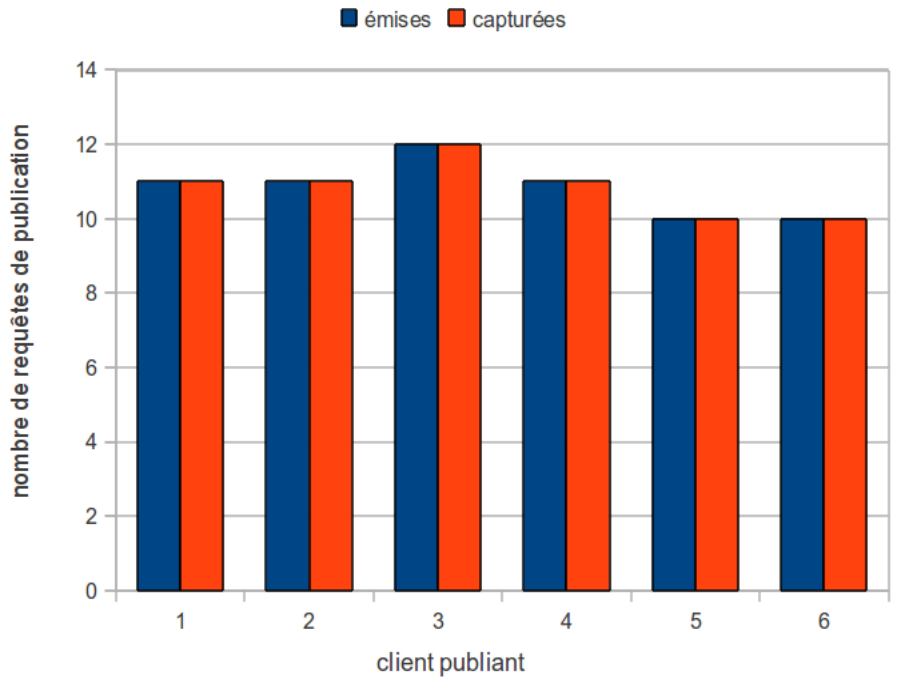


FIGURE 5.10 – Nombre de requêtes de publication capturées pour les différentes sources

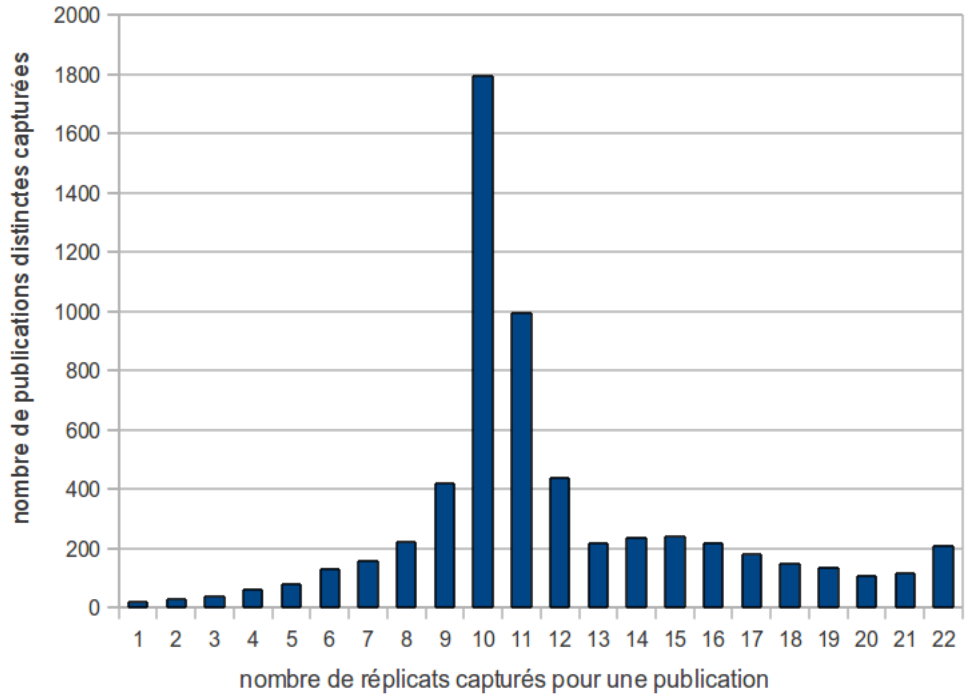


FIGURE 5.11 – Nombre de publications observées pour un facteur de réplication donné

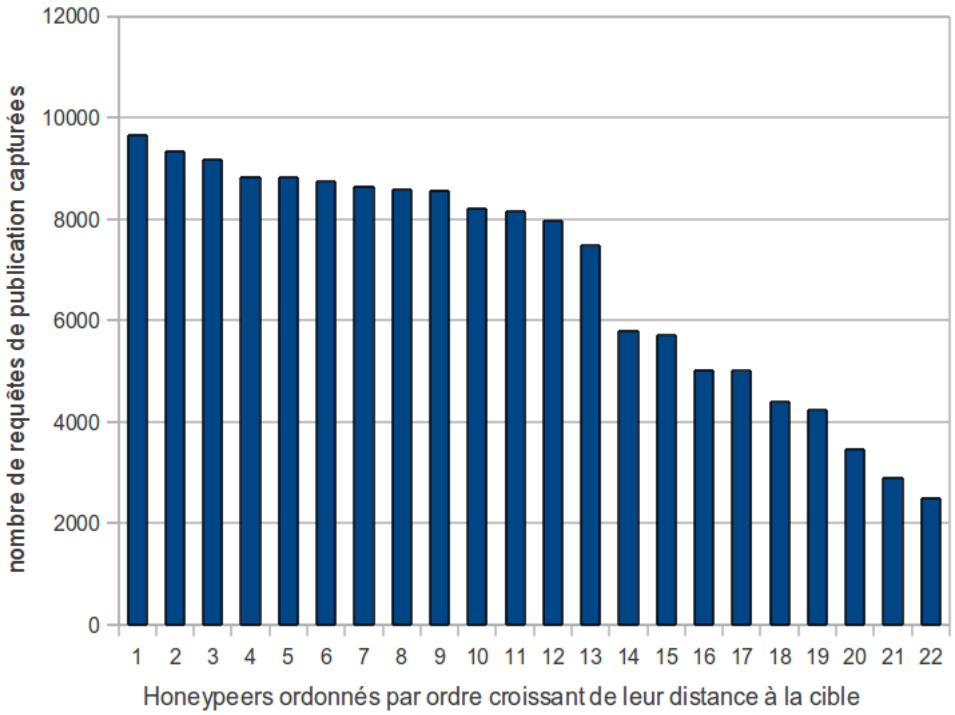


FIGURE 5.12 – Distribution de la charge sur les Honeypeers

des Honeypeers, nous avons remarqué que les Honeypeers avec les identifiants entre 1 et 13 ont choisi le 97ème bit conformément à la cible alors que les autres entre 14 et 22 ont choisi l'autre valeur, ce qui est visible graphiquement car l'efficacité de ces derniers décroît significativement avec la perte d'un bit commun. Ces résultats indiquent également que les pairs normaux, étant en comparaison très éloignés de la cible, ont peu de chance de recevoir des requêtes au détriment des Honeypeers.

### 5.3.4 Expérimentation des fonctionnalités avancées de supervision

Nous avons ensuite testé les fonctionnalités avancées de notre architecture. L'expérience consiste à éclipser les fichiers existants pour un mot-clé populaire et à falsifier son index avec des fichiers appâts servant les pots de miel. Pour cela, nous avons choisi de cibler le mot clé « spiderman » pendant une journée, éclipsant les fichiers indexés et les remplaçant par 4 faux fichiers affichant un nombre différent de sources, deux apparaissant comme populaires et deux autres comme étant faiblement partagés. Les résultats d'une recherche effectuée par un client KAD normal pendant ce déploiement sont présentés par la capture d'écran 5.3.4. Nous pouvons observer que le Honeynet réussit parfaitement à éclipser les véritables références et à les remplacer par les 4 faux fichiers proposés.

Pour chacun des fichiers appâts, nous avons ensuite mesuré la proportion de recherches de source uniques émises par des pairs distincts, ce qu'illustre le diagramme 5.14. Nous pouvons constater que les deux fichiers populaires sont choisis en priorité par 96% des utilisateurs. Ce résultat confirme notre hypothèse initiale sur l'importance que revêt le nombre de sources affiché quand les utilisateurs doivent choisir entre plusieurs fichiers et, par conséquent, que sa maîtrise est nécessaire pour créer un pot de miel efficace. Cette dernière expérience illustre l'efficacité de

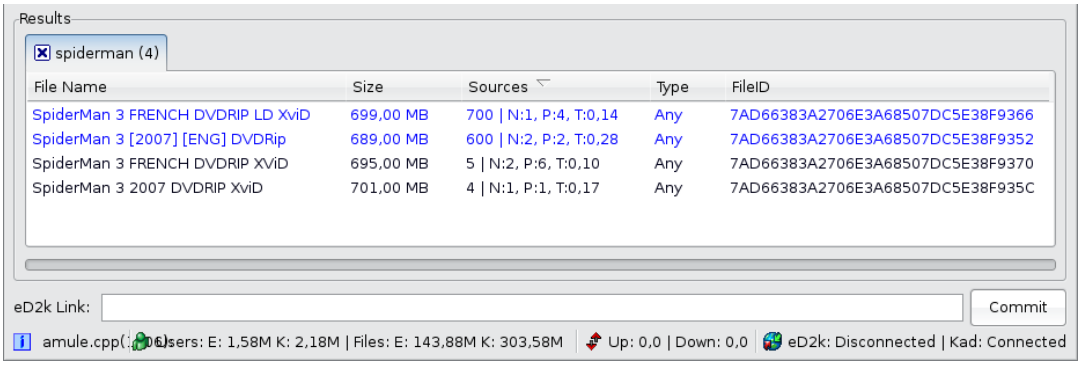


FIGURE 5.13 – Résultat d'une recherche sur "spiderman" ciblé par le Honeynet avec 4 faux fichiers

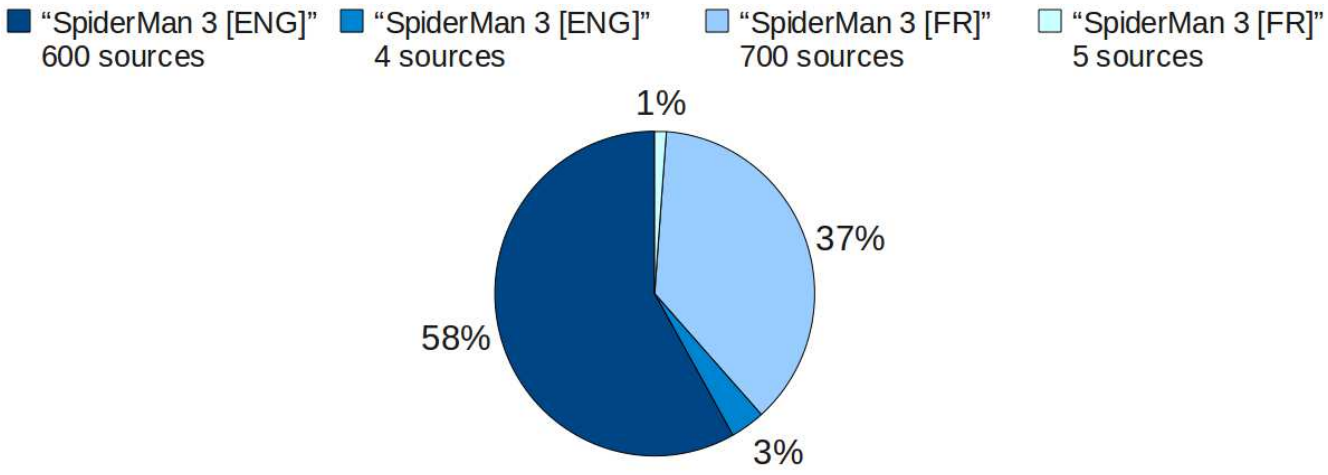


FIGURE 5.14 – Proportion des recherches de sources reçues pour chaque faux fichier

notre approche et le contrôle qu'elle permet sur le réseau KAD.

## Conclusion

Nous présentons dans ce chapitre notre architecture de supervision pour le réseau P2P KAD, HAMACK capable de superviser précisément et de contrôler l'activité des contenus diffusés à travers l'indexation des informations (mots-clés et fichiers) servant à leur partage. Celle-ci repose sur l'utilisation conjointe de pots de miel et de sondes distribuées permettant de prendre localement le contrôle de la DHT en contournant les mécanismes de protection présentés dans le chapitre précédent.

En concevant HAMACK, notre objectif était de créer une architecture capable de fournir une supervision fiable et performante des accès aux contenus dans le réseau P2P KAD. L'utilisation de pots de miel classiques, où plusieurs clients annoncent des fichiers appâts, n'est pas satisfaisante car elle ne peut atteindre une bonne visibilité, au regard du nombre de sources estimé, sans d'importants moyens. D'autre part, superviser l'accès à des fichiers existants est sujet à de nombreux faux positifs car un utilisateur peut accéder à un contenu par différents mots-clés dont

certains sont sans rapport avec celui-ci du fait de la pollution du réseau.

Notre solution, en prenant le contrôle du mécanisme d'indexation malgré les dernières protections contre l'attaque Sybil, est capable de générer et d'annoncer des fichiers-appât très attractifs avec peu de ressources, tout en éclipsant les autres références existantes concurrentes. En dehors des références ciblées, notre architecture n'est absolument pas intrusive pour le reste du réseau. Par ailleurs, elle permet également de superviser de manière passive toutes les requêtes destinées à un contenu spécifique (mot clé ou fichier) indexé sur la DHT. En capturant l'ensemble de ces requêtes depuis la recherche de mots-clés jusqu'à la demande de téléchargement d'un fichier, nous pouvons ainsi prouver l'intérêt des pairs pour un contenu spécifique.

La validation formelle puis les évaluations effectuées sur KAD suite à la mise en œuvre de l'architecture grâce à PlanetLab ont montré l'efficacité de notre approche, celle-ci étant capable d'attirer les requêtes émises pour une référence donnée tout en nécessitant peu de ressources (20 nœuds) et permet ainsi de contrôler l'indexation d'un mot clé très populaire, comme nous l'avons montré pour « spiderman ». Le chapitre suivant applique notre architecture à la supervision de contenus malveillants diffusés dans KAD.

# Chapitre 6

# Application à la supervision des contenus pédophiles

**Sommaire**



## Introduction

Notre architecture de supervision HAMACK a été conçue et implantée dans le contexte du projet ANR MAPE « Measurement and Analysis of Peer-to-peer Exchanges for pedocriminality fighting and traffic profiling » ayant pour but l'étude et la caractérisation des activités à caractère pédophile dans les réseaux P2P. Dans ce cadre, ce chapitre présente les résultats d'un déploiement réalisé pendant 2 semaines consécutives sur le réseau KAD permettant d'étudier l'activité de contenus à caractère pédophile échangés sur ce réseau à travers la supervision de l'activité de mots-clés relatifs.

Afin de ne pas interférer avec une supervision du réseau pouvant être menée par les forces de l'ordre, nous avons limité les fonctionnalités d'HAMACK aux fonctionnalités passives, supervisant ainsi l'activité des mots-clés sans en modifier les références (ni éclipse, ni annonce de fichiers appâts). L'activité de 72 mots-clés a ainsi été supervisée pendant 2 semaines du 19 octobre au 02 novembre 2010. Parmi ces mots-clés, la moitié sont à caractère pédophile et l'autre est constituée de mots-clés courants à des fins de comparaison. Comme l'objectif n'est pas ici de prendre le contrôle d'un mot-clé mais de capturer au moins une des requêtes émises pour chaque service, seules 5 sondes ont été déployées à proximité de l'identifiant du mot-clé au lieu des 20 nécessaires à son contrôle. Nous avons montré que le nœud le plus proche d'une référence est retrouvé dans 90% des cas ce qui a également été mesuré par les auteurs de [KCTHK09]. Insérer 5 Honeypeers permet de réduire encore davantage le taux de requêtes manquées.

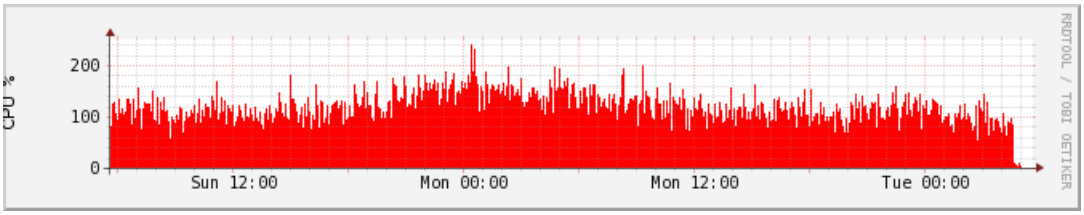


FIGURE 6.1 – Consommation processeur relevée pour 360 sondes actives

Ce chapitre présente tout d'abord le contexte de la collecte de données puis leur analyse descriptive par l'interrogation de la base de données relationnelle associée.

## 6.1 Collecte des données

### 6.1.1 Environnement expérimental

#### 6.1.1.1 Ressources utilisées

La supervision des 72 mots clés a nécessité l'exécution de 360 Honeypeers, à raison de 5 sondes par mot-clé. Celles-ci sont exécutées sur des nœuds du réseau de recherche PlanetLab. Nous avons ainsi distribué l'ensemble des sondes sur 35 nœuds PlanetLab différents, chaque ensemble de 5 nœuds supervisant le même ensemble de 10 à 11 mots-clés. Cette distribution répond à deux contraintes. La première concerne la disponibilité des sondes constituant l'architecture de supervision. Cette disponibilité est liée à celle des nœuds PlanetLab les exécutant or ceux-ci sont fréquemment indisponibles. Les sondes alors exécutées sur le nœud défaillant doivent être rapidement redéployées sur un nouveau nœud sain. Réduire le nombre de sondes instanciées par nœud permet de limiter le nombre de sondes indisponibles entre le moment où une panne intervient et sa prise en charge. Par ailleurs, le réseau PlanetLab étant mutualisé, peu de ressources sont disponibles par nœud pour une expérience donnée. Distribuer la supervision sur davantage de nœuds génère ainsi une charge plus conciliante pour la plateforme expérimentale. Les graphiques 6.1, 6.2 et 6.3 montrent respectivement la consommation des ressources processeur, mémoire et réseau de notre architecture de supervision pendant la campagne de mesure, telles que relevées par l'interface de supervision de PlanetLab : CoMon[17]. Les diagrammes montrent que la consommation de ressource est stable et modérée sur l'ensemble du temps d'exécution. La consommation agrégée des 360 sondes utilise l'équivalent d'un cœur de processeur, 3Go de mémoire vive et 2Mbit/s de bande passante descendante ce qui rendrait, d'un strict point de vue de la consommation de ressources, un déploiement centralisé possible sur un seul ordinateur de bureau.

L'autre composant constituant notre architecture de supervision est la base de données relationnelle enregistrant les informations capturées par les sondes. Celle-ci est mise en œuvre par un serveur postgreSQL 8.3 hébergé au sein du LHS[18] au LORIA. Les graphiques 6.4 montrent que la bande passante consommée pour l'enregistrement des informations dans la base de données est modérée (environ 1.1Mbit/s) et relativement constante malgré un léger effet journalier pouvant induire une variation inférieure à 10% dans le trafic généré.

17. `http://comon.cs.princeton.edu/`
18. Laboratoire de Haute Sécurité `http://lhs.loria.fr`

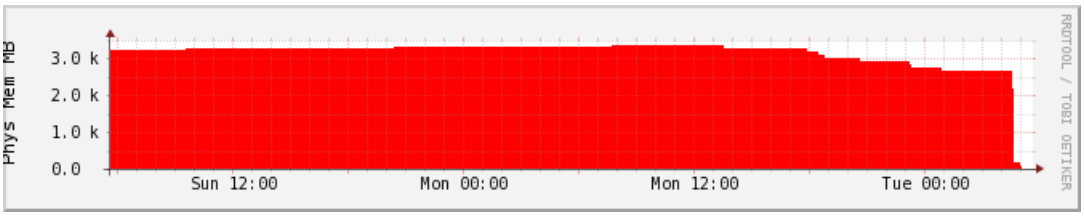


FIGURE 6.2 – Consommation mémoire relevée pour 360 sondes actives

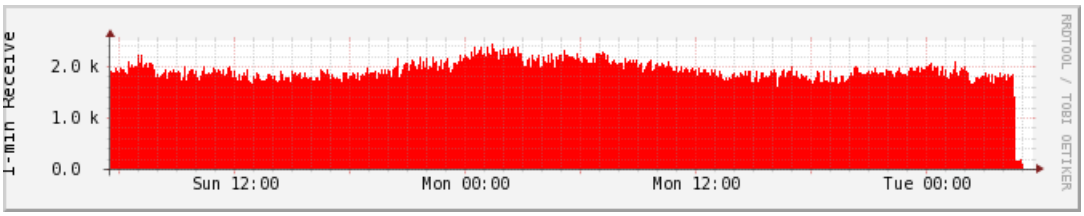


FIGURE 6.3 – Consommation de bande passante descendante relevée pour 360 sondes actives

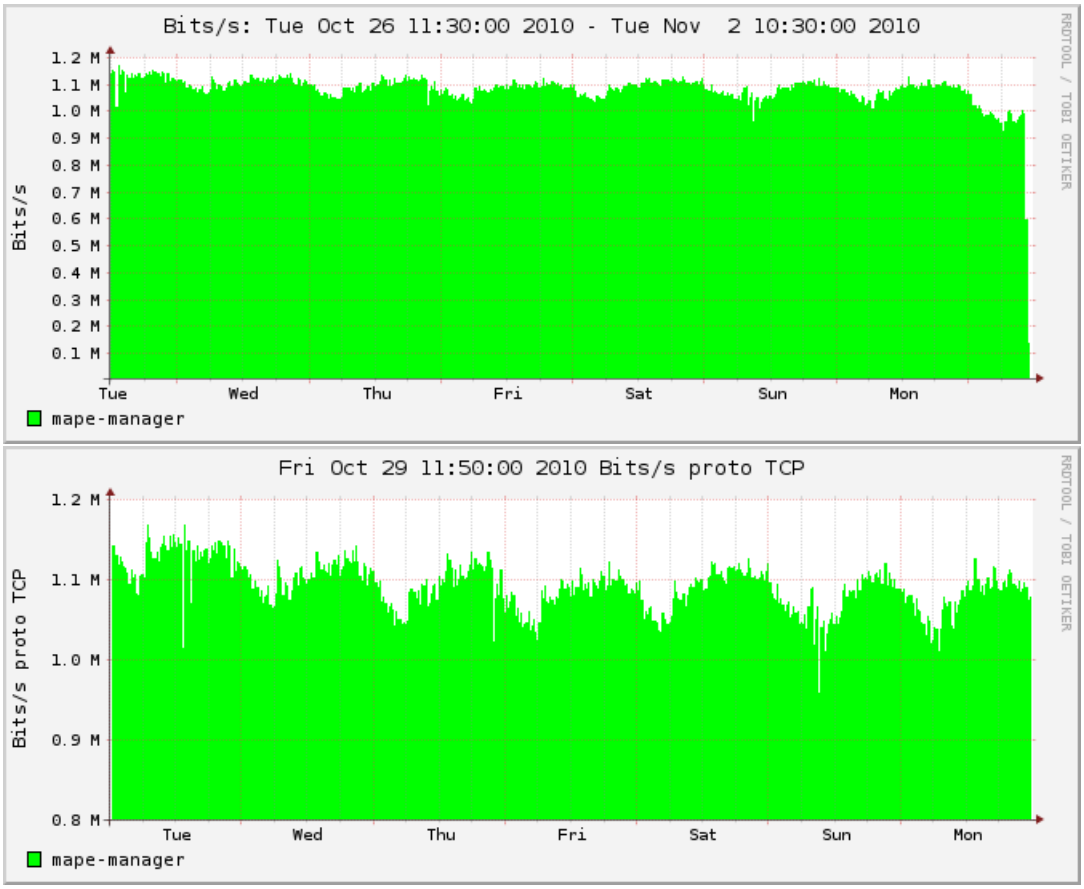


FIGURE 6.4 – Consommation de bande passante descendante relevée pour la base de données

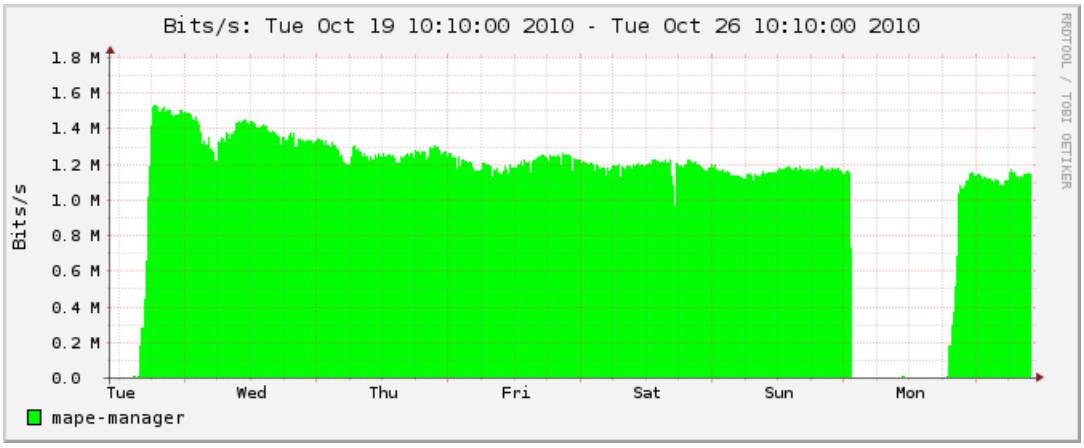


FIGURE 6.5 – Trafic descendant relevé pour la base de données pendant une semaine

#### 6.1.1.2 Fiabilité

Chaque composant de l'architecture peut être affecté par des problèmes de fiabilité. D'une part, les nœuds PlanetLab exécutant les sondes sont régulièrement indisponibles ce qui implique une supervision active de l'architecture et un redéploiement des sondes affectées en cas de défaillance d'un nœud. Ainsi, au long des deux semaines de mesure et parmi les 35 machines utilisées, 5 ont dû être remplacées. Planetlab, bien que fonctionnel dans le cas de notre architecture, s'est cependant montré peu adapté à de longues expériences pour lesquelles la disponibilité des nœuds est importante.

Le second point critique de l'architecture est la fiabilité de la base de données. La fiabilité du serveur postgreSQL n'a pas été mise en défaut durant le déploiement. En revanche, la connectivité de l'ordinateur l'exécutant fut momentanément interrompue. L'ensemble des sondes ont ainsi été déconnectées de la base de données ce qui a nécessité leur redéploiement une fois la connexion internet restaurée. Cet incident est notamment visible sur le graphique 6.5 où l'on peut remarquer l'annulation brutale du trafic à destination de la base de données pour la journée de lundi, suivi d'un redéploiement visible par une reprise du trafic dans la même journée.

### 6.1.2 Base de données

Les informations collectées par les sondes sont stockées dans différentes tables de la base de donnée décrites ci-après. Une partie des tables n'est utilisée que pendant le déploiement de l'architecture afin de partager des informations entre les sondes (tables sybils, replicats, keywords). De plus la table « fakefiles » n'est utilisée que dans le cadre d'une supervision active falsifiant l'indexation des mots-clés. Elle contient alors les fausses références de fichiers retournées par les sondes lorsqu'elles sont sollicitées. Comme nous avons considéré une supervision passive pour cette expérience, cette table n'est pas utilisée. Le diagramme de classes 6.6 présente les relations liant les différentes tables utiles à cette expérience. Chacune des tables est décrite en détails en annexe A.

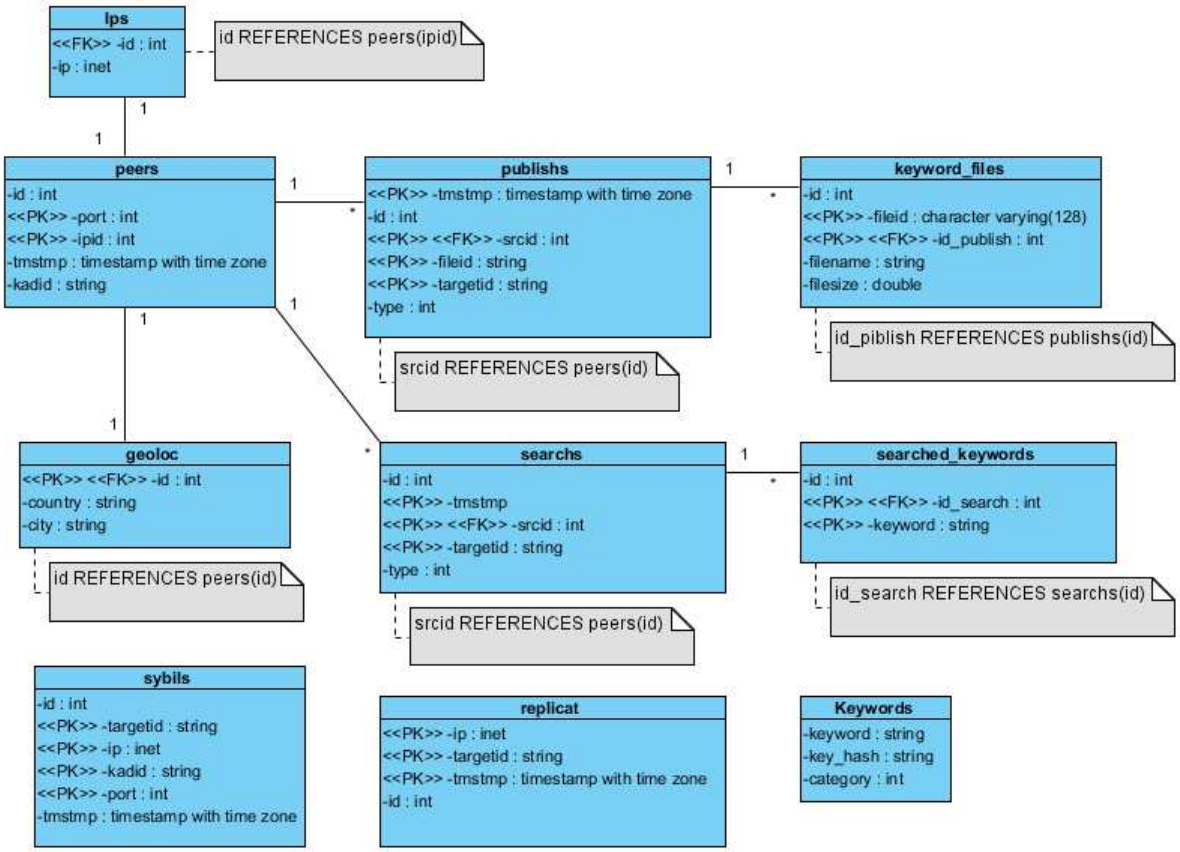


FIGURE 6.6 – Diagramme de classes de la base de données

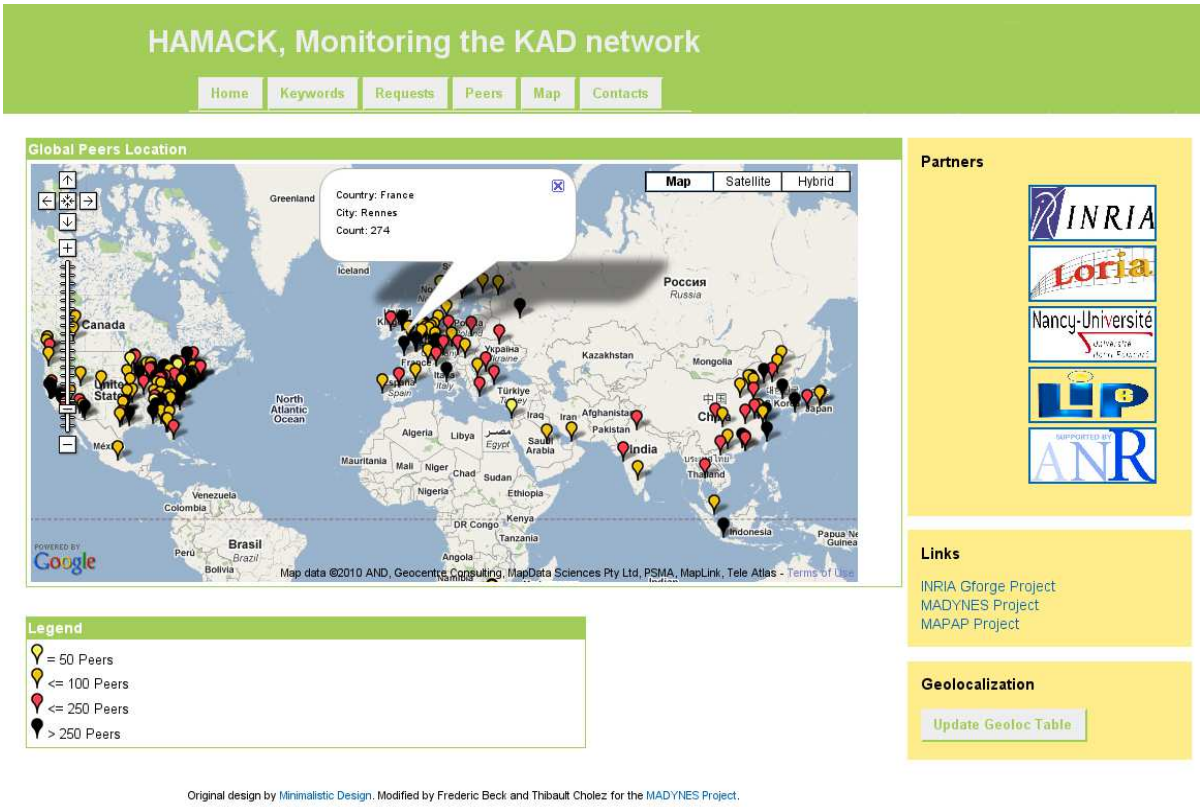


FIGURE 6.7 – Illustration des données de géolocalisation par une interface graphique

### 6.1.3 Présentation des données

Afin de faciliter la présentation des données collectées, une interface graphique a été réalisée en PHP et propose une vue synthétique de chaque table en présentant des statistiques générales. La capture d'écran 6.7 illustre ainsi l'onglet présentant les différentes données concernant la géolocalisation des pairs.

## 6.2 Quantification des contenus pédophiles

### 6.2.1 Préparation des données collectées

Les deux semaines de supervision du réseau KAD ont permis la collecte de données dans les proportions indiquées par le tableau 6.1. Ces données représentent un total de 28Go une fois stockées dans la base. Une lecture de ces résultats généraux corroborent les précédentes observations réalisées sur KAD. On note ainsi que le nombre de pairs et d'adresses IP distincts sont proches ($12842034/9244808 = 1,39$) car la majorité des pairs utilise un identifiant unique et dispose d'une adresse IP publique. Cependant, le nombre de pairs partageant une même adresse IP sur la durée des mesures n'est pas négligeable. Un ordre de grandeur sépare le nombre de requêtes de publications observées par rapport au nombre de recherches ($35807630/1117418 = 32,04$). Ceci a déjà été constaté dans KAD et s'explique par le fait que les publications sont générées automatiquement par la fonction de partage de fichiers alors que les recherches sont générées manuellement par les utilisateurs. Parmi celles-ci, seules 254616 requêtes de recherche

| donnée observée | nombre d'occurrences |
|---|---|
| mots-clés | 72 |
| pairs | 12842034 |
| adresses IP | 9244808 |
| publications | 35807630 |
| fichiers publiés | 158447734 |
| fichiers distincts | 34359954 |
| recherches | 1117418 |
| mots-clés annexes | 526535 |

TABLE 6.1 – Nombre global d'observations pour chacune des données

| catégorie | nb de mots-clés | mots |
|---|---|---|
| âges | 16 | 1yo, 2yo, 3yo, ... , 15yo, 16yo |
| pédo-explicites | 21 | pthc, pedo, hussyfan , pedofilia, ptsc, kingpass mafiasex, raygold, babyshivid, childlover, babyj preteen, qqaazz, youngvideomodels, childporn pedophile, yamad, pedoland, kidzilla, pedofilo, rape |
| pédo-liés | 7 | boy, girl, man, sex, mom, webcam, hot |
| normaux | 28 | saison, soundtrack, nokia, housewives, smallville, vista, black avi, new, day, rar, dvdrip, house, live, love, remix, rock windows, world, grosse, early, flowers, doing, christina christmas, madonna, pokemon, virtual |

TABLE 6.2 – mots-clés supervisés

renseignent des mots-clés complémentaires soit seulement 23%.

Afin de faciliter l'analyse des résultats, nous distinguons 4 catégories de mots-clés parmi les 72 mots supervisés à savoir : les mots-clés indiquant un âge, les mots-clés explicitement liés aux contenus pédophiles, les mots-clés pouvant être liés aux contenus pédophiles et les mots-clés normaux. Les différents mots-clés sont répartis comme indiqué par le tableau 6.2.

Nous souhaitons également pouvoir facilement distinguer parmi les données collectées celles à caractère pédophile des autres, ceci afin de pouvoir mettre en évidence les caractéristiques des échanges d'informations liés à ces contenus. Nous utilisons l'algorithme d'étiquetage des requêtes pédophiles proposé par Latapy et al [LMF10] et dont la précision a été validée, que nous avons appliqué à notre jeu de données. Nous avons besoin de différencier 3 nouvelles sous-catégories de mots-clés pour la catégorie pédo-lié à savoir : une liée à la sexualité (« sex », « rape », ...), une liée à l'enfance (« boy », « girl ») et une liée à la famille (« mom », « man »). L'algorithme d'étiquetage appliqué à notre base de données permet de quantifier les recherches (algorithme

1), les publications (algorithme 2) et les pairs pédophiles (algorithme3).

**Algorithme 1** : Algorithme d'étiquetage des requêtes de recherche

```
   Input : table searchs ; table searched-keywords ; table keywords ;
   Output : recherches étiquetées
 1 foreach requête search in table searchs do
 2 |   set search=normal ;
 3 |   mots-associés [ 0 ] = select keyword in keywords where searchs.targetid =
   |   keywords.keyhash ;
 4 |   mots-associés [ ] = select keyword in searched-keywords where searchs.id =
   |   searched-keywords.idsearch ;
 5 |   if mots-associés [ ] contient (pédo-explicite) OR (âge) OR ((sexualité OR famille)
   |   AND enfance) then
 6 |   |   set search=pedo ;
 7 |   end
 8 |   else
 9 |   |   if mots-associés [ ] contient sexualité OR famille OR enfance then
10 |   |   |   set search=pedo-lié
11 |   |   end
12 |   end
13 end
```

**Algorithme 2** : Algorithme d'étiquetage des publications

```
   Input : table publications ; table published-files ; table keywords ;
   Output : publications étiquetées
 1 foreach requête publish in table publications do
 2 |   set publish=normal ;
 3 |   filenames [ ] = select filename in published-files where publishs.id =
   |   published-files.idpublish) ;
 4 |   foreach name in filenames do
 5 |   |   mots-associés [ 0 ] = select keyword in keywords where publishs.targetid =
   |   |   keywords.keyhash ;
 6 |   |   mots-associés [ ] = parseKeywords(name) ;
 7 |   |   if mots-associés [ ] contient (pédo-explicite) OR (âge) OR ((sexualité OR famille)
   |   |   AND enfance) then
 8 |   |   |   set publish=pedo ; break ;
 9 |   |   end
10 |   |   else
11 |   |   |   if mots-associés [ ] contient sexualité OR famille OR enfance then
12 |   |   |   |   set publish=pedo-lié
13 |   |   |   end
14 |   |   end
15 |   end
16 end
```

**Algorithme 3** : Algorithme d'étiquetage des pairs

```
Input : table searchs ; table publications ; table peers ;
Output : pairs étiquetés
foreach pair in table peers do
    set pair=normal ;
    requetes-associées [ ] = select * in searchs where searchs.srcid=peers.id ;
    requetes-associées [ ] = select * in publishs where publishs.srcid=peers.id ;
    foreach requete in requetes-associées do
        if requete est pedo then
            set pair=pedo ; break ;
        end
        else
            if requetes est pedo-lié then
                set pair=pedo-lié ;
            end
        end
    end
end
```

La suite de ce chapitre analyse plus en détails les données collectées selon les catégories des mots-clés et la nature des requêtes (pédophile ou non). Les interrogations de la base de données ont été réalisées avec la collaboration d'Andreea Orosanu.

### 6.2.2 Analyse des requêtes de recherche

Pour chaque recherche effectuée par un utilisateur à partir de mots-clés, le client KAD émet des requêtes de recherche sur la DHT. Cependant un seul mot-clé parmi ceux entrés est utilisé pour interroger la DHT. Selon les versions du client, soit le mot-clé considéré est le premier de la liste, soit le plus long. Les autres mots-clés spécifiés servent alors à filtrer les résultats obtenus à partir du mot clé « principal ». De ce fait, en supervisant seulement le mot clé "linux", une recherche d'utilisateur "linux debian" peut être manquée si le plus long mot-clé est considéré par le client émetteur, de même que "debian linux" si le premier mot-clé est choisi. Néanmoins, nous avons vu que la grande majorité des recherches (77%) ne spécifie qu'un seul mot-clé. Nous analysons d'abord les mots-clés recherchés avant de considérer les requêtes de recherche complexes.

#### 6.2.2.1 Quantification des recherches par mot-clé

La quantification des recherches concernant les âges permet d'identifier les âges les plus demandés. Le graphique 6.8 montre que la demande croît jusqu'à l'âge de 12 ans avec 3 âges particulièrement demandés à savoir 6, 10 et 12 ans. La quantification des recherches pédo-explicites révèlent une grande différence de popularité entre les différents mots-clés nécessitant l'utilisation d'une échelle logarithmique pour présenter les résultats. Le graphique 6.9 montre ainsi que presque 3 ordres de grandeur séparent le mot-clé le plus populaire, à savoir « pthc » qui représente 43% des recherches de sa catégorie (67042 recherches) par rapport au mot le moins populaire « pédophile » représentant seulement 0.19% (295 recherches). Cette étude permet en outre de privilégier les mots-clés populaires à superviser lors des futurs déploiement afin d'améliorer l'effi-

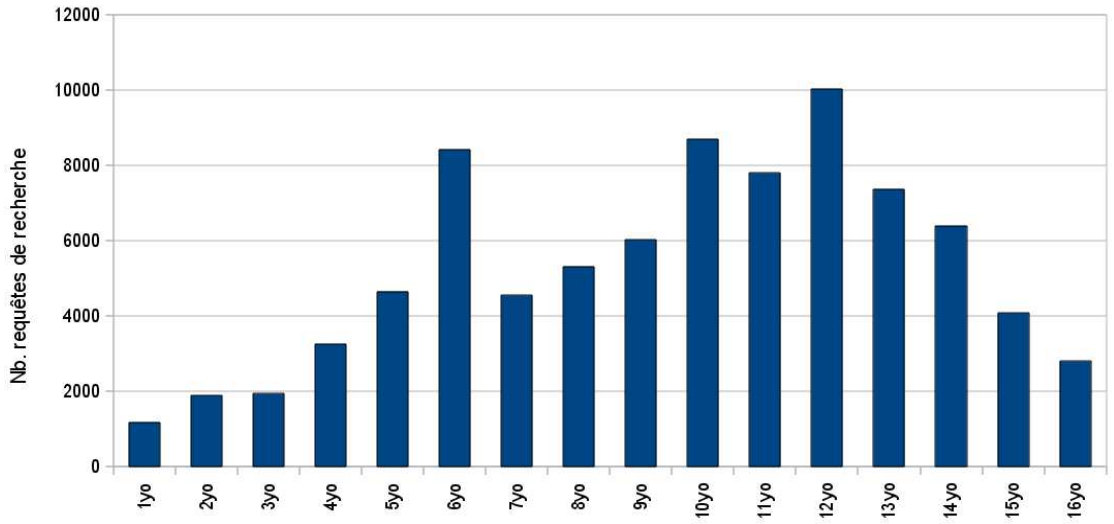


FIGURE 6.8 – Nombre de requêtes de recherche capturées selon les âges

cacité de la collecte de données. Cet écart de popularité est également visible pour les mots-clés normaux tels que présentés par le graphique 6.11. En dehors de l'âge, les mots-clés les plus recherchés pour chaque catégorie présentent le même ordre de grandeur : sex (34625), pthc (67042), avi (129456). L'étiquetage des requêtes de recherche capturées par l'algoritme 1 donne les proportions suivantes : 20.67% sont pédophiles, 12.49% peuvent être liées et 66.84% sont normales.

#### 6.2.2.2 Étude des mots-clés associés

L'étude des autres mots-clés présents dans les recherches complexes permet, d'une part, d'apprendre de nouveaux mots-clés associés à ceux supervisés et d'autre part, d'établir des corrélations entre les différents mots-clés selon les associations entrées par les utilisateurs. La matrice de corrélation étant trop conséquente pour être présentée avec ses valeurs, la figure 6.12 en présente une version simplifiée. Chaque carré de couleur indique au moins une co-apparition entre les mots-clés. La matrice de corrélation est bien symétrique mais un coloriage est réalisé suivant les lignes afin de mieux mettre en évidence les corrélation entre les différentes catégories de mots. Ainsi, on peut observer que les âges sont associés à d'autres âges (annotation 1), à des mots-clés pédophiles explicites (« pthc », « rape », ... (2)), à des mots-clés liés à la pédophilie (« sex », « boy », ... (3)), et à quelques mots clés normaux spécifiant le type de fichier (« rar » (4) et « avi » (5)) ou d'autres éléments de contexte (« new », « black » (6)). Le nombre de corrélations mesuré apporte une information supplémentaire. Par exemple, les âges proches sont souvent corrélés tels que « 11yo » et « 12yo ». La plus forte corrélation enregistrée parmi les mots-clés supervisés étant pour « windows » et « vista » avec 88%.

### 6.2.3 Analyse des requêtes de publication

L'analyse des requêtes de publication apporte d'autres connaissances sur l'activité des contenus pédophiles dans KAD. D'une part, d'un point de vue qualitatif car les requêtes de publication renseignent sur l'offre des contenus disponibles sur le réseau à travers les fichiers partagés par les pairs pour un mot-clé donné, ce qui est complémentaire par rapport aux recherches. Les

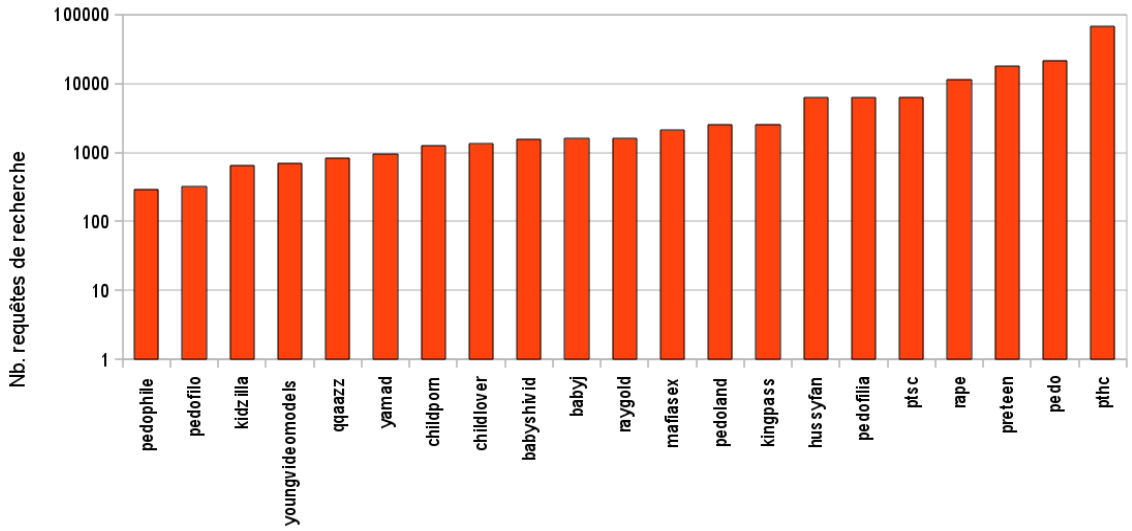


FIGURE 6.9 – Nombre de requêtes de recherche capturées pour les mots-clés pédo-explicites

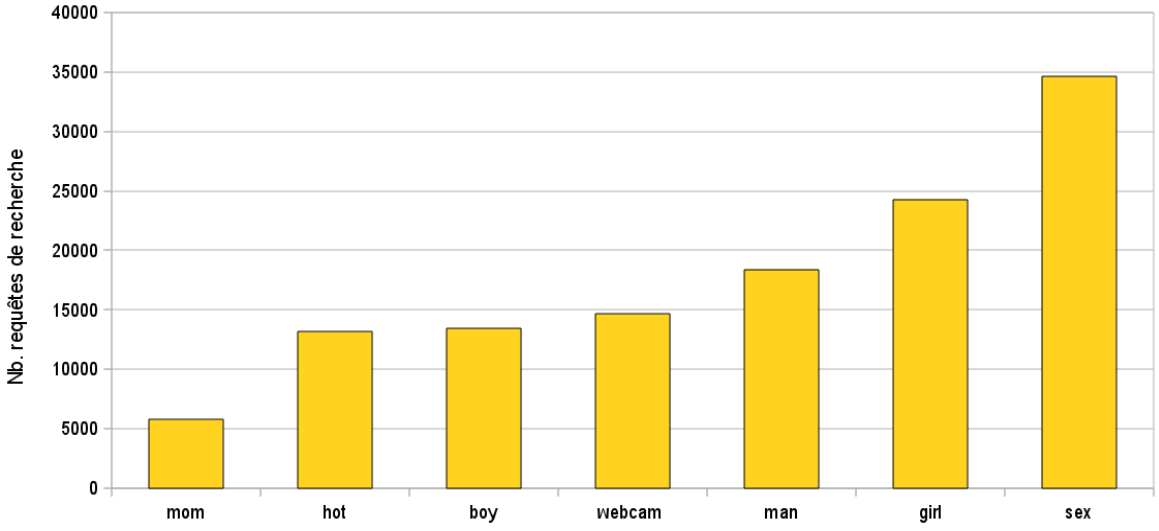


FIGURE 6.10 – Nombre de requêtes de recherche capturées pour les mots-clés pédo-liés

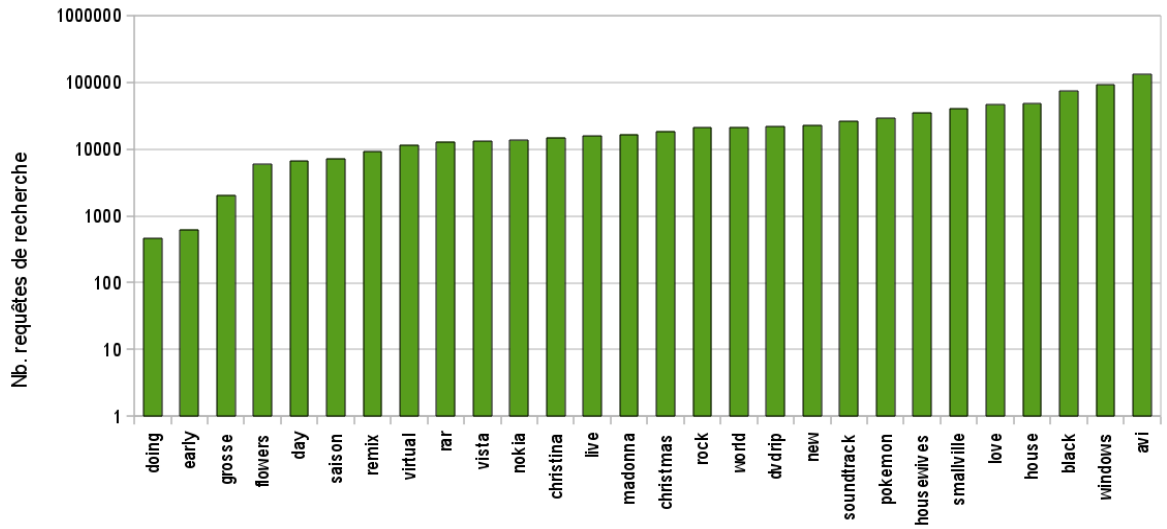


FIGURE 6.11 – Nombre de requêtes de recherche capturées pour les mots-clés normaux

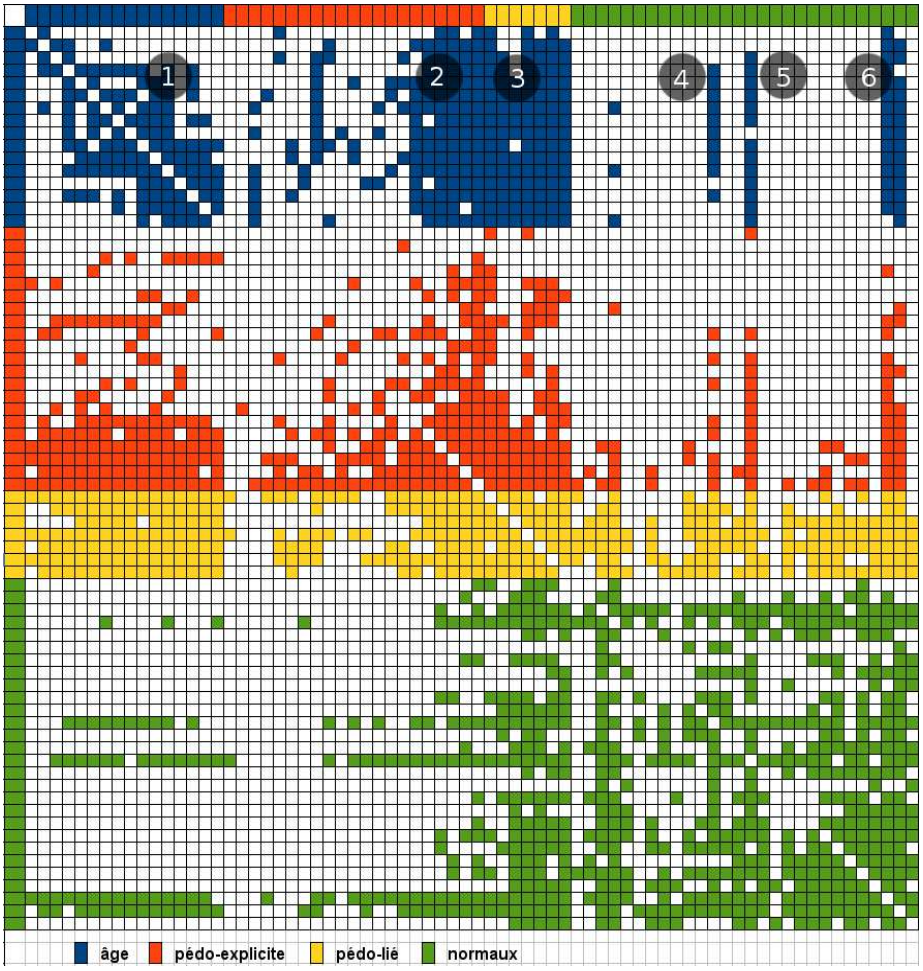


FIGURE 6.12 – Matrice de corrélation des mots-clés selon les recherches complexes

publications sont également moins sujettes aux erreurs de saisies et comportent des informations plus précises (de nombreux mots-clés dans le nom de fichier, le type de fichier, la taille...). D'autre part, d'un point de vue quantitatif car les requêtes de publication sont émises automatiquement par les clients pour tous les fichiers précédemment téléchargés, même en dehors de la période d'observation.

#### 6.2.3.1 Quantification des publications par mot-clé

Comme pour les recherches, les graphiques 6.13, 6.14 montrent respectivement le nombre de requêtes de publications pour les âges et les normaux. On peut tout d'abord remarquer que la quantité absolue de requêtes de publication capturées pour chaque mot-clé est plus importante que pour les recherches ce qui est une conséquence immédiate du processus de publication de KAD. On peut observer que la distribution des mots-clés normaux est beaucoup plus homogène. Par exemple, un facteur de 273 sépare le mot-clé normal le moins recherché « doing » du plus recherché « avi » (figure 6.11) alors que seul un facteur de 6 les sépare concernant les publications (figure 6.14). Ceci indique clairement que certains mots-clés sont peu utilisés dans les recherches malgré leur présence dans les noms de fichiers.

Il est cependant difficile de savoir précisément à quoi sont dues les différences constatées entre l'activité de recherche et de publication de certains mots-clés. Plusieurs raisons peuvent expliquer cette différence :

- une disproportion entre l'offre et la demande
- le fait qu'un mot-clé soit peu discriminant dans le cadre d'une recherche (par exemple « new »)
- le fait qu'un mot-clé ne soit pas placé en première position lors d'une recherche complexe (et donc ne soit pas utilisé pour interroger la DHT)

Une autre différence évidente est visible dans le graphique des âges publiés 6.13. La distribution des âges est différente de celle des âges recherchés, avec notamment l'absence du premier pic autour de 6 ans. Ceci peut indiquer une différence entre l'offre et la demande pour certains âges. Afin de mieux appréhender la différence de popularité entre les recherches et les publications, le ratio recherche / publication pour chaque catégorie de mots-clés est disponible en annexe B. De même sont disponibles les graphiques des publications de mots-clés pédo-explicites et (graphique B.1) et liés (graphique B.2). L'étiquetage des requêtes de publication indique que 22.28% sont pédophiles, 27.58% peuvent être liées et 50.15% sont normales.

#### 6.2.3.2 Étude des fichiers publiés

Pour chaque requête de publication reçue pour un mot-clé , nous disposons de la liste des fichiers partagés par le pair pour ce mot-clé, ainsi que leurs caractéristiques. Il est donc possible d'établir le nombre total de fichiers partagés dans KAD pour chacun des mots-clés supervisés. Cependant, le principe des réseaux P2P étant de partager, et donc de répliquer les contenus, le nombre de fichiers partagés par mot-clé ne rend pas compte du nombre des contenus originaux associés à chaque mot-clé. Pour cela, il faut considérer le nombre de fichiers distincts (dont l'identifiant MD4 est différent) associés à chaque mot-clé. Les graphiques 6.15 et 6.16 présentent respectivement la distribution des fichiers, et des fichiers distincts, publiés par catégorie de mots-clés. On peut remarquer que les fichiers explicitement pédophiles et indiquant les âges représentent une fraction beaucoup plus importante des fichiers distincts que des fichiers partagés. Cela indique que ces contenus sont en moyenne beaucoup moins répliqués que les autres. Les nombres de fichiers et fichiers distincts détaillés pour chaque catégorie de mots-clés sont disponibles dans

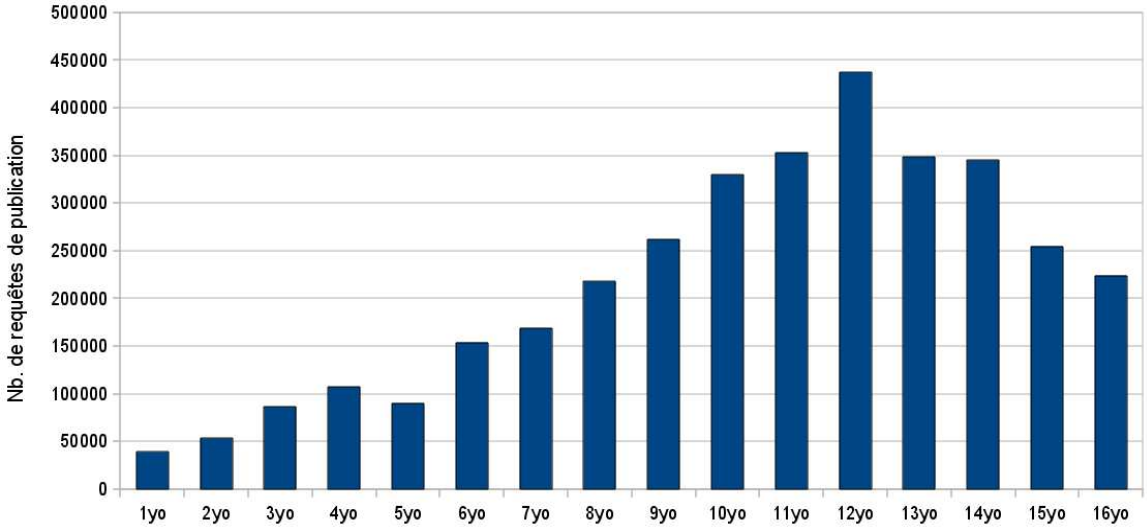


FIGURE 6.13 – Nombre de requêtes de publication capturées selon les âges

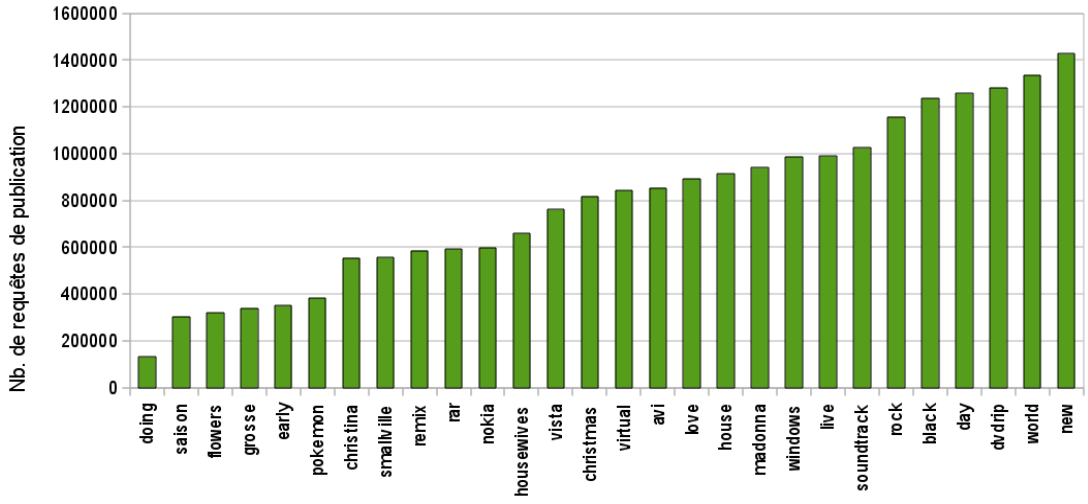


FIGURE 6.14 – Nombre de requêtes de publication capturées pour les mots-clés normaux

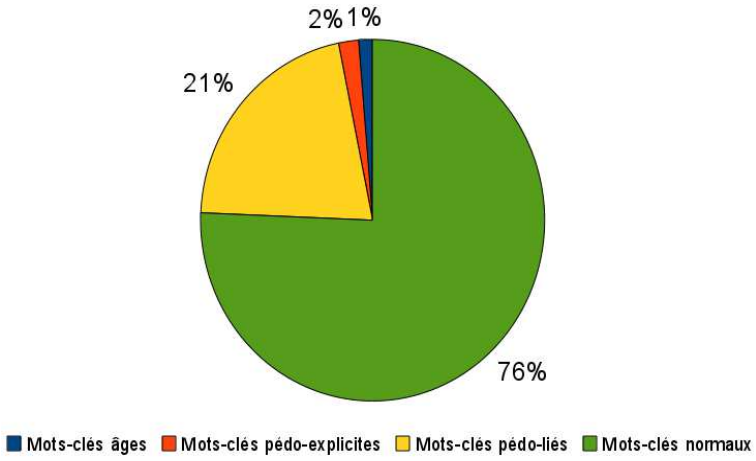


FIGURE 6.15 – Distribution des fichiers publiés par catégorie de mots-clés

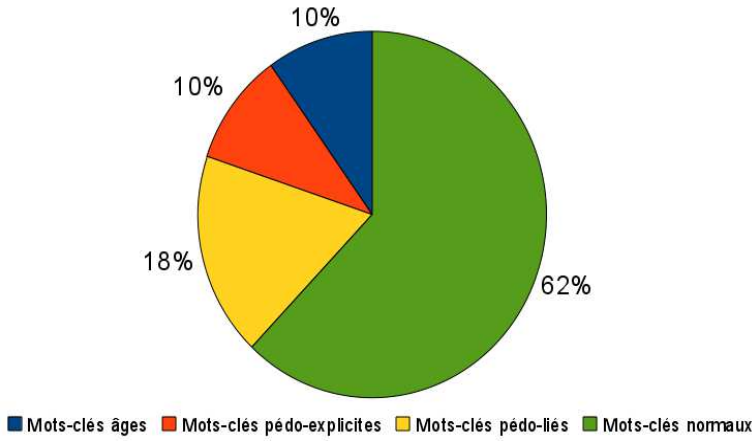


FIGURE 6.16 – Distribution des fichiers distincts publiés par catégorie de mots-clés

l'annexe B.3.

A l'instar des recherches complexes affichant plusieurs mots-clés, les noms de fichiers peuvent être étudiés afin de découvrir les co-apparitions des mots-clés dans une même description de contenu. La matrice de corrélation obtenue à partir des fichiers publiés est disponible en annexe B.3.

### 6.2.4 Analyse des pairs

Nous terminons notre description des données par l'activité des pairs avec l'objectif de mettre en évidence de possibles caractéristiques pour chaque groupe. Les pairs sont préalablement étiquetés d'après leurs requêtes émises grâce à l'algorithme 3. L'étiquetage des pairs d'après les données récoltées est présenté dans le tableau 6.3. On peut remarquer que l'étiquetage des pairs selon les publications est presque équivalent au résultat global. Cela s'explique par la faible proportion des pairs ayant effectué des recherches (3.25%) alors que presque tous ont effectué des publications (98.39%). L'activité des groupes de pairs peut ensuite être considérée selon plusieurs

| catégorie | selon recherches | selon publications | globalement |
|---|---|---|---|
| pédo | 24.81% | 12.32% | 12.38% |
| pédo-lié | 24.10% | 36.77% | 36.38% |
| normaux | 51.09% | 50.91% | 51.24% |

TABLE 6.3 – Résultats de l'étiquetage des pairs d'après l'algorithme 3

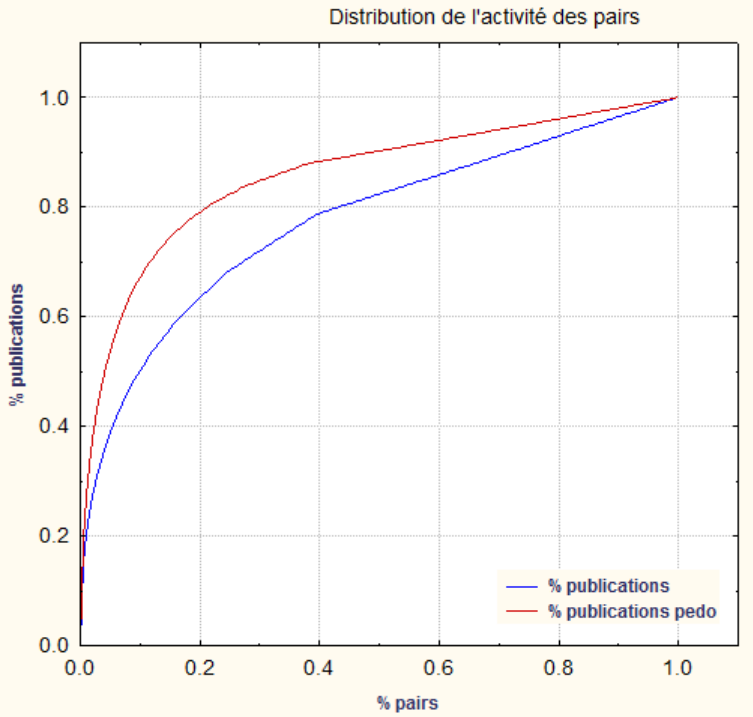


FIGURE 6.17 – Diagramme cumulatif des publications par rapport aux pairs

axes tels que leur activité temporelle ou géographique.

#### 6.2.4.1 Activité des pairs

L'activité des pairs peut être représentée par un diagramme cumulatif exprimant le pourcentage des requêtes capturées en fonction du pourcentage des pairs correspondant ; ceux-ci étant ordonnés selon leur contribution. Le diagramme 6.17 présente ainsi l'activité de publication des pairs normaux et ceux étiquetés comme pédophiles et montre que l'activité de ces derniers est moins uniforme.

En utilisant les requêtes étiquetées, nous avons également considéré leur répartition selon les jours de la semaine et les heures de la journée traduisant l'activité temporelle des pairs. Ainsi, les graphiques 6.18 et 6.19 montrent la distribution de l'activité des pairs concernant les recherches respectivement selon le jour et l'intervalle horaire. Les graphiques B.18 et B.17, disponibles en annexe, concernant l'activité de publication. On remarque notamment que les pairs effectuant des requêtes pédophiles ont une activité plus tardive et plus accentuée le dimanche.

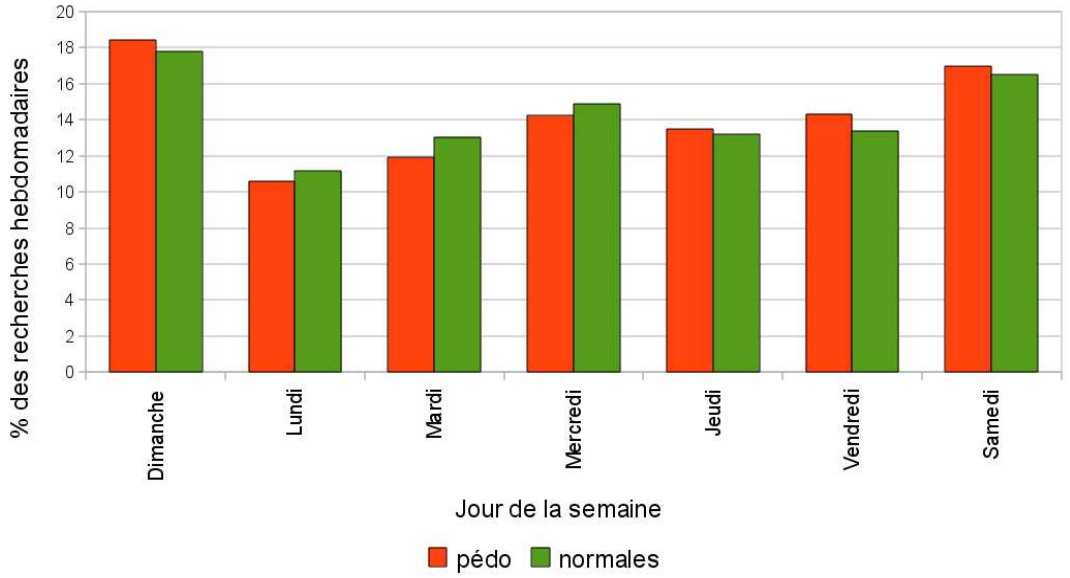


FIGURE 6.18 – Distribution journalière des requêtes de recherche capturées

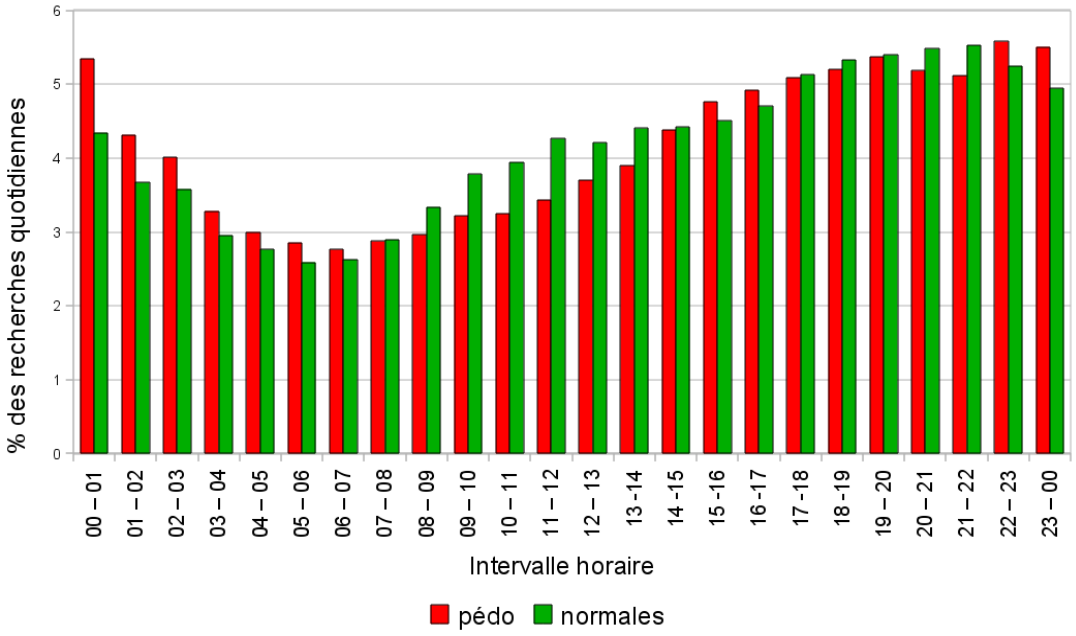


FIGURE 6.19 – Distribution des requêtes de recherche reçues par intervalle horaire

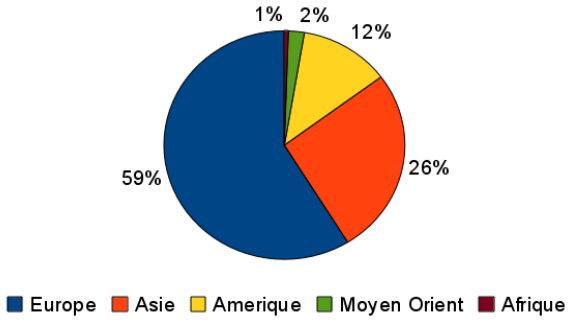


FIGURE 6.20 – Répartition géographique des pairs dans le monde

#### 6.2.4.2 Géolocalisation

La géolocalisation des pairs est basée sur leur adresse IP. Le graphique 6.20 montre la répartitions des pairs dans les grandes régions du monde. Celle-ci est conforme aux précédentes études réalisées sur KAD à l'exception d'une sous-représentation de l'Asie qui peut être expliquée par le fait que les mots-clés observés sont écrits en caractères latins.

Le tableau 6.4 donne le nombre et la répartition des pairs relevés pour les 25 principaux pays en terme d'activité, ainsi que la quantification et la répartition de ceux ayant effectués des requêtes pédophiles. Nous présentons également le ratio des pairs étiquetés pédophiles par pays obtenu d'après nos données et le même ratio obtenu par Matthieu Latapy et al d'après l'analyse de recherches capturées sur le réseau eDonkey [LMFS09]. Nous pouvons remarquer que la proportion de pairs étiquetés comme pédophiles dans chaque pays est tout à fait similaire entre les deux jeux de données.

## Conclusion

Nous avons présenté un déploiement à grande échelle de l'architecture de supervision élaborée dans le chapitre 5. Même si l'ensemble des fonctionnalités de l'architecture n'a pas été utilisé, en particulier l'annonce de faux fichiers et les pots de miel associés, la quantité et la qualité des données collectées valident notre approche basée sur l'insertion ciblée de sondes sur la DHT. Les sondes déployées sur PlanetLab ont montré une consommation de ressources constante et modérée parfaitement compatible avec cette plateforme expérimentale. Après quelques optimisations, en particulier la création d'une mémoire à court terme pour traiter rapidement les réplicats, la base de donnée a pu être remplie en temps réel pour les 72 mots-clés supervisés ce qui a permis d'éviter un stockage des informations au niveau des sondes et un délai de traitement supplémentaire. La base de données reste cependant le point critique de l'architecture, notamment en cas de perte de connectivité de celle-ci.

Nous avons ensuite réalisé une description statistique des données collectées avec pour objectif la caractérisation des activités à caractère pédophile. L'étiquetage des requêtes dans un premier temps, puis les analyses des recherches, des publications et enfin des pairs ont permis de mettre en évidence des premiers résultats tels que l'attrait des âges, les corrélations entre mots-clés, l'activité des horaires des pairs ou encore leur quantification par pays à l'échelle mondiale. Une analyse plus approfondie de ces données est nécessaire pour une meilleure caractérisation des activités pédophiles mais sort du cadre de cette thèse.

| Pays | Nb pairs | % des pairs | Nb pairs pédo | % des pairs pédo | ratio des pairs pédo | ratio des rech. pédo (LIP6) |
|---|---|---|---|---|---|---|
| Italy | 3267132 | 25.44 | 238154 | 14.98 | 0.07 | 0.08 |
| China | 2409843 | 18.77 | 130037 | 8.18 | 0.05 | 0.05 |
| France | 1328839 | 10.35 | 111279 | 7.00 | 0.08 | 0.11 |
| Spain | 1273897 | 9.92 | 95844 | 6.03 | 0.08 | 0.06 |
| Brazil | 944479 | 7.35 | 151605 | 9.54 | 0.16 | 0.17 |
| Germany | 490101 | 3.82 | 190765 | 12.00 | 0.39 | 0.54 |
| Taiwan | 331592 | 2.58 | 32313 | 2.03 | 0.10 | 0.03 |
| Israel | 256818 | 2.00 | 23975 | 1.51 | 0.09 | 0.12 |
| Poland | 233040 | 1.81 | 26055 | 1.64 | 0.11 | 0.12 |
| Republic of Korea | 211157 | 1.64 | 21647 | 1.36 | 0.10 | 0.02 |
| United States | 196663 | 1.53 | 77291 | 4.86 | 0.39 | 0.59 |
| Argentina | 165739 | 1.29 | 23048 | 1.45 | 0.14 | 0.18 |
| Portugal | 162160 | 1.26 | 25594 | 1.61 | 0.16 | 0.08 |
| Belgium | 128429 | 1.00 | 19841 | 1.25 | 0.15 | 0.16 |
| Russian Federation | 119217 | 0.93 | 75917 | 4.78 | 0.64 | 1.35 |
| Switzerland | 118058 | 0.92 | 27082 | 1.70 | 0.23 | 0.53 |
| United Kingdom | 91819 | 0.71 | 27324 | 1.72 | 0.30 | 0.53 |
| Netherlands | 81298 | 0.63 | 32154 | 2.02 | 0.40 | 0.46 |
| Canada | 76346 | 0.59 | 17862 | 1.12 | 0.23 | 0.51 |
| HongKong | 63743 | 0.50 | 3463 | 0.22 | 0.05 | ND |
| Mexico | 53439 | 0.42 | 23252 | 1.46 | 0.44 | 0.52 |
| Austria | 47413 | 0.37 | 14328 | 0.90 | 0.30 | 0.53 |
| Japan | 46033 | 0.36 | 7849 | 0.49 | 0.17 | 0.14 |

TABLE 6.4 – Quantification géographique des pairs étiquetés comme pédophiles

Les données collectées ouvrent ainsi plusieurs perspectives de recherche. Une approche simple par des statistiques descriptives montre rapidement ses limites car les données sont trop importantes pour être traitées manuellement. L'application de méthodes de fouille de données est nécessaire afin d'extraire de nouvelles connaissances utiles à partir des données. Une première approche non supervisée pourrait ainsi mettre en évidence des règles d'association permettant d'apprendre de nouveaux mots-clés utilisés pour améliorer l'étiquetage, puis, une fois les classes bien établies, une approche supervisée permettrait de dégager des caractéristiques parmi toutes les combinaisons de paramètres possibles différenciant les activités pédophiles par rapport aux normales pour les différents objets (recherches, publications, pairs).

Nous avons montré dans cette première partie que les contenus diffusés au sein des réseaux P2P pouvaient être supervisés rigoureusement et efficacement, ce qui est la première étape nécessaire à la lutte contre les contenus illégaux tels que ceux liés à la pédophilie. La diffusion tels contenus, au delà des pairs les recherchant, est réalisée par des pairs malveillants corrompant le mécanisme d'indexation de la DHT par diverses attaques dont l'étude fait l'objet de la troisième partie de cette thèse.

Troisième partie

# Supervision des attaques et protection de la DHT

# Chapitre 7

# Métrologie des attaques contre la DHT

**Sommaire**



## Introduction

Nous avons décrit et exploité en pratique, dans la seconde partie de ce manuscrit, des vulnérabilités permettant l'insertion ciblée de nœuds au sein de KAD malgré les plus récentes protections. Or, les nœuds ainsi placés peuvent réaliser plusieurs actions malveillantes (pollution, surveillance, déni de service).

Si ces différentes attaques sont d'ores et déjà connues et ont été expérimentées ponctuellement, aucune étude à ce jour ne s'est intéressée à les recenser en pratique. De plus, les précédentes études [LKXR05] [LNR06] ayant mis en évidence la pollution des réseaux P2P datent de 2005 et ont été réalisées sur des réseaux cibles aujourd'hui obsolètes tels qu'Overnet ou Kazaa. Dès lors, on peut légitimement se demander si la quantification de la pollution et les différentes formes relevées alors correspondent aux réseaux actuels tels que KAD.

Notre objectif est ici d'étudier différentes attaques affectant la DHT de KAD et plus particulièrement deux menaces. D'une part, la pollution affectant son mécanisme d'indexation et pouvant amener les utilisateurs à télécharger inconsciemment des contenus indésirables portant atteinte à leur sécurité. D'autre part, les attaques ciblées sur des contenus et réalisées par des nœuds placés intentionnellement dans le réseau. Il est important de recenser ces attaques afin de savoir si et dans quelle mesure, les dizaines de millions d'utilisateurs utilisant les tables de hachage distribuées publiques sont affectés par ces menaces.

## 7.1 Mesure de la pollution du mécanisme d'indexation

### Introduction : les différents types de pollution

La pollution des réseaux P2P étant un sujet vaste et complexe, nous proposons tout d'abord de recenser les différentes formes de pollution pouvant s'appliquer au réseau KAD ainsi que leurs moyens possibles de mise en oeuvre afin de bien positionner notre contribution dans ce domaine. Nous distinguons deux types de pollution affectant les réseaux P2P selon qu'ils impliquent directement la corruption du contenu ou celle du système d'indexation de la DHT.

#### Pollution des contenus

La première forme de pollution par des contenus consiste à partager des fichiers dont le contenu correspond à la description mais dont celui-ci est fortement dégradé. Cette forme de pollution a notamment été largement constatée sur le réseau Kazaa [LKXR05]. La dégradation peut avoir plusieurs formes telles que l'ajout de bruit, la fin prématurée du contenu, l'annonce d'un message, etc. Une autre forme de pollution des contenus vise à dégrader les performances de téléchargement d'un fichier existant non pollué en partageant de fausses pièces d'information. Cette forme de pollution est notamment utilisée dans le cadre de BitTorrent dont l'indexation des fichiers partagés est contrôlée par des site web qui limitent l'indexation de fichiers indésirables.

Du point de vue du pollueur, ces deux formes de pollution ont le désavantage de requérir, au moins dans un premier temps, des ressources pour diffuser rapidement les fichiers corrompus. Dans le cadre des réseaux P2P basés sur des DHTs, l'indexation des contenus partagés est réalisée par les pairs eux-mêmes et est peu contrôlée. Cette faiblesse permet une autre forme plus efficace de pollution appelée pollution de l'indexation que nous allons étudier plus en détail dans KAD.

#### Pollution de l'indexation

Ainsi la pollution dite par empoisonnement d'index consiste à remplir le système d'indexation de fausses références de fichiers, qui ne sont en réalité partagés par aucune source. Cette forme de pollution a été constatée dans le réseau Overnet [LNR06] qui opérait une DHT similaire à KAD. Afin de sembler populaire, chaque faux fichier est annoncé de nombreuses fois. L'article a montré que cette forme de pollution peut également affecter KAD [LMSW10]. Si le nombre de faux fichiers annoncés pour un mot-clé donné est suffisamment élevé, ceux-ci peuvent de plus saturer la table d'indexation des pairs chargés dudit mot-clé et empêcher ainsi le référencement de fichiers légitimes.

Cependant, une autre forme de corruption de l'indexation existe que nous appelons falsification d'indexation. Elle consiste à indexer un même fichier sous différents noms, et par conséquent différents mots-clés, qui ne sont pas liés au contenu réel du fichier. Pour un titre donné, le fichier pollué est publié de nombreuses fois par de fausses sources afin de le rendre populaire. Cette forme de pollution est la plus néfaste constatée pour deux raisons. D'abord, car elle aboutit au téléchargement complet d'un contenu indésirable et gaspille par conséquent inutilement des ressources, mais surtout, car le contenu téléchargé peut être dangereux pour l'utilisateur. Le contenu réel peut ainsi être un virus ou une vidéo pouvant gravement heurter la sensibilité de l'utilisateur (contenu pornographique, pédophile, etc.). Par ailleurs, cette forme de pollution génère des faux positifs lors de la supervision des fichiers illégaux dont l'indexation est ainsi falsifiée. Or, si les utilisateurs peuvent expérimenter régulièrement cette forme de pollution sur un réseau P2P tel que KAD, aucune étude à ce jour n'a décrit ni quantifié cette forme de pollution.

Nous nous intéressons donc ici à la détection et à la quantification de cette pollution falsifiant l'indexation des fichiers car elle n'a pas encore été étudiée et parce qu'il s'agit de la forme la plus néfaste de pollution affectant les contenus.

### 7.1.1 Détection de la pollution

#### 7.1.1.1 Visibilité de la falsification d'indexation

Le principe de cette pollution repose sur le fait d'associer de nombreux noms différents à un même fichier. Cependant, le schéma d'indexation à deux niveaux de KAD rend très difficile la détection de cette pollution. En effet, si l'on se place du point de vue des pairs responsables d'un mot-clé, ces derniers ne reçoivent que les publications, pour un fichier pollué donné, incluant le mot-clé dont ils ont la charge dans leur nom. Cette contrainte forte sur la présence d'un mot-clé empêche de détecter le mélange d'indexation au niveau des mots-clés. Dans le cas où deux noms complètement différents sont associés à un fichier, ce dernier sera indexé au travers des mots-clés sur des pairs complètement différents. L'interrogation des pairs responsables des mots-clés ne permet donc pas détecter le mélange d'indexation ce qu'illustre le schéma. Nous avons confirmé cette hypothèse en plaçant une sonde à proximité du mot-clé « avatar ». L'ensemble des noms de fichiers recueillis par la sonde contiennent effectivement le mot-clé « avatar » et ne laissent présager aucun contenu pollué alors que ce mot-clé est en réalité fortement affecté, comme nous le montrerons en section 7.1.2.

Si l'on se place du point de vue des pairs responsables d'indexer les sources d'un fichier, ceux-ci ne sont pas non plus capables d'appréhender le mélange d'indexation. En effet, bien que l'ensemble des sources publie le fichier pollué sur les mêmes pairs proches de son identifiant, le nom de fichier ne fait pas partie des informations publiées lors de l'envoi d'une requête de type `KADEMLIA2_PUBLISH_SOURCE_REQ` associant une source (adresse IP, port) à un fichier (identifiant MD4). Le nom du fichier partagé est une information uniquement associée aux requêtes de type `KADEMLIA2_PUBLISH_KEY_REQ` afin de guider le choix de l'utilisateur lors de la présentation des différents fichiers disponibles pour un mot-clé. L'interrogation de la DHT de KAD ne permet donc pas de détecter le mélange d'indexation. Les pairs indexant les mots-clés ne vooient pas les noms de fichiers contradictoires et les pairs responsables des sources n'ont pas connaissance des noms de fichiers associés.

Il est cependant possible de détecter le mélange d'indexation grâce à une fonctionnalité des clients KAD externe à la DHT et disponible lors du téléchargement d'un fichier. Lorsqu'un fichier doit être téléchargé, les sources potentielles sont découvertes par interrogation de la DHT (`KADEMLIA2_SEARCH_SOURCE_REQ`). L'utilisation de la DHT de KAD s'arrête alors et une connexion TCP vers chacune des sources potentielles est alors initiée afin de commencer le téléchargement. Les pairs acceptant la connexion constituent les sources réelles à un moment donné et, selon leur charge, partagent une partie du fichier ou placent la demande dans une file d'attente. Une requête spécifique peut alors être envoyée aux sources réelles par l'intermédiaire de la connexion TCP afin de connaître le nom par lequel elles partagent le fichier demandé. Cette information est accessible dans la fenêtre « Détails du fichier » de l'interface graphique. Grâce à ces informations, des noms contradictoires annoncés par les différentes sources peuvent apparaître et le mélange d'indexation peut ainsi être constaté. La capture d'écran 7.1 montre ainsi les noms annoncés pour un fichier sain alors que la capture 7.2 montre un fichier dont l'indexation est corrompue. Concernant le fichier sain, nous pouvons constater que la majorité des sources (22) annoncent exactement le nom de fichier requis par l'utilisateur et les autres sources annoncent une variante minime du même nom en gardant de nombreux mots-clés en commun. En revanche, concernant

le fichier pollué, aucun des pairs n'annonce le nom de fichier désiré par l'utilisateur, à savoir « Indiana Jones et les Aventuriers de l'Arche Perdue » , ni même ne s'accorde sur un autre nom, tous les noms de fichier annoncés étant différents les uns des autres et ne partageant aucun mot-clé commun.


FIGURE 7.1 – Noms de fichier annoncés par les sources d'un fichier sain

#### 7.1.1.2 Métrique de détection

Étant donné un fichier dont le contenu est identifié par son empreinte MD4, nous souhaitons savoir si celui-ci est fiable ou fait l'objet d'une pollution par mélange d'indexation en analysant les différents noms de fichier donnés par les sources contactées. Nous utilisons pour cela une métrique capable d'apprécier la similarité entre deux ensembles de mots comme la distance de Jaccard [MS99]. Soit X et Y deux ensembles de mots-clés, X étant l'ensemble des mots composant le nom de fichier choisi par l'utilisateur et Y étant l'ensemble des mots composant un des noms de fichier donné par les sources, l'indice de similarité de Tversky [Tve77] $S(X, Y)$ est une valeur entre 0 et 1 définie par la formule suivante :


FIGURE 7.2 – Noms de fichier annoncés par les sources d'un fichier pollué

$$S(X,Y) = \frac{|X \cap Y|}{|X \cap Y| + \alpha * |X - Y| + \beta * |Y - X|} \tag{7.1}$$

Pour détecter la pollution, nous utilisons un cas particulier de cette formule en fixant $\alpha = \beta = 0.5$ car aucun des deux noms ne peut être privilégié et considéré comme une référence, ce qui produit le coefficient de similarité de Dice [MS99] définit par :

$$S(X,Y) = \frac{|X \cap Y|}{|X \cap Y| + 0,5 * |X - Y| + 0,5 * |Y - X|} = \frac{2 * |X \cap Y|}{|X| + |Y|} \tag{7.2}$$

Si les deux noms de fichiers sont identiques, le coefficient résultant est de 1, si les deux noms sont complètement disjoints, le coefficient résultant vaut 0. Pour attribuer un indice de pollution $P$ à un fichier $X$, nous calculons la moyenne de l'ensemble des indices de similarité obtenus par les différents noms de fichiers trouvés, soit :

$$P(X) = \frac{\sum_{i=1}^{n} S(X, Y_i)}{n} \tag{7.3}$$

### 7.1.2 Quantification et caractérisation de la pollution

#### 7.1.2.1 Exploration des fichiers

Afin de quantifier la pollution affectant le réseau P2P KAD, nous avons collecté les informations présentées ci-avant pour de nombreux fichiers. L'accès aux noms de fichiers contradictoires est très couteux puisqu'il nécessite : (1) une recherche de fichiers sur la DHT, (2) une recherche de sources potentielles sur la DHT, (3) l'établissement d'une connexion TCP pour chaque source réelle. Nous limitons pour cela notre collecte d'information à l'étude des 100 contenus les plus téléchargés de 2010 selon l'un des principaux site d'indexation de torrents [19] recevant plus de 100 millions de recherches par an [20]. La liste des 100 mots-clés recherchés est disponible en annexe C. Pour chacun des contenus, nous collectons les différents noms associés aux 20 fichiers les plus populaires trouvés dans KAD, c'est à dire dont le nombre de sources estimé à l'issue de la recherche par mots-clés est le plus important. Nous avons en effet montré en section 5.3.4 que le nombre de source estimé est un paramètre privilégié par les utilisateurs lors du choix d'un fichier à télécharger. Nous estimons donc la pollution du réseau à partir d'une base de 2000 fichiers parmi les plus populaires. La collecte de ces informations a été réalisée avec la collaboration de Guillaume Montassier.

Une fois le téléchargement d'un fichier lancé, la découverte des sources réelles est progressive et prend un certain temps que nous souhaitons estimer avant de lancer la collecte des différents noms pour les 2000 fichiers. Sur un échantillon de 150 fichiers, nous avons ainsi mesuré l'évolution du nombre de sources réelles obtenues pendant une heure dont le graphique 7.3 illustre les dix premières minutes. Il apparaît qu'en moyenne plus de 97% des sources trouvées à l'issue d'une heure sont obtenues dès 300 secondes. Pour collecter les données nécessaires à la quantification de la pollution, nous instrumentons un client KAD recherchant séquentiellement les mots-clés définis en annexe C, lançant pour chacun d'eux le téléchargement des 20 fichiers les plus populaires et enregistrant les différents noms de fichiers obtenus des sources réelles après 300 secondes.

19. http://www.kickasstorrents.com/
20. http://torrentfreak.com/bittorrent-zeitgeist-what-people-searched-for-in-2010-101227/

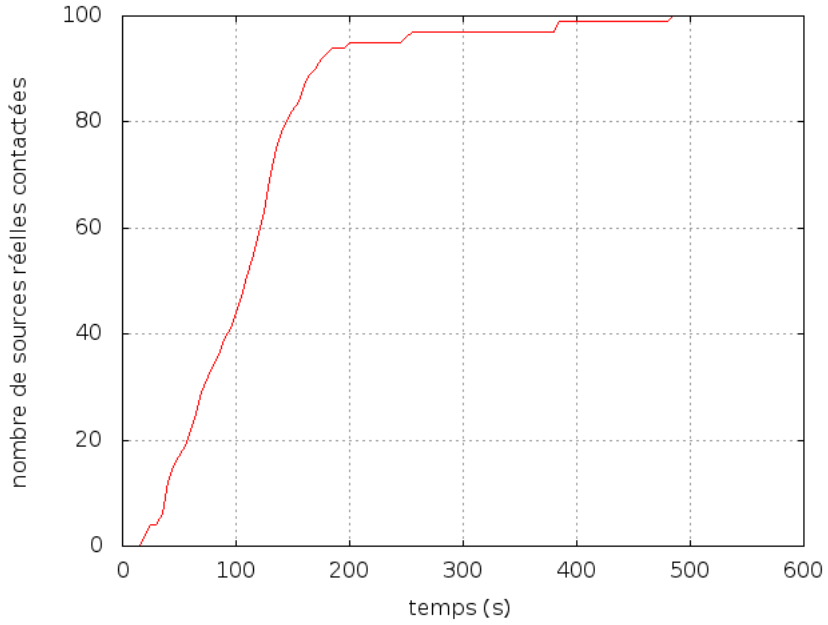


FIGURE 7.3 – Nombre moyen de sources réelles trouvées en fonction du temps

#### 7.1.2.2 Résultats

Après avoir collecté les différents noms possibles des 2000 fichiers, nous avons appliqué notre métrique à chacun d'eux. Les graphiques 7.4 et 7.5 montrent respectivement la distribution et la distribution cumulative du nombre de fichiers en fonction des indices de pollution obtenus. Nous pouvons remarquer que la métrique est bien discriminante en établissant la majorité des scores aux extrémités de l'échelle ce qui facilite l'établissement de seuils de détection. Nous définissons ainsi trois seuils. Les fichiers dont le score de similarité est supérieur à 0.7 sont considérés comme sains, entre 0.7 et 0.3 comme peut-être pollués et inférieur à 0.3 comme pollués.

L'application de ces trois seuils à l'ensemble des 2000 fichiers populaires étudiés donne la répartition présentée dans le diagramme 7.6. Plus de 41% des contenus populaires sont ainsi clairement pollués par le mélange d'indexation. De plus, pour 21% des fichiers, aucune source réelle n'a pu être trouvée malgré le fait que les fichiers apparaissent avec un nombre élevé de sources estimées lors de la recherche par mot-clé, ceci correspond donc à une pollution par empoisonnement d'index. En considérant les deux formes de pollution, plus de 62% des fichiers sont pollués, bien que le mélange d'indexation soit la plus néfaste des deux formes de pollution constatées. Seuls 29% des fichiers peuvent être considérés comme parfaitement sains ; le doute subsistant par rapport à la métrique pour moins de 10% des fichiers.

Afin d'évaluer la précision de notre métrique, nous avons procédé à l'analyse des noms de fichiers obtenus par deux experts. Chacun d'eux a évalué 100 fichiers dont des sources réelles ont été trouvées, soit 10% de l'échantillon global et ont classé, tout comme la métrique, chaque fichier parmi les trois catégories : pollué, peut-être pollué ou sain. Les experts ont cependant très peu recours à la catégorie « peut-être pollué » qui représente moins de 1% des fichiers analysés. Parmi les fichiers classés dans cette catégorie par la métrique, 78% ont été jugés sains par les expert et 12% pollués. L'avis des experts nous permet également de calculer le taux d'erreur de notre métrique. Ainsi, les faux positifs sont les fichiers considérés comme pollués par la métrique et sains par les experts. A l'inverse, les faux négatifs sont les fichiers considérés comme sains par

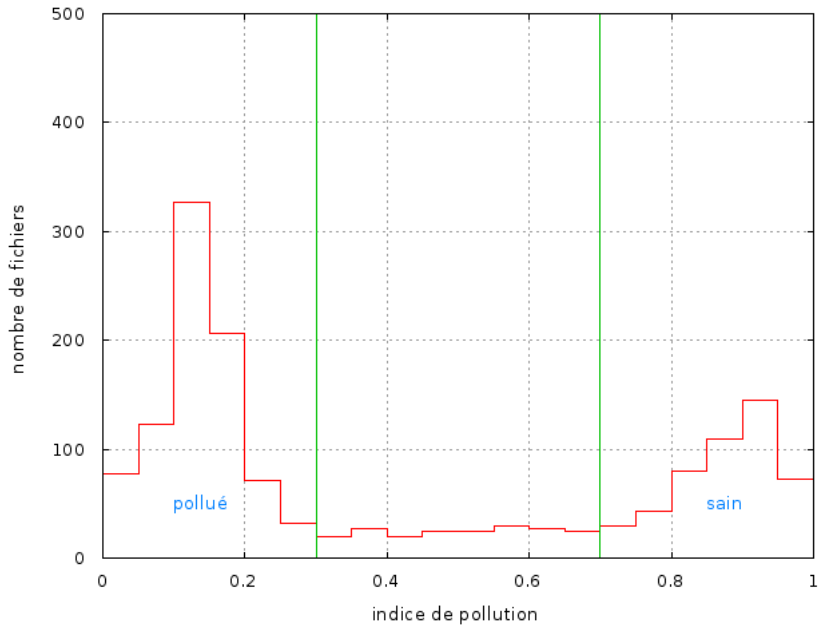


FIGURE 7.4 – Distribution du nombre de fichiers en fonction de l'indice de pollution obtenu

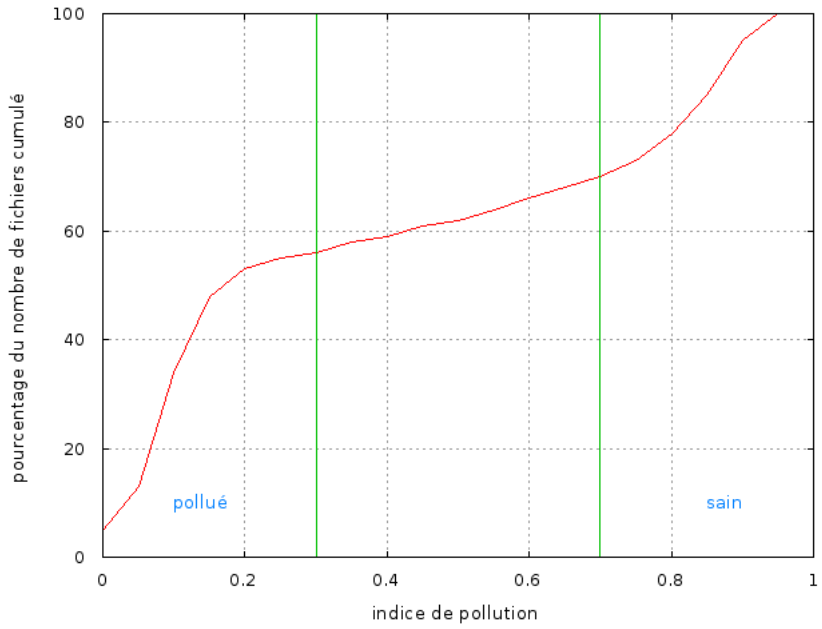


FIGURE 7.5 – Distribution cumulative du nombre de fichiers en fonction de l'indice de pollution obtenu

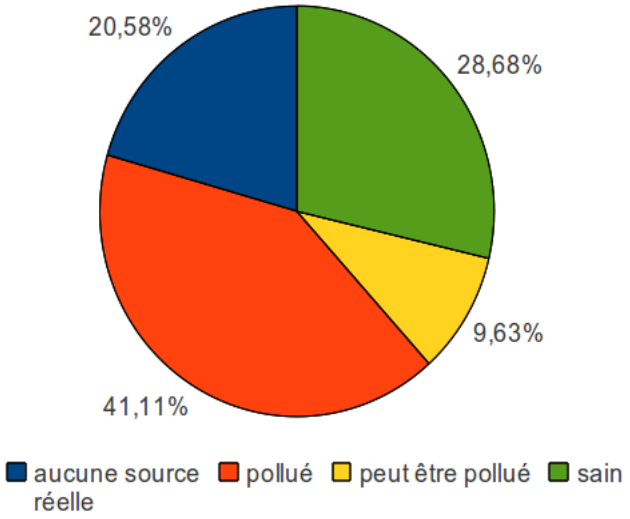


FIGURE 7.6 – Quantification de la pollution des contenus dans le réseau P2P KAD

| % de faux positifs | % de faux négatifs |
|---|---|
| 3.78 | 0.88 |

TABLE 7.1 – Taux d'erreur de la métrique de détection du mélange d'indexation

la métrique et pollués par les experts. Le tableau 7.1 donne les valeurs des taux d'erreur. Au regard du taux d'erreur très faible constaté, la métrique que nous proposons détecte très bien la pollution du mélange d'indexation. La proportion de fichiers sains est en revanche sous évaluée car une partie d'entre eux est considérée comme peut être-pollué et un faible taux de faux positifs subsiste.

Nous sommes cependant capables d'identifier les rares cas de faux positifs. Il s'agit de version localisée de films dont le titre a été traduit. Ainsi, il peut arriver qu'une partie des utilisateurs nomme le fichier par le titre original du film alors qu'une autre partie utilise la version localisée du titre. Le tableau 7.2 illustre ce problème pour deux fichiers ayant mal été identifiés par notre méthode. Du point de vue de la métrique, les différents noms ne semblent pas liés, seule une connaissance sémantique des mots traduits ou des différents noms existant pour un même film permet de considérer que le contenu est sain, ce qu'ont bien identifié les experts.

Nous avons également regardé comment la pollution affectait chacun des 100 contenus étudiés au travers des 20 fichiers considérés pour chacun d'eux. Il apparaît que tous sont affectés avec entre 5 et la totalité des fichiers pollués par la falsification d'indexation. Ainsi, le contenu le moins pollué est « the big bang theory » avec 5 des 20 fichiers considérés pollués alors que l'ensemble des 20 fichiers sélectionnés pour « avatar » étaient pollués. La pollution n'affecte pas uniquement

| Nom de fichier choisi (X) | Nom de fichier majoritaire des sources (Y) |
|---|---|
| Il Cigno Nero Sub Ita.avi | Black.Swan.2010.DVDSCR.XviD-TiMKY.avi |
| (DivX ITA) 2012. 2009.avi | Segnali.Dal.Futuro.2009.iTALiAN.LD.DVDRip.XviD-SiLENT.CD1.avi |

TABLE 7.2 – Exemple de faux positifs lors de la détection de la pollution

| | % de fichiers pédophiles | % de fichiers pornographiques |
|---|---|---|
| pour tous fichiers | 3,6 | 21,1 |
| pour les fichiers pollués | 8,8 | 55,7 |

TABLE 7.3 – Pourcentage des fichiers supervisés potentiellement pédophile ou pornographique

des contenus soumis aux droits d'auteur puisque le mot-clé « ubuntu » désignant une distribution linux sous licence GNU GPL est également largement pollué avec 15 fichiers pollués parmi les 20.

Pour terminer, nous avons évalué la proportion de fichiers affectés par le mélange d'indexation dont au moins un des titres proposés indique un contenu pédophile ou pornographique. Nous avons recherché pour cela les mots-clés pédophiles explicites définis dans le tableau 6.2 (disponible en section 6.2.1), et des mots-clés à caractère pornographiques utilisés par le logiciel ProCon [21] pour réaliser un contrôle parental. Les résultats sont présentés dans le tableau 7.3. Il apparaît que la majorité des fichiers pollués par la falsification d'indexation sont potentiellement des fichiers pornographiques. Plus grâve, 8.8% des fichiers pollués peuvent contenir au final un contenu pédophile ce qui, étant donné la popularité des fichiers étudiés, implique une diffusion importante de ces contenus illégaux à l'insu des utilisateurs et peut générer de nombreux faux positifs en cas de supervision inadaptée.

### 7.1.3 Modification du mécanisme d'indexation

Nous avons montré que notre méthode de détection de la pollution par falsification d'indexation était très fiable. Elle pourrait ainsi constituer un mécanisme de défense, ou au moins de prévention en indiquant à un utilisateur s'apprêtant à télécharger un fichier si celui-ci est suspecté de pollution. Cependant la structure actuelle du mécanisme d'indexation de KAD ne permet pas d'obtenir les informations nécessaires à l'application de notre métrique avant de télécharger un fichier. Cette détection tardive est dommageable car l'utilisateur peut être amené à être supervisé accédant à un fichier non désiré. Dans le cas de petits fichiers, le téléchargement peut de plus être très rapide et s'effectuer avant la détection de la pollution. Notre méthode est également très coûteuse d'un point de vue réseau puisque des centaines de connexions TCP doivent être initiées afin de récupérer les noms de fichiers des sources, et ce, sans réelle intention de téléchargement dans le cas où le contenu est pollué.

Il serait cependant facile de rendre cette détection possible au niveau de la DHT. Il suffit pour cela de modifier très légèrement le protocole d'indexation de KAD en ajoutant un tag contenant le nom de fichier lorsqu'un pair se publie comme source de celui-ci. Les pairs chargés de l'indexation des sources du fichier reçoivent alors les noms de fichier contradictoires et peuvent appliquer notre métrique pour détecter la pollution. Dès lors, un simple message UDP serait suffisant pour obtenir l'indice de pollution d'un fichier en contactant les pairs proches de l'identifiant du fichier sur la DHT.

Cette solution a cependant une limite forte résultant d'un autre problème de sécurité majeur affectant les DHT : l'insertion localisée de pairs pouvant prendre le contrôle des références indexées. En effet, si des pairs malveillants sont insérés à proximité de l'identifiant du fichier dont l'indexation est mélangée, ceux-ci peuvent facilement mentir sur la valeur réelle de l'indice de pollution afin de la masquer et rendre cette protection caduque.

21. `https://addons.mozilla.org/fr/firefox/addon/procon-latte/`

### Conclusion

Nous avons identifié dans cette section une nouvelle forme de pollution affectant largement le réseau P2P KAD, appelée falsification d'indexation. Celle-ci est très néfaste car elle amène un utilisateur à télécharger un contenu indésirable et potentiellement dangereux. L'étude de 2000 fichiers parmi les plus populaires de KAD a montré que plus de 41% d'entre eux sont victimes de cette forme de pollution. Cette quantification de la pollution est la première réalisée depuis l'étude menée sur Overnet en 2006 [LNR06] et montre que la pollution des réseaux P2P reste un problème majeur aujourd'hui, la pollution des contenus étant encore plus présente et néfaste qu'alors. Nous avons proposé et validé une métrique basée sur le coefficient de similarité de Dice qui est capable de détecter le mélange d'indexation avec une grande fiabilité. Cette détection est cependant réalisée tardivement lors du début du téléchargement mais pourrait faire partie intégrante du mécanisme d'indexation au prix d'une modification mineure du protocole de KAD. Cette solution contre la pollution est cependant vulnérable à un problème plus large affectant les données indexées dans la DHT à savoir, l'insertion ciblée de pairs malveillants. Nous allons étudier plus précisément ce problème et proposer une solution dans la suite de ce manuscrit.

## 7.2 Détection des placements de pairs suspects

### Introduction

Nous proposons ensuite de détecter les pairs suspects dans le réseau P2P KAD. Pour cela nous réalisons une cartographie du réseau grâce à un explorateur spécifiquement conçu pour obtenir une image très précise de la DHT. Nous analysons ensuite les résultats afin de détecter deux types de positionnements suspects selon qu'ils impliquent localement un groupe de pairs malveillants ou uniquement un seul pair. Nous constatons ainsi pour la première fois la réalité de certaines attaques publiées et pouvons estimer leur nombre au sein du réseau.

#### Fonctionnement des attaques ciblées

Nous avons déjà présenté différents types d'attaques sur la DHT de KAD. D'une part dans l'état de l'art, en présentant plusieurs travaux dont ceux de Steiner et al [SENB07a] ayant mis en œuvre le principe de l'attaque Sybil dans KAD insérant de nombreux pairs dans une zone, mais aussi ceux de Memon et al [MRGS09] insérant des pairs à proximité d'autres pour en attirer une copie des messages. Le vecteur d'attaque utilisé ici est la table de routage des pairs, les Sybils s'annonçant directement pour se propager. D'autre part, dans les chapitre 4 et 5, nous avons montré que malgré certains mécanismes de protection ayant été implantés pour protéger la table de routage des attaques massives, le fait d'insérer quelques sondes distribuées sur la DHT extrêmement proches d'une information, permettait d'attirer toutes les requêtes de service à l'issue du processus de localisation comme l'illustre le schéma 7.7.

Dès lors, plusieurs applications exploitant cette vulnérabilité sont possibles telles que la surveillance des échanges [SENB07a] [MRGS09], la pollution [LNR06] [LMSW10], la suppression d'informations [SENB07a] (éclipse) ou encore le déni de service distribué [NR06] [SENB07a] [WTCT+08]. De même l'architecture HAMACK, présentée dans le chapitre 5, pourrait être détournée de sa fonction première pour annoncer et propager de faux fichiers dangereux (virus...) à l'insu des utilisateurs. L'insertion de nœuds malveillants constitue donc une véritable menace pour les utilisateurs des réseaux P2P et de KAD.

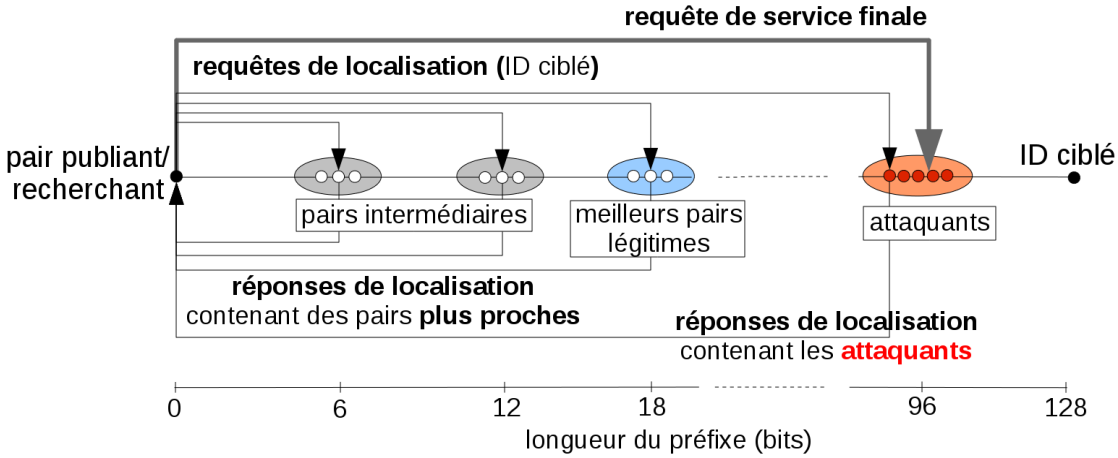


FIGURE 7.7 – Prise de contrôle d'une référence sur la DHT de KAD

#### Précédentes explorations

Un explorateur ou "*crawler*" est un outil capable de découvrir l'ensemble des pairs d'un réseau et de stocker les différentes informations les concernant.

Plusieurs explorations du réseau KAD ont déjà été réalisées à diverses fins. Les auteurs de [WTCT+08] et [SENB07a] découvrent ainsi les pairs du réseau à des fins d'attaque. Pour chaque pair découvert, ils interrogent ce dernier en émettant de nombreuses requêtes de localisation (kademlia request) vers des identifiants pré-calculés de manière à obtenir tous les contacts de la table de routage du pair interrogé. Ces informations servent ensuite à insérer des Sybils [WTCT+08] ou à corrompre les références de contacts existants [SENB07a]. Utilisant le même explorateur *Blizzard* que [WTCT+08], [SENB07b] réalise des explorations périodiques de la DHT de manière à étudier certaines caractéristiques des pairs dans le temps.

Les auteurs de [YFX+09] utilisent une autre approche basée sur l'interrogation de contacts par des requêtes d'amorçage (bootstrap request). Cette approche est sensée être plus performante (20 contacts retournés par requête d'amorçage contre 11 pour celle de localisation). Cependant les contacts obtenus sont choisis aléatoirement dans la table alors que les requêtes de localisation spécifient une adresse cible permettant de contrôler le parcours des tables. Les résultats de cette exploration ont mis en évidence un nombre important de pairs (20%) partageant leur identifiant dont les auteurs étudient les causes possibles.

Si de nombreuses observations du réseau KAD ont été réalisées, aucune jusqu'à présent ne s'est intéressée à recenser les attaques pouvant affecter la DHT. De même, aucune n'estime l'efficacité de leur explorateur dont les algorithmes sont peu détaillés, s'ils sont mentionnés.

### 7.2.1 Exploration du réseau

#### 7.2.1.1 Méthode d'exploration

La conception de notre explorateur, réalisé avec la participation de Christopher Hénard, vise deux objectifs. D'une part, obtenir une vision précise du réseau, et d'autre part, limiter l'empreinte de l'exploration sur le réseau en limitant le nombre de requêtes envoyées à chaque pair. Ceci permet en outre d'obtenir une exploration compatible avec les limitations implantées dans les derniers clients, contrairement aux précédentes stratégies d'exploration désormais lim-

itées, notamment par rapport à la protection contre l'inondation empêchant un pair de recevoir rapidement des messages d'une même source.

Notre méthode d'exploration se divise en trois phases décrites ci-après.

#### Amorçage

La phase d'amorçage sert à obtenir une première image imprécise de l'ensemble de la DHT. Pour cela, des requêtes d'amorçage (bootstrap) sont émises. Les requêtes d'amorçage permettent d'obtenir 20 contacts tirés aléatoirement dans la table de routage du pair sollicité et sont donc parfaitement adaptées à une première découverte globale de la DHT. De nouveaux contacts sont ainsi progressivement interrogés au fur et à mesure des réponses jusqu'à ce que 500000 contacts aient été découverts dont au moins 500 par zone [22]. Au delà de cette valeur, les contacts retournés étant sélectionnés au hasard, il est de plus en plus difficile d'apprendre de nouveaux contacts par cette méthode.

#### Exploration complète

Ensuite, chaque zone est explorée avec précision grâce aux requêtes de localisation (kademlia request). Un pair ainsi interrogé retourne les 4 contacts les plus proches connus de l'identifiant spécifié en paramètre. Afin de découvrir l'ensemble des pairs, nous générons $2^{21} \approx 2$ millions de « KADIDs cibles » uniformément répartis et envoyons pour chacun d'eux une requête de localisation au pair le plus proche déjà découvert. Ainsi, $2^{13}(2^{21}/2^8)$ KADIDs cibles sont générés dans chaque zone selon le format :

$$\underbrace{ZZZZZZZZ}_{8\ bits\ de\ la\ zone}\overbrace{FFFFFFFFFFFFF}^{13\ bits\ fixes\ de\ 0\ \ 2^{13}-1}\underbrace{RRRRRR...R}_{107\ bits\ alatoires}$$

où $Z$, $F$ et $R$ désignent respectivement des bits de zone, les bits fixés et ceux tirés aléatoirement une fois.

#### Seconde passe

Dès qu'une zone a été explorée, c'est à dire quand tous les KADIDs cibles de cette zone ont été envoyés, une seconde exploration de celle-ci a lieu pour en améliorer la cartographie. Pour chaque contact précédemment découvert, on calcule alors son voisin le plus proche dont on extrait ensuite le préfixe commun de longueur $x$ bits entre les deux KADIDs. On construit ensuite un nouveau « KADID cible » partageant ce préfixe et où les $(128 - x)$ bits restants sont aléatoires. Une requête de localisation pour ce KADID cible est finalement envoyée au contact. Cette phase permet de découvrir quelques contacts manqués lors de l'exploration complète. L'exploration se termine lorsque tous les contacts ont ainsi été interrogés sur leur voisinage immédiat.

### 7.2.1.2 Cartographie obtenue

#### Informations enregistrées

Pour chaque pair découvert, nous enregistrons les informations suivantes <KADID, adresse IP [23], port TCP, port UDP, version de KAD, état du pair>. La version de KAD fait référence

22. une zone est une subdivision artificielle de l'espace d'adressage basée sur le premier octet de poids fort des identifiants (de *0x00* à *0xFF*)

23. certaines adresses IP sont anonymisées dans le cadre de ce manuscrit

à la version du protocole implantée par le client, l'état du pair est P(*possible*), T(*tried*) ou R(*responded*) selon respectivement que le contact a juste été découvert, a été contacté ou a répondu.

```
[...]
<32FFF76959F6A7095347FB338B304330, #.#.#.#, 38060, 16905, 0, T>
<32FFFC5C4D5AE9A082871FF68B1F0D9C, #.#.#.#, 5149, 1025, 4, R>
<32FFFC5C4D5AE9A082871FF68B1F0D9C, #.#.#.#, 5149, 5159, 4, P>
Zone 33: 15196 contacts
<3300048A90460A8AAC3DD2FF542ADF98, #.#.#.#, 12399, 39949, 9, R>
<3300083A0480CFA91B8C142401DD26F2, #.#.#.#, 5611, 5621, 8, T>
<330018506569424D7CBA7133F437EDC8, #.#.#.#, 6647, 6657, 8, P>
<33002596F7AAAA4348FB4349F0A14FA4, #.#.#.#, 46318, 61632, 9, R>
<33002EF905E27753B1900BC602D29C20, #.#.#.#, 19774, 19774, 8, T>
<33004546934FABE9685674DE1598548F, #.#.#.#, 51478, 52073, 9, R>
[...]
```

### Résultats généraux

L'exploration d'une zone compte entre 13000 et 17000 contacts, le nombre total de pairs mesuré allant de 3,3M à 4,3M selon le jour et l'heure de l'exploration. D'un point de vue macroscopique, la répartition des pairs sur l'ensemble de l'espace d'adressage de la DHT est bien uniforme (figure 7.8), conformément à ce qu'on peut attendre de la majorité des pairs légitimes générant aléatoirement leur identifiant à la première connexion.

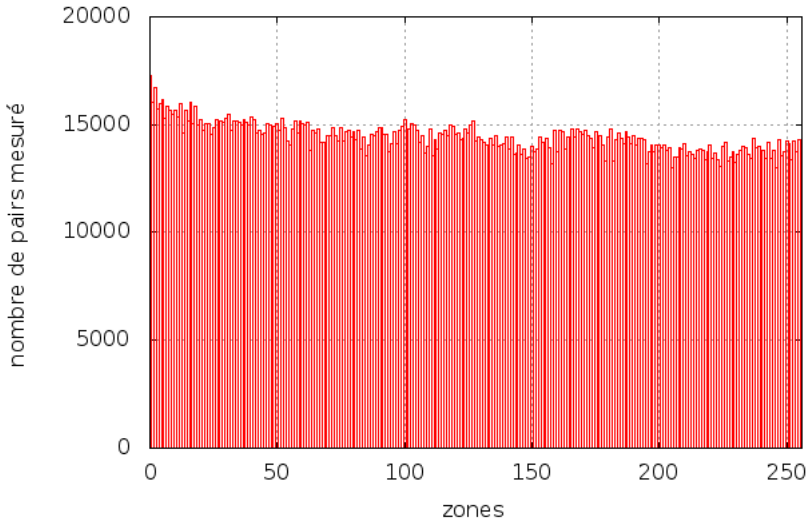


FIGURE 7.8 – Répartition des pairs sur la DHT

Nous analysons plus précisément les résultats d'exploration dans la section suivante, avec pour objectif de détecter les placements traduisant des comportements déviants. Les résultats obtenus pour les différentes explorations réalisées étant similaires, la suite de cet article utilise les données d'une exploration réalisée le 8 Juillet 2010 et comptant 3688932 pairs.

#### 7.2.1.3 Évaluation

Nous avons évalué notre explorateur de deux façons. Nous avons tout d'abord injecté 360 pairs dans KAD suivant une configuration d'attaque depuis l'infrastructure d'expérimentation distribuée PlanetLab[24]. Les Sybils sont ainsi répartis par groupe de 5 sur 72 identifiants cibles dont ils partagent au moins 96 bits, ce qui correspond au déploiement d'HAMACK présenté dans le chapitre 6. A l'issue d'une exploration du réseau concomitante à l'attaque, la totalité des pairs insérés était bien présente dans les résultats de l'exploration. L'extrait ci-dessous montre une analyse des données recherchant les pairs à proximité d'identifiants donnés en paramètre (ici les 72 identifiants ciblés).

```
[...]
KADID 71: 19856E29730F11CA0E0C210630ADCB36
<19856E29730F11CA0E0C210621142E70, 62.108.171.74, 14337, 13602, 8, T> [prefix = 99]
<19856E29730F11CA0E0C2106546F8C89, 193.167.187.186, 14690, 13799, 8, T> [prefix = 97]
<19856E29730F11CA0E0C21065622F60F, 155.245.47.241, 13953, 13779, 8, T> [prefix = 97]
<19856E29730F11CA0E0C210676E74885, 212.51.218.235, 13897, 14465, 8, T> [prefix = 97]
<19856E29730F11CA0E0C21069636476A, 129.97.74.14, 14308, 13853, 8, T> [prefix = 96]
KADID 72: EBCBA6D72037ED01F56809A9FFE6A86E
<EBCBA6D72037ED01F56809A9268DA7FB, 155.245.47.241, 13915, 13842, 8, T> [prefix = 96]
<EBCBA6D72037ED01F56809A94519B1D4, 129.97.74.14, 14029, 13914, 8, T> [prefix = 96]
<EBCBA6D72037ED01F56809A9702F72B7, 193.167.187.186, 13666, 14427, 8, T> [prefix = 96]
<EBCBA6D72037ED01F56809A9892C91A4, 62.108.171.74, 13853, 14683, 8, T> [prefix = 97]
<EBCBA6D72037ED01F56809A9BAD2A19E, 212.51.218.235, 13861, 13939, 8, R> [prefix = 97]

72/72 of the proposed KADIDs are targeted with at least 96 bits by:
37 IP addresses (showing 361 unique KADIDs in the whole crawler's data)
21 subnets /24  (showing 362 unique KADIDs in the whole crawler's data)
```

Une seconde évaluation a consisté à modifier un client KAD afin d'afficher la liste des contacts trouvés lors d'une publication et à explorer conjointement la zone correspondante. L'ensemble des pairs trouvés par le client aMule l'a également été par l'explorateur, ce qui tend également à montrer l'efficacité de notre exploration.

### 7.2.2 Détection par densité des pairs

Comme expliqué précédemment, une attaque sur la DHT implique l'insertion d'un ou plusieurs pairs à proximité de l'identifiant ciblé, afin d'attirer tout ou partie des requêtes à son attention. Pour une meilleure efficacité, plusieurs pairs peuvent être insérés conjointement afin d'attirer davantage de requêtes.

Notre première analyse s'intéresse à localiser de tels groupes de pairs sur la DHT. Nous cherchons ainsi à détecter les couples de pairs dont la distance trop proche traduit un placement intentionnel à proximité d'un tierce identifiant plutôt qu'un choix aléatoire de leurs identifiants.

$$F(x) = \frac{N}{2^x} \tag{7.4}$$

Soit $F$ la fonction donnant le nombre moyen de pairs partageant $x$ bits avec un pair courant étant donné un nombre total $N$ de pairs dans le réseau. Nous considérons un nombre de 4 millions de pairs connectés simultanément. Le tableau 7.4 en présente certaines valeurs pour

24. http ://www.planet-lab.org/

| Nombre de bits en commun | Nombre moyen de pairs |
|---|---|
| 1 | 2,000,000 |
| 8 | 15625 |
| 12 | 976.5 |
| 16 | 61 |
| 18 | 15,25 |
| 20 | 3.8 |
| 24 | 0.24 |
| 28 | 0.015 |
| 32 | $9.32 * 10^{-4}$ |
| 64 | $2.17 * 10^{-13}$ |
| 96 | $5.05 * 10^{-23}$ |

TABLE 7.4 – Nombre moyen de pairs partageant un préfixe donné avec une référence pour une DHT de 4 millions

$N = 4 \times 10^6$ et $x \in [1; 128]$. De plus, le préfixe moyen partagé entre deux pairs consécutifs est de $d_{moy} = log2(N) = 21.93$ bits.

Étant donné notre exploration de la DHT, nous avons calculé le préfixe commun entre chaque pair et son plus proche voisin, les résultats sont présentés par la figure 7.9. Si les préfixes jusqu'à 35 bits sont communément partagés entre voisins et ne permettent pas de détecter les attaques, les contacts partageant davantage de bits traduisent un placement intentionnel. Le premier graphe de la figure 7.10 illustre cette déviation de la norme théorique (équation 1) pour les contacts partageant entre 22 et 45 bits. Plus le préfixe commun est élevé, plus l'espérance de trouver de tels voisins est faible et traduit un placement intentionnel ce qui est illustré par le second graphe de la figure 7.10. Nous avons ainsi relevé 426 groupes distincts impliquant des contacts anormalement proches (partageant un préfixe unique compris entre 35 et 127 bits) et traduisant autant d'attaques groupées potentielles. Cependant, l'écart le plus important concerne le préfixe de 128 bits (1 million de pairs) qui correspond aux pairs partageant exactement le même identifiant et qui mérite une analyse à part.

### Identifiants partagés

En effet, sur les 3688932 pairs trouvés lors de l'exploration et caractérisés par le quadruplet (KADID, adresse IP, port TCP, prt UDP), on ne dénombre après analyse que 2613963 KADIDs différents. Tout comme [YFX+09], nous constatons donc l'existence de KADIDs partagés par plusieurs pairs. Plus précisément, parmi les KADIDs relevés :

- 82,36% (2152900) des KADIDs sont utilisés par un pair unique,
- 17.64% (461063) des KADIDs sont partagés par plusieurs pairs dont :
  - 10,42% des KADIDs sont communs à 2 pairs,
  - 2,85% des KADIDs sont communs à 3 pairs,
  - les pourcentages décroissants jusqu'à 1 KADID partagé par 259 pairs.

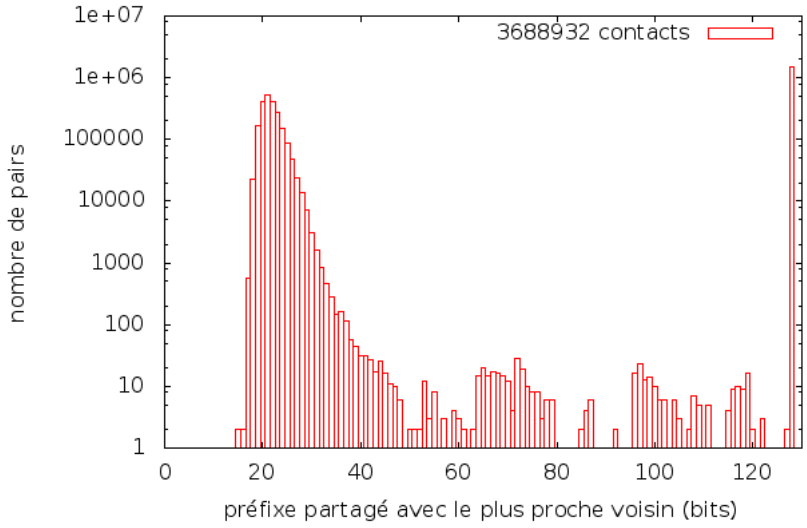


FIGURE 7.9 – Répartition des préfixes entre voisins sur la DHT

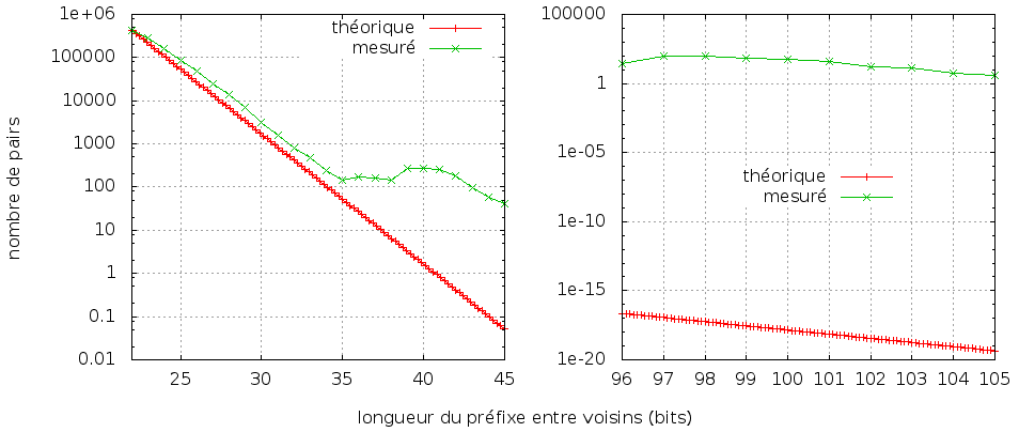


FIGURE 7.10 – Nombre absolu de pair théorique et mesuré partageant un préfixe

Le partage de préfixe peut traduire une attaque si plusieurs pairs sont insérés exactement avec l'identifiant de la cible. Cependant, il peut également traduire un changement bénin de configuration d'un pair. En effet, un pair changeant d'adresse IP (allocation dynamique d'adresse, mobilité), ou de port de communication durant sa connexion au réseau apparaîtra deux fois avec le même identifiant le temps que la DHT mette à jour ses références. Afin d'éviter de compter ces cas, nous supprimons de la liste des identifiants suspects les cas pour lesquels deux pairs partagent un identifiant où seule l'adresse IP ou seul le port diffèrent entre les deux pairs. Ainsi parmi les KADIDs partagés entre deux pairs (272149) :

- 49.73% ne diffèrent que par l'adresse IP (ports UDP et TCP identiques),
- 26.91% ne diffèrent que par le port UDP,
- 1.44% ne diffèrent que par le port TCP,
- 21.92% sont suspects.

Par cette méthode, 248569 identifiants différents peuvent être suspectés. Malgré les précautions prises, ce chiffre peut être soumis à des faux positifs. Nous proposons une dernière estimation plus fiable des attaques affectant KAD, car basée sur les contenus et non uniquement sur les pairs.

### 7.2.3 Détection par proximité aux ressources

Les analyses précédentes ont une limite importante : elles permettent d'identifier des attributions d'identifiants suspects sans pour autant pouvoir les corréler à un contenu précis. Par ailleurs, les analyses précédentes étant basées sur des proximités entre pairs, au moins deux pairs doivent être insérés pour être détectés, les attaques n'impliquant qu'un pair ne peuvent être détectées en analysant la densité des pairs.

Une manière fiable de détecter les attaques est donc de pouvoir mettre en évidence la proximité anormale des pairs malveillants par rapport à une ressource plutôt que la proximité des pairs entre eux. La difficulté de cette approche est que les identifiants des ressources ne sont pas connus a priori. Pour appliquer cette méthode, nous avons extrait des mots-clés de contenus pouvant être partagés sur KAD depuis plusieurs sources d'information (meilleures ventes Amazon, iTunes, fichiers populaires sur ThePirateBay). Nous avons ensuite calculé l'identifiant de chacun des mots-clés composant les différents titres par la fonction MD4 utilisée par KAD. Nous avons finalement recherché les contacts anormalement proches de ces identifiants (partageant un préfixe supérieur à 30 bits) dans les données des explorations. Un extrait des résultats est donné ci-après.

```
[...]
twilight 4D62D26BB2A686195DA7078D3720F60A
<4D62D26BB2A686195DA7078D3720F632, X.Y.#.#, 7290, 7294, 8, R> [prefix = 122]
soundtrack AC213377BB53F608390BD94A6AE6DD35
<AC213377BB53F608390BD94A82582F42, #.#.#.#, 5003, 5002, 8, R> [prefix = 96]
harry 770CF5279AB34348C8FECF9672747B94
<770CF5279AB34348C8FECF96524D8CDE, #.#.#.#, 5003, 5002, 8, P> [prefix = 98]
robin B9DF47E5BFAD75F8EE5E3F50EA217983
<B9DF47E5BFAD75F8EE5E3F5051F34AA8, #.#.#.#, 5003, 5002, 8, R> [prefix = 96]
<B9DF47E5BFAD75F8EE5E3F50EA21799F, X.Y.#.#, 7290, 7294, 8, R> [prefix = 123]
[...]

216/888 of the proposed keywords are targeted with at least 96 bits by:
44 IP addresses (showing 2119 unique KADIDs in the whole crawler's data)
41 subnets /24  (showing 2155 unique KADIDs in the whole crawler's data)
```

Sur les 888 mots-clés utilisés pour cette analyse, un quart d'entre eux a un pair proche partageant au moins 96 bits ce qui, étant donné l'espérance de trouver un pair légitime avec un tel préfixe (voir tableau 7.4) traduit sans équivoque un placement intentionnel et un comportement malveillant. Un échantillon de ces mots-clés est donné dans le tableau 7.5, certains faisant référence à un contenu explicite, d'autres étant plus génériques.

| mot-clé | meilleur préfixe |
|---|---|
| avatar | 126 |
| invictus | 123 |
| sherlock | 122 |
| princess | 122 |
| frog | 98 |
| ncis | 96 |
| nero | 96 |
| ... | ... |

| mot-clé | meilleur préfixe |
|---|---|
| nine | 122 |
| love | 122 |
| american | 97 |
| russian | 97 |
| the | 96 |
| black | 96 |
| pirate | 96 |
| ... | ... |

TABLE 7.5 – Exemples de mots-clés attaqués

Pour les pairs malveillants ainsi détectés, nous avons recherché leur présence sur l'ensemble de la DHT afin de découvrir d'autres identifiants ciblés et absents de la liste initiale de mots-clés. Nous avons ainsi relevé que les seuls mots-clés recherchés ne représentent que 10% de la présence de ces noeuds (adresse IP + port) sur la DHT. En comptant les 216 identifiants de mots-clés attaqués parmi notre liste, ces noeuds sont au total présents sur 2119 KADIDs. Ce résultat montre clairement que de nombreux contenus de la DHT sont attaqués, parmi les plus populaires. De plus, des configurations d'attaques émergent rapidement des données. Par exemple, parmi les 216 identifiants, 205 sont ciblés par des pairs ayant exactement les ports suivants : UDP=5003, TCP=5002, un préfixe de 96bits mais des adresses IP distribuées sur plusieurs réseaux. Un autre attaquant cible 16 identifiants parmi les 216 en utilisant des pairs ayant exactement les ports : UDP=7290, TCP=7294, un préfixe de 122bits et une adresse IP venant d'un sous réseau spécifique ( 16 de la forme `X.Y.#.#`). Même si certaines configurations utilisées par un même attaquant semblent évidentes, une caractérisation plus précise de celles-ci (adresses IP, ports, nombre de noeuds insérés, distance des noeuds, etc.) par apprentissage permettrait d'identifier plus précisément le nombre et l'importance des différents attaquants. Une étude approfondie du comportement de ces pairs par des communications directes permettrait en outre de découvrir la nature de leur activité (surveillance, pollution, DDoS, etc.).

Bien que cette estimation soit fiable, elle a également des limites, notamment quant au jeu de caractères utilisé par les mots-clés. Ceux considérés pour notre expérience utilisent en effet l'alphabet latin ; or, KAD est pour moitié utilisé en Asie. Les pairs ciblant spécifiquement des contenus décrits avec des caractères asiatiques peuvent échapper à cette analyse. Par ailleurs d'autres attaques peuvent cibler exclusivement les fichiers et non les mots-clés.

## Conclusion

Alors que plusieurs attaques pouvant affecter le réseau KAD ont été décrites dans de précédents travaux et que de nombreuses observations de ce réseau ont déjà été réalisées, aucune d'entre elles ne s'était intéressée jusqu'alors aux questions de sécurité affectant la DHT. Nous nous sommes tout d'abord intéressés à la question de la pollution et avons observé que le mécanisme d'indexation de KAD est largement corrompu car 2/3 des contenus populaires étudiés

sont pollués. Nous également identifié une nouvelle forme de pollution, appelée mélange d'indexation, qui est particulièrement dangereuse car amenant un utilisateur à télécharger un fichier indésirable. Celle-ci effecte 41% des contenus étudiés. Nous avons cependant montré qu'il est possible de détecter de manière fiable les fichiers pollués en étudiant la similarité des différents noms affichés par les sources durant le téléchargement. Une modification mineure du protocole KAD permettrait en outre une détection plus rapide et moins coûteuse en ressources. Cependant celle-ci est inapplicable tant que des nœuds peuvent être insérés pour prendre le contrôle du mécanisme d'indexation.

Nous avons donc ensuite estimé les positionnements anormaux des pairs traduisant de telles attaques ciblées. Nous avons tout d'abord développé et évalué un explorateur capable de découvrir précisément la DHT de KAD, malgré les limitations récemment incluses dans les clients. Une première analyse considérant la proximité entre les identifiants des pairs a mis en évidence des regroupements de pairs anormaux, quelques pairs étant trop proches les uns des autres (426) mais la grande majorité d'entre eux partageant un même identifiant (248569). Une seconde analyse basée sur l'étude de mots-clés populaires a mis en évidence qu'une grande proportion de ceux-ci est attaquée. Les pairs impliqués sont d'ailleurs présents sur de nombreux identifiants de la DHT (2119) et des configurations d'attaques peuvent être clairement mises en évidence. Concernant les mots-clés ciblés, les attaquants insèrent un seul pair extrêmement proche du contenu (96 bits ou 122 bits communs) mais ne semblent en revanche pas réaliser d'attaques impliquant plusieurs pairs.

Le chapitre suivant présente une méthode capable de protéger les pairs légitimes contre l'insertion de pairs malveillants tout en étant compatible avec les contraintes du réseau P2P KAD à savoir : l'absence de composant centralisé et la rétro-compatibilité entre clients.

# Chapitre 8

# Protection de la DHT en considérant la distribution des identifiants

**Sommaire**



## Introduction

Malgré les différentes protections insérées dans KAD (évaluées dans le chapitre 4 [CCF09a]) la DHT peut toujours être contrôlée par une attaque distribuée ciblant précisément certaines références. Ce contrôle, s'il est utilisé à des fins malveillantes, peut entraîner d'importants problèmes de sécurité et de qualité de service dans le réseau. Parmi les applications possibles, outre la surveillance du trafic P2P, l'attaquant peut éclipser des fichiers du système d'indexation ou encore lancer des attaques de déni de service distribué vers un hôte cible. Le contrôle de la DHT permet également une pollution massive et indétectable des références ciblées, amenant les utilisateurs à télécharger un contenu indésirable, ce qui pose des problèmes évidents de sécurité. Par analogie avec notre architecture de supervision, l'utilisation de quelques machines distribuées ($\sim 20$ nœuds) suffit pour contrôler le mécanisme d'indexation. Cette faiblesse peut donc entrainer des problèmes de sécurité et de respect de la vie privée pour les utilisateurs du réseau.

| Type | KADID | préfixe |
|---|---|---|
| référence ciblée | 477221265829086C74988C40EFE63DAF | - |
| attaquant | 477221265829086C74988C4070D6E0F1 | 96 bits |
| pair légitime | 477229E3D7CFC729F337ABBB69C983C6 | 20 bits |

TABLE 8.1 – Exemples d'identifiants de pairs

Le fait qu'un pair puisse librement choisir son identifiant couplé au processus de localisation très performant constitue donc actuellement le problème de sécurité majeur de KAD et plus généralement des DHT ouvertes (Gnutella DHT, Bittorrent DHT...). Des mécanismes de protections complémentaires doivent donc être conçus pour empêcher cette attaque. En particulier, pour empêcher les attaquants de se positionner extrêmement près d'une référence pour en attirer les requêtes, tel qu'illustré par le tableau 8.1 et le schéma 7.7. Un attaquant peut ainsi partager un grand nombre de bits communs (appelé préfixe, de longueur 96 bits dans l'exemple) avec une référence en faisant un pair privilégié pour recevoir les requêtes. Le mécanisme de localisation reste donc encore vulnérable aux attaques, ce qu'exploitent les travaux les plus récents sur KAD : HAMACK à des fins de supervision ou [KLR09] pour réaliser des attaques éclipses. Au contraire, [SENB07a] et [WTCT+08] ciblaient davantage la table de routage des pairs, alors vulnérable, pour réaliser leurs attaques.

Par ailleurs, les différentes solutions existantes visant à sécuriser les DHT et présentées dans le chapitre 3 ont montré de nombreuses limites. La plupart imposent des contraintes devant exister dès la création de la DHT (réseaux sociaux, certification distribuée), d'autres faisant intervenir une entité centrale (de certification des identifiants) sont incompatibles avec le paradigme pair à pair ou introduisent un surcoût important (puzzle cryptographiques). Aucune solution ne répond aux contraintes de KAD qui doit rester entièrement distribué et garder une rétro-compatibilité avec les anciennes versions des clients. Ces contraintes limitent fortement les modifications possibles de la DHT et rendent la conception de protections d'autant plus difficile.

Notre objectif dans ce chapitre est de concevoir de nouvelles protections contre les attaques ciblées qui soient adaptées à une DHT déjà déployée et ouverte telle que KAD. Deux stratégies de défense sont possibles : soit contraindre les identifiants des pairs, soit détecter les attaquants avant d'émettre des requêtes de service. La première solution mène au problème de la gestion des identités dans les réseaux P2P, largement étudié et présentant les limites déjà mentionnées ce qui rend cette approche peu indiquée pour une application réelle. De plus, changer la gestion des identités dans KAD nécessiterait, outre de profondes modifications de l'architecture et du code source, une scission du réseau entre les pairs ayant un identifiant validé et les autres. Nous présentons dans ce chapitre [CCF10a] une nouvelle approche innovante et très efficace pour empêcher les attaques ciblées. Celle-ci consiste à détecter les attaques en comparant la distribution anormale des identifiants induite par l'insertion des attaquants autour des références ciblées, et ce par rapport à une distribution théorique. Nous présentons ensuite les contre-mesures permettant d'éviter les pairs malveillants.

## 8.1 Analyse de distributions saines

Les attaques ciblées sont très efficaces sur la DHT de KAD car les pairs trouvés durant la phase de localisation sont toujours triés selon leur distance par rapport à la cible, puis interrogés dans cet ordre. Dès lors, si 10 pairs ou davantage opérés par un attaquant sont trouvés comme

étant les plus proches d'une référence, ils sont en mesure d'attirer toutes les requêtes relatives. Cet algorithme garantit la performance des recherches et ne peut être modifié aisément. Cependant, l'insertion des pairs malveillants autour d'une référence a des effets mesurables. Nous montrons en effet qu'une attaque ciblée introduit des anomalies dans la distribution des identifiants autour de la cible, à la fois sur la densité et la proximité des pairs dans son voisinage. Ainsi, pour détecter les attaques, nous considérons les incohérences des distributions d'identifiants résultant d'attaques. En connaissant la distribution théorique des identifiants et celle obtenue après l'étape de localisation, nous pouvons alors détecter les attaques avant la réalisation du service sans aucun surcoût.

### 8.1.1 Distribution théorique des identifiants

Dans KAD, comme dans la plupart des DHT, le comportement normal d'un pair est de choisir aléatoirement son identifiant (les 128 bits composant le KADID) à sa première connexion. Si cette distribution aléatoire l'est vraiment, la distance moyenne entre deux pairs est uniquement fonction du nombre de pairs participant à la DHT. De plus, nous pouvons calculer le nombre de pairs potentiels partageant un certain préfixe avec une référence donnée, et en déduire une probabilité pour un pair trouvé à l'issue de la localisation de pouvoir légitimement exister étant donné son préfixe.

Soit $F$ la fonction donnant le nombre moyen de pairs partageant au moins $x$ bits avec une référence et $N$ le nombre total de pairs dans le réseau. Dans notre application à KAD, nous considérons un nombre de 4 millions de pairs connectés simultanément.

$$F(x) = \frac{N}{2^x} \tag{8.1}$$

avec $N = 4 \times 10^6$ et $x \in [1; 128]$.

Plus précisément, la distribution théorique des préfixes autour d'une référence suit la loi géométrique de paramètre $p = 1/2$. Le tableau 7.4 en présente certaines valeurs.

$$P(X = k) = (1 - p)^{k-1} p \tag{8.2}$$

Le graphique 8.1 montre le nombre de pairs légitimes potentiels existant dans la DHT étant donné la taille du préfixe commun (c'est à dire nombre de bits consécutifs de poids fort similaires) avec une référence donnée. Nous pouvons observer que le nombre de pairs potentiels suit parfaitement une fonction logarithmique. Cependant, même si la distribution théorique des identifiants est bien connue à l'échelle du réseau, il n'est pas certain que les meilleurs contacts trouvés à l'issue du processus de localisation reflètent parfaitement cette distribution. Les pairs n'ont en effet qu'une vue partielle du réseau à travers leur table de routage et comptent sur le processus de localisation pour trouver les pairs les plus proches de l'objet recherché. Il est difficile d'inférer si les pairs trouvés à l'issue de la localisation sont en réalité les meilleurs pairs potentiels existant dans la DHT. De plus, plusieurs paramètres peuvent affecter l'efficacité du routage (temps de connexion, distance à l'objet...) et doivent être considérés.

### 8.1.2 Distribution réelle des identifiants

Pour confronter les résultats de l'algorithme de localisation à la distribution théorique des identifiants, nous avons mené une expérience consistant à réaliser de nombreuses requêtes de localisation vers des références aléatoires (vierge de toute attaque), et nous avons enregistré, à l'issue de chaque localisation, les identifiants des meilleurs contacts trouvés. Nous mesurons ainsi

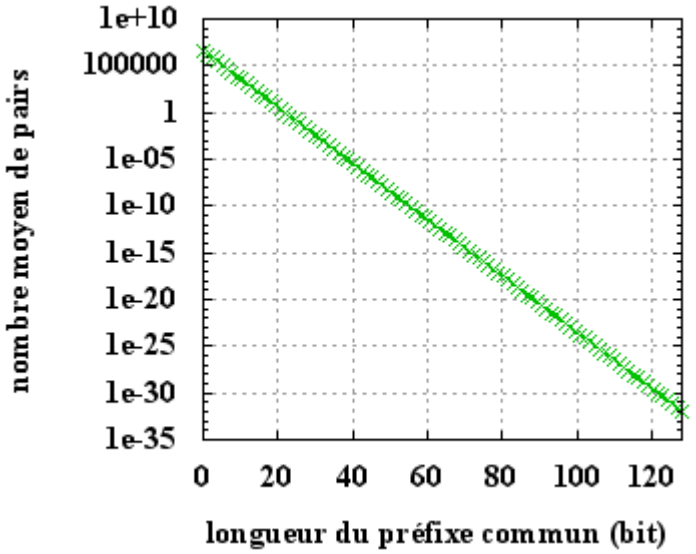


FIGURE 8.1 – Nombre moyen de pairs partageant un préfixe donné avec une référence

la distribution réelle des identifiants telle que perçue à la fin d'une localisation, c'est à dire, avant que les requêtes de service ne soient émises vers les 10 meilleurs contacts. Nous ne considérons que la distribution des 10 meilleurs contacts pour notre analyse, même si davantage sont trouvés par la localisation.

Nous avons publié 100 fichiers sur une durée de 60 heures. Pour être certain qu'aucune attaque n'affecte les références publiées, les deux mots-clés associés à chaque fichier ainsi que son contenu sont générés aléatoirement. Notre client KAD publie les mots-clés toutes les 24 heures et les fichiers toutes les 5 heures, l'expérience a donc généré approximativement 1800 ($12 \times 100 + 3 \times 200$) publications sur la DHT. Pour chacune, nous calculons le préfixe partagé entre les contacts et la référence publiée.

Le graphique 8.2 montre que la longueur moyenne du préfixe commun des 10 meilleurs contacts trouvés se situe entre 18 bits et 19 bits, ce qui constitue un très bon résultat étant donné le nombre moyen de pairs potentiels partageant ce préfixe dans une DHT de 4 millions (entre 15.25 et 7.6 d'après le tableau 8.2). Ceci confirme la très bonne efficacité de la localisation réalisée par KAD puisque les 10 meilleurs pairs trouvés sont proches des 10 meilleurs existant en réalité dans la DHT. Le graphique 8.2 montre également que le temps passé dans le réseau n'a pas d'influence sur le préfixe moyen des contacts trouvés, malgré une amélioration de la consistance de la table de routage du client avec le temps. Nous pouvons également remarquer une légère variation périodique de la taille moyenne du préfixe pouvant être attribué à l'évolution journalière du nombre de pairs dans le réseau. Sa faible amplitude est expliquée par le fait que KAD est principalement utilisé en Europe et en Chine, les heures d'activités de ses deux régions étant complémentaires.

Nous avons étudié plusieurs autres paramètres pouvant affecter le routage et donc la proximité des meilleurs contacts trouvés. Ainsi, la distance entre le pair publiant et la référence n'a pas d'influence, le graphique 8.4 montrant que le préfixe moyen reste stable pour toute distance. Le processus de localisation masque donc complètement l'avantage fourni par une table de routage plus précise quand le pair et la référence sont proches. Le type de contenu publié (mot-clé ou fichier) et le type du service demandé (publication ou recherche) n'ont pas d'impact sur la qualité

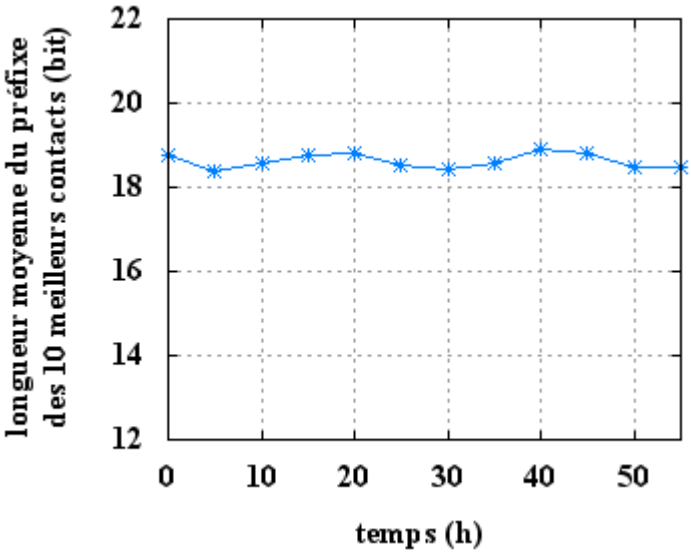


FIGURE 8.2 – Préfixe moyen des 10 meilleurs contacts trouvés en fonction du temps

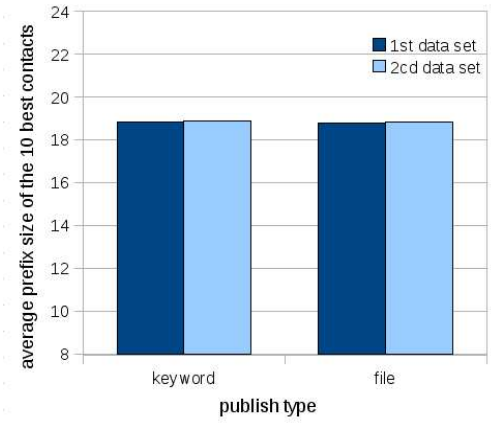


FIGURE 8.3 – Préfixe moyen des 10 meilleurs contacts trouvés en fonction du type de la publication

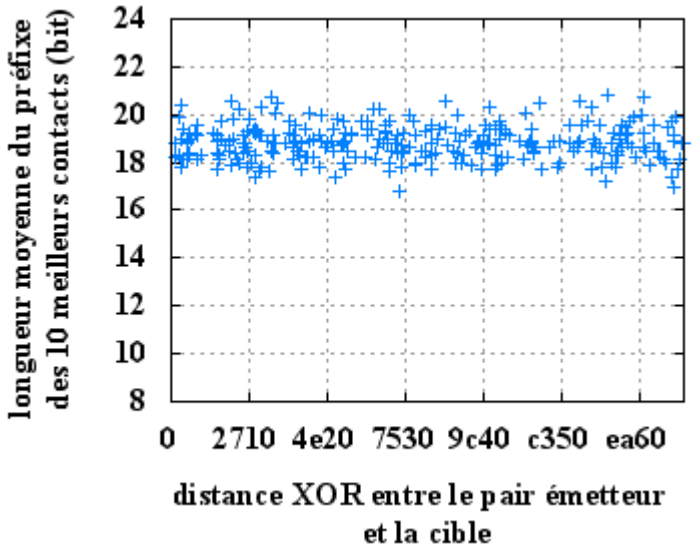


FIGURE 8.4 – Préfixe moyen des 10 meilleurs contacts trouvés en fonction de la distance entre le pair publiant et la référence

des contacts trouvés.

Enfin, le résultat le plus important concerne la distribution mesurée des identifiants des meilleurs pairs présentée par le graphique 8.5. Pour chaque publication, nous avons compté le nombre de pairs partageant un préfixe donné avec la référence courante. Il apparaît qu'au delà du préfixe de 17 bits, la distribution mesurée suit exactement la distribution théorique des identifiants des pairs décrite dans la sous-section précédente. Par conséquent, la procédure de localisation de KAD est assez performante pour donner une vue réaliste des contacts les plus proches possibles existant dans la DHT. Ce résultat montre également que la distribution aléatoire théorique des identifiants est suffisante pour caractériser les résultats obtenus à l'issue de la localisation. Même si une approche basée sur l'apprentissage des distributions saines est également possible, celle-ci n'est pas nécessaire (dans un premier temps) pour détecter les attaques.

Grâce à la connaissance de la distribution régulière des identifiants des pairs, nous définissons tout d'abord des contraintes évitant les attaques les plus évidentes. Dans un second temps, nous présentons une métrique pouvant détecter précisément les attaques restantes.

### 8.1.3 Règles de sécurité préventives

Les mécanismes de protection introduits dans KAD contre l'attaque Sybil ont montré que quelques règles passives peuvent éviter les attaques les plus simples et les plus efficaces, et ce, sans présenter d'inconvénient notable. De la même manière, les contraintes suivantes, une fois appliquées au processus de localisation, rendent les attaques ciblées plus difficiles à réaliser.

#### 8.1.3.1 Limitation des adresses IP

La première contrainte consiste à limiter le nombre de contacts découverts à l'issue du processus de localisation et partageant une même adresse IP ou venant du même sous réseau /24, ce qui est déjà appliqué pour protéger la table de routage. Ainsi, seul un pair par sous réseau

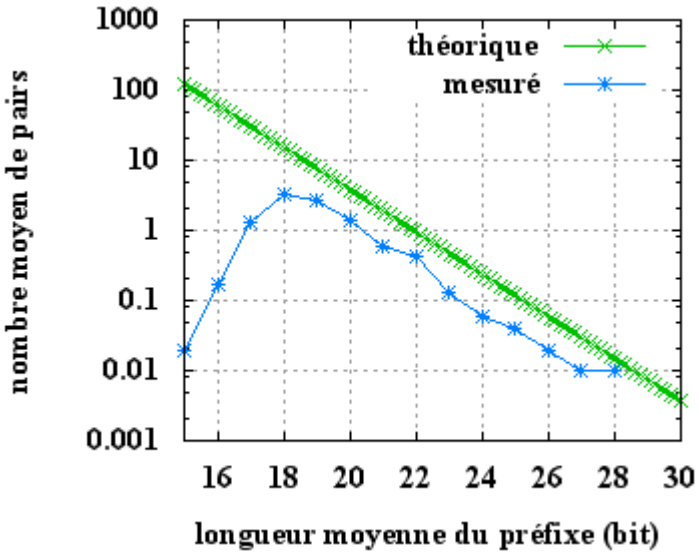


FIGURE 8.5 – Distribution du préfixe commun des 10 meilleurs contacts trouvés

| Taille du préfixe commun | Nombre moyen de pairs trouvés | Nombre théorique de pairs existant (pour 4 millions) |
|---|---|---|
| 15 | 0.0196 | 122 |
| 16 | 0.1667 | 61 |
| 17 | 1.2647 | 30.5 |
| 18 | 3.2255 | 15.26 |
| 19 | 2.6274 | 7.63 |
| 20 | 1.4118 | 3.81 |
| 21 | 0.6078 | 1.9 |
| 22 | 0.4118 | 0.95 |
| 23 | 0.1274 | 0.48 |
| 24 | 0.0588 | 0.24 |
| 25 | 0.0392 | 0.12 |
| 26 | 0.0196 | 0.06 |
| 27 | 0.0098 | 0.03 |
| 28 | 0.0098 | 0.015 |

TABLE 8.2 – Nombre moyen de pairs parmi les 10 meilleurs contacts trouvés partageant un préfixe donné avec la référence

peut être retenu pour la demande du service. Cette contrainte permet de distribuer les pairs responsables d'une référence à l'échelle du réseau IP. Cela rend plus difficile la prise de contrôle d'une référence qui nécessite alors une architecture distribuée complexe à mettre en œuvre.

#### 8.1.3.2 Suppression des pairs trop proches

La seconde protection consiste à définir une distance minimum entre une référence et les pairs qui en sont responsables. Comme montré précédemment, les pairs légitimes choisissent aléatoirement leur identifiant et l'espérance de trouver un pair partageant un préfixe peut être établie, comme présenté par le tableau 7.4. Par conséquent, nous pouvons définir un seuil de sûreté établissant la proximité maximale entre un pair et une référence. Au delà de ce préfixe, il devient très peu probable que le pair ait choisi aléatoirement son identifiant, celui-ci est alors suspecté d'attaquer l'entrée de la DHT.

Sans cette protection, la seule contrainte s'appliquant à l'identifiant d'un pair est définie par la zone de tolérance, c'est à dire qu'un pair doit partager au moins 8 bits avec les références qu'il peut indexer. Autrement dit, tout pair partageant entre 8 bits et 128 bits avec une référence peut être sélectionné pour réaliser le service. Cependant, les pairs partageant un préfixe très élevé (entre 28 and 128 bits) sont suspicieux comme illustré par le graphique 8.5. Nous définissons donc une distance minimum de 28 bits entre un pair et les références qu'il peut indexer, ce qui réduit l'étendue de la zone de tolérance de [8 ;128] à [8 ;28]. Cette protection évite les attaques les plus évidentes et ce, sans introduire d'inconvénient. Elle n'est toutefois pas suffisante car plusieurs pairs partageant 28 bits avec une référence seront toujours en mesure d'attirer l'ensemble des requêtes.

## 8.2 Modèle de détection des attaques

### 8.2.1 Analyse des distributions par la divergence de Kullback-Leibler

Pour détecter les attaques, nous comparons la distribution des préfixes des dix meilleurs pairs à la distribution théorique précédemment établie. Leur sélection lors du service constitue en effet la base de toutes les attaques ciblées sur la DHT. La difficulté majeure vient du fait qu'un échantillon de dix pairs est très petit et rend difficile l'utilisation des outils statistiques classiques comparant les distributions. Nous avons tout d'abord essayé la méthode du chi-carré et celle de Kolmogorov-Smirnov pour comparer les distributions des identifiants mais celles-ci ne donnaient pas de résultats exploitables dans notre cas. Nous avons ensuite utilisé une métrique basée sur la divergence de Kullback-Leibler (également appelé G-test) donnant cette fois-ci une meilleure identification des distributions et permettant la détection des attaques. Cette méthode est connue pour être plus efficace pour comparer des distributions empiriques comportant un petit nombre d'observations [SR94]. Soient les distributions de probabilité M et T d'une variable aléatoire discrète, la divergence K-L entre M et T est définie comme suit :

$$D_{KL}(M \mid T) = \sum_i M(i) log \frac{M(i)}{T(i)} \tag{8.3}$$

Nous définissons une variable aléatoire discrète prenant 11 valeurs représentant la distribution possible des préfixes entre 18 bits et 28 bits. T représente la distribution théorique des préfixes suivant la loi géométrique de paramètre 1/2. M représente la distribution des préfixes des 10 meilleurs contacts mesurée à l'issue du processus de localisation. Le tableau 8.3 montre les valeurs définies pour T et deux exemples de distribution pour M, l'une après une localisation saine, l'autre

| Préfixe | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| M (attaque) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.5 | 0.5 | 0 |
| M (saine) | 0.6 | 0.2 | 0.1 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| T | 1/2 | $1/2^2$ | $1/2^3$ | $1/2^4$ | $1/2^5$ | $1/2^6$ | $1/2^7$ | $1/2^8$ | $1/2^9$ | $1/2^{10}$ | $1/2^{11}$ |

TABLE 8.3 – Exemples de distribution de préfixes comparée par la divergence K-L pour détecter les attaques

attaquée. Pour détecter les attaques, nous ne considérons pas les identifiants présentant moins de 18 bits communs avec la référence pour deux raisons. D'une part, comme le montre le tableau 8.2 les pairs partageant un préfixe de 17 bits avec la référence sont trop éloignés pour attirer efficacement les requêtes et n'en reçoivent qu'en moyenne 1,26 sur les 10 émises tout en subissant une compétition importante des autres pairs pour ce préfixe. D'autre part, comme le montre le graphique 8.5, la distribution moyenne actuellement observée dans KAD suit parfaitement la distribution géométrique (de paramètre 1/2) pour les préfixes d'au moins 18 bits. Cette borne peut évoluer vers des préfixes plus grands si le nombre de pairs dans la DHT augmente (le préfixe minimum est augmenté d'un bit à chaque fois que le nombre de pairs double), ou inversement, des préfixes plus petits s'il diminue. A propos de la borne supérieure de 28 bits, nous avons expliqué dans le paragraphe précédent pourquoi les contacts plus proches doivent être filtrés de manière préventive.

La divergence K-L n'est pas une mesure symétrique $D_{KL}(M \mid T) \neq D_{KL}(T \mid M)$ (ce n'est donc pas une distance). Dans notre cas, calculer la divergence de M par rapport à T est plus approprié pour détecter les attaques. Ainsi, pour chaque préfixe entre 18 et 28 bits pour lequel un pair est trouvé ($M(i) \neq 0$), la distribution mesurée pour ce préfixe est comparée à la distribution théorique puis sommée aux résultats des autres préfixes afin d'obtenir la divergence K-L globale pour l'échantillon. Les préfixes les plus élevés (avec une faible valeur de T correspondante) et les contacts groupés sur un même préfixe (engendrant une valeur de M élevée) augmentent fortement la divergence globale ce qui est pertinent pour détecter les attaques. La divergence K-L présente l'avantage d'être utilisable avec de très petits échantillons mais également de se calculer de manière incrémentale, ce qui permet de trouver quel préfixe augmente le plus la divergence et ainsi d'y appliquer précisément des contre-mesures en cas d'attaque. Cependant, la divergence K-L ne donne pas une probabilité de similarité entre deux distributions mais un indice relatif. Celui-ci doit donc être étudié afin de définir quelles valeurs indiquent une attaque.

### 8.2.2 Évaluation de la méthode de détection

Afin de définir le seuil de détection, nous calculons la divergence K-L pour deux ensembles de distributions, l'un représentant une distribution saine et l'autre en cas d'attaque. Le premier ensemble de distributions est basé sur les nombreux résultats des requêtes de localisation pour des références aléatoires, comme présenté dans la section précédente. Le second ensemble de distributions simule des positionnements suspects de pairs observables en cas d'attaque.

#### 8.2.2.1 Simulation des distributions d'attaque

Une attaque ciblée peut être mise en œuvre suivant différents paramètres. Les deux principaux devant être définis par un attaquant sont le nombre de pairs malveillants à insérer autour de la cible, ainsi que le préfixe qu'ils partagent avec celle-ci et qui définit leur proximité. Ainsi, en

| nombre de pairs malveillants insérés | nombre de préfixes corrompus | répartition des pairs | nombre d'attaques générées |
|---|---|---|---|
| 10 | 1 | 10 | 11 |
| 10 | 2 | 7-3 | 9 |
| 10 | 2 | 5-5 | 9 |
| 10 | 3 | 5-3-2 | 8 |
| 10 | 4 | 4-3-2-1 | 7 |
| 10 | 5 | 4-2-2-1-1 | 6 |
| 10 | 6 | 2-2-2-2-1-1 | 5 |
| 10 | 7 | 2-2-2-1-1-1-1 | 4 |
| 10 | 10 | 1-1-1-1-1-...-1 | 2 |
| 5 | 1 | 5 | 11 |
| 5 | 3 | 2-2-1 | 8 |
| 5 | 5 | 1-1-1-1-1 | 6 |

TABLE 8.4 – Répartition possible des pairs ayant un préfixe commun compris entre 18 et 28 bits

fonction de ces paramètres, plusieurs distributions de préfixes peuvent traduire une attaque selon le nombre de pairs insérés et leur distance à la cible.

Comme mentionné, notre mécanisme de détection considère la distribution des préfixes des 10 meilleurs pairs lorsqu'ils sont compris entre 18 bits et 28 bits. Pour simuler les distributions d'attaque, nous avons d'abord considéré, pour chaque longueur de préfixe, un nombre de pairs équivalent au nombre moyen observé lors de nos observations aléatoires. Nous insérons ensuite les attaquants dans cette distribution typique en suivant diverses stratégies de répartition possibles impliquant l'insertion de 5 et 10 Sybils sur les préfixes observés. Le tableau 8.4 présente les différentes répartitions testées allant de la plus évidente, à savoir 10 pairs insérés sur un unique préfixe (une attaque étant générée pour chacun des préfixes ciblés possibles entre 18 et 28 bits), à la moins évidente, à savoir 5 pairs insérés et répartis sur 5 préfixes. Enfin, nous calculons la distribution finale des préfixes pour les 10 contacts les plus proches en confondant alors les pairs légitimes et les attaquants.

Les différentes configurations d'attaque simulées, et la distribution des pairs associée, sont plus ou moins efficaces et faciles à détecter. Ainsi, 10 pairs malveillants insérés pour partager 28 bits avec la cible pourront attirer facilement les requêtes mais devraient également dépasser largement notre seuil de détection.

Pour évaluer notre mécanisme de détection, nous ne simulons pas d'attaques massives impliquant plus de 10 pairs malveillants car seule la distribution des 10 meilleurs pairs trouvés est considérée lors de la détection. Ainsi une attaque reposant sur 15 pairs répartis sur 3 préfixes tels que $[5 * 20; 5 * 21; 5 * 22]$ n'apparaît qu'à travers les 10 meilleurs pairs au mécanisme, soit $[5 * 21; 5 * 22]$. Néanmoins, les attaques impliquant davantage de pairs sont bien considérées par la contre-mesure présentée dans la section suivante.

#### 8.2.2.2 Résultats

Afin de définir le seuil de détection le plus approprié à notre cas d'étude, nous calculons les divergences KL obtenues pour les deux ensembles de distribution, définissons un seuil, et évaluons la précision de la détection. La divergence KL appliquée à l'ensemble des distributions saines (obtenue par localisation d'identifiants aléatoires) donne une moyenne de 0.41 avec un

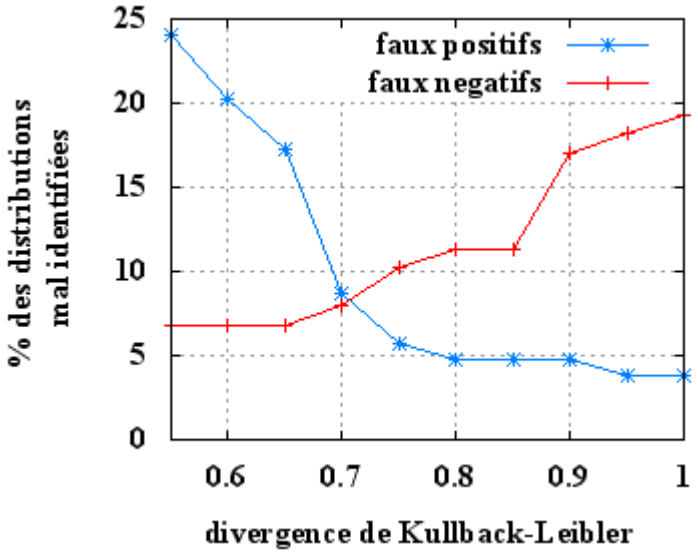


FIGURE 8.6 – Taux de faux positifs et de faux négatifs selon le seuil de détection (divergence K-L)

écart type de 0.24. Nous considérons une valeur de 0.55 (moyenne + écart type) comme le premier seuil de détection testé. Nous augmentons ensuite progressivement ce seuil (c'est à dire la divergence autorisée) et mesurons le nombre de distributions mal identifiées. Les faux négatifs sont les distributions issues des attaques simulées non détectées comme telles et les faux positifs sont des distributions saines considérées comme des attaques. Le graphique 8.6 montre qu'une divergence de 0.7 constitue un bon compromis pour détecter la plupart des attaques avec un faible taux de faux négatifs (7.95%) tout en limitant les faux positifs (8.65%). En regardant plus précisément les distributions mal classées, on peut observer que les attaques les plus dangereuses impliquant l'insertion de dix pairs sont détectées avec un taux de faux négatifs extrêmement faible de 1.56%.

Les attaques les plus difficiles à détecter sont celles impliquant 5 pairs. Elles présentent un taux de faux négatifs de 21.73% car le nombre limité de pairs insérés réduit l'impact de l'attaque sur la distribution des identifiants. Cependant, de telles attaques impliquant un nombre de pairs inférieur au facteur de réplication ne peuvent réussir à contrôler entièrement une référence. Notre mécanisme parvient à détecter les configurations les plus efficaces. Une analyse détaillée montre que nous manquons principalement deux types de distributions relativement inoffensives :

- Les attaques ciblant uniquement le préfixe de 18 bits. De telles attaques sont inefficaces car trop éloignées de la cible. En effet, de nombreux pairs légitimes se trouvent plus proches (19 bits communs ou plus) et recevront les requêtes avant les attaquants alors que ceux-ci sont en même temps en compétition avec de nombreux pairs légitimes sur le préfixe de 18 bits pour attirer les requêtes.
- Les attaques impliquant 5 pairs distribués sur plus de 3 préfixes inférieurs à 23 bits. Ces attaques distribuées sur plusieurs préfixes sont difficiles car elles doivent compter sur l'absence de pairs légitimes sur les préfixes élevés pour ne pas être détectées tout en étant en compétition avec les pairs légitimes sur les préfixes plus petits.

En résumé, les attaques les plus efficaces nécessitant l'insertion de nombreux pairs sont com-

plètement détectées alors que les attaques manquées insèrent trop peu de pairs, ou insèrent les pairs trop éloignés de la cible, pour constituer un véritable danger. Plus une attaque est efficace, mieux celle-ci est détectée. Maintenant que nous pouvons détecter les attaques localisées en analysant la distribution des identifiants trouvés, la section suivante présente comment protéger la réalisation du service des pairs malveillants en cas d'attaque.

## 8.3 Protection contre les attaques

### 8.3.1 Contre-mesure supprimant les pairs suspects

**Algorithme 4** : Algoritme de contre-mesure contre les attaques ciblées

```
   Input : contact_list [ ] ; prefixes_distribution [ ] ; KL_increments [ ] ; KL_div ;
           max_div ;
   Output : updated contact_list [ ]
 1 foreach prefix in prefixes_ distribution do
 2 |   KL_increments.add(partial_KL_div(prefix));
 3 end
 4 KL_div = SUM(KL_increments);
 5 while KL_div > max_div AND MAX(KL_increments) > 0 do
 6 |   prefix=KL_increments.index(MAX(KL_increments));
 7 |   remove_contacts(contact_list, prefix);
 8 |   remove_distance(KL_increments, prefix);
 9 |   KL_div=SUM(KL_increments);
10 end
```

Notre contre-mesure supprime progressivement les pairs dont le préfixe contribue le plus à l'augmentation de la divergence. Cette procédure est décrite par l'algorithme 4. Elle est appliquée lorsqu'une distribution est détectée en tant qu'attaque. Elle vise à fournir une liste de contacts potentiels saine pour la réalisation du service. La contre-mesure partage des similarités avec la procédure de détection : elle manipule également la liste de contacts trouvés à l'issue de la phase de localisation (elle est donc indépendante du type de service exécuté) et compare les mêmes distributions de préfixes compris entre 18 bits et 28 bits grâce à la divergence KL. Cependant, contrairement à la détection, il ne suffit pas de considérer seulement les 10 meilleurs contacts pour éviter tous les pairs malveillants potentiels. La contre-mesure est donc appliquée de manière itérative, re-vérifiant la distribution des 10 meilleurs contacts à chaque fois que certains pairs suspicieux sont supprimés de la liste, jusqu'à ce que celle-ci soit considérée comme sûre : soit car la divergence globale après correction est inférieure à un seuil, soit car aucun préfixe ne contribue à la divergence. Si un attaquant insère de nombreux pairs autour de la cible, ils sont ainsi progressivement supprimés de la liste des 10 meilleurs contacts à chaque itération, jusqu'à ce que la divergence soit sûre, ou qu'aucun préfixe ne reste avec un nombre de pairs trouvés supérieur à la valeur théorique.

Si besoin, d'autres contacts trouvés sur le chemin vers la cible et présentant un préfixe plus petit avec celle-ci (17 bits ou moins) peuvent être sélectionnés pour combler les places libres parmi la liste après application de la contre-mesure, sans nécessiter de nouvelle procédure de localisation. Cette approche progressive a l'avantage d'éviter les pairs malveillants tout en préservant les contacts les moins suspicieux parmi les meilleurs trouvés. De cette manière, la qualité des contacts

| Préfixe | Nombre moyen de contacts |
|---|---|
| 13 | 0.60 |
| 14 | 1.36 |
| 15 | 2.78 |
| 16 | 3.62 |
| 17 | 3.75 |

TABLE 8.5 – Meilleurs contacts restant trouvés avec un préfixe inférieur à 18bits

restant à l'issue de la contre-mesure est peu affectée en cas de faux-positifs.

La contre-mesure n'induit donc pas de surcoût en terme de paquets échangés, agissant localement sur les contacts trouvés par le processus de localisation. Par ailleurs, le coût calculatoire de la contre-mesure est négligeable avec une complexité en O(1). Une itération est nécessaire pour les préfixes attaqués entre entre 18 et 28 bits, chaque itération étant limitée à parcourir deux tableaux de taille fixe.

### 8.3.2 Évaluation de la contre-mesure par simulation d'attaques

Afin d'évaluer notre contre-mesure, nous avons utilisé les distributions ayant servi à l'évaluation de la phase de détection. Nous comptons ici le nombre moyen de contacts supprimés par la contre-mesure lorsqu'une distribution est détectée comme attaque tout en faisant varier la divergence maximum autorisée, c'est à dire une des deux conditions d'arrêt. Nous avons défini un seuil de 0.7 (égale au seuil de détection) comme la valeur de divergence la plus laxiste à considérer. Le graphique 8.7 montre que la contre-mesure adapte précisément le nombre de contacts supprimés au nombre de pairs insérés. Ainsi, le nombre de contacts supprimés correspond aux différents cas d'attaques. En considérant tout d'abord les cas de faux positifs issus des distributions saines considérées comme des attaques (8.65% des cas), seuls deux à trois contacts parmi celles-ci sont finalement supprimés par la contre-mesure ce qui limite l'impact négatif en cas de faux positifs, les résultats de la localisation demeurant peu affectés. Au contraire, les distributions simulées d'attaques insérant 10 pairs sont fortement impactées par la contre-mesure avec entre 8 et 10 pairs malveillants supprimés. Notre protection parvient donc efficacement à éviter les attaques les plus dangereuses. Enfin, les stratégies insérant 5 pairs sont également largement affaiblies avec en moyenne entre 4 et 5 pairs malveillants supprimés, ce qui correspond à l'attaque. La contre-mesure affecte donc peu les faux positifs alors que les pairs malveillants issus d'attaques sont tous supprimés.

Tout comme pour le seuil de détection, la divergence maximum autorisée après application de la contre-mesure peut être adaptée afin de trouver un compromis entre l'efficacité de la contre-mesure et l'impact en cas de faux positifs. D'après la figure 8.7, nous considérons qu'une divergence maximum établie entre 0 et 0.3 est pertinente pour éviter au mieux les attaques tout en supprimant peu de contacts légitimes. Malgré la perte de contacts légitimes sur les préfixes attaqués, l'information faisant l'objet du service restera sur des pairs proches de son identifiant. En effet, le processus de localisation découvre toujours davantage de pairs que les 10 meilleurs utilisés au final. D'après nos mesures de publications saines, le tableau 8.5 montre que les contacts restants ont, en moyenne, un préfixe compris entre 17 bits et 15 bits avec la cible. Ces valeurs étant bien supérieures à la distance minimum (zone de tolérance) définie à 8 bits par le protocole, ces pairs sont utilisés pour remplacer les pairs légitimes supprimés par la contre-mesure.

Finalement, l'intégralité de notre solution de défense contre les attaques ciblées sur la DHT

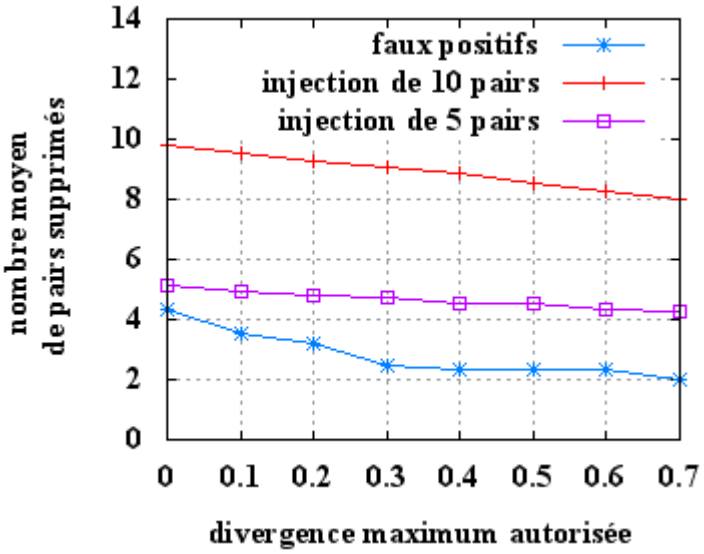


FIGURE 8.7 – Nombre moyen de contacts supprimés par la contre-mesure en fonction de la divergence autorisée

est présentée par le schéma 8.8. Celle-ci pourrait être implantée dans les prochaines versions des clients KAD, les protégeant contre les attaques de la DHT tout en préservant la rétro-compatibilité avec les anciens clients, ce qui assure la durabilité du réseau. Plus généralement, cette solution peut être appliquée à d'autres DHT, et ce, d'autant plus efficacement que celles-ci partagent les caractéristiques de KAD à savoir : un mécanisme de localisation itératif (sans délégation à des pairs intermédiaires), un découplage entre les phases de localisation et de service (évitant ainsi un surcoût de message en cas d'attaque) et la réplication de l'information sur plusieurs pairs (qui est une solution courante pour lutter contre le va-et-vient des pairs).

## 8.4 Mise en œuvre et améliorations

La validité des protections proposées a été éprouvée par deux implantations dans des clients opérant deux DHT différentes (basées sur Kademlia), à savoir aMule (KAD) et gtk-gnutella (Gnutella DHT). Comme attendu, les implantations furent relativement directes et n'ont requis que quelques modifications dans le code des clients au niveau de la fonction de localisation des pairs qui précède tout service (*Search.cpp* pour KAD, *lookup.c* pour gtk-gnutella). En revanche, la configuration de plusieurs paramètres nécessaires afin d'adapter la détection au réseau ciblé s'est avérée plus délicate.

Après avoir présenté les deux implantations, nous décrivons comment les paramètres de notre solution peuvent être adaptés dynamiquement pour protéger au mieux les différents réseaux P2P. Nous terminons avec le test d'un client protégé contre une attaque réelle.

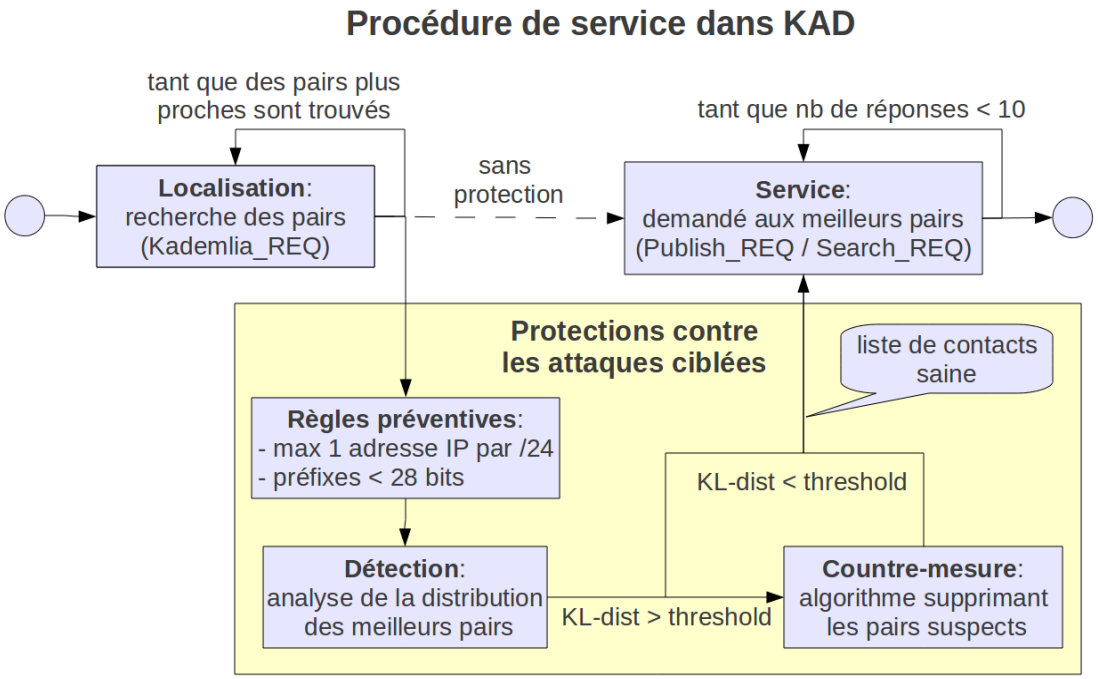


FIGURE 8.8 – Schéma complet de notre protection appliquée à KAD

| Préfixe | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| T (simulation) | 0 | 1/2 | $1/2^2$ | $1/2^3$ | $1/2^4$ | $1/2^5$ | $1/2^6$ | $1/2^7$ | $1/2^8$ | $1/2^9$ | $1/2^{10}$ | $1/2^{11}$ |
| T (corrigée) | 2/10 | 3/10 | $1/2^2$ | $1/2^3$ | $1/2^4$ | $1/2^5$ | $1/2^6$ | $1/2^7$ | $1/2^8$ | $1/2^9$ | $1/2^{10}$ | 0 |

TABLE 8.6 – Préfixes considérés après correction

### 8.4.1 Implantations

#### 8.4.1.1 KAD

KAD étant le réseau P2P utilisé pour concevoir notre solution, notamment en se basant des mesures réelles, l'implantation des protections a été directe. Cependant, les exécutions ont montré un décalage dans la distribution des préfixes des 10 meilleurs contacts constatée dans les précédentes mesures et celles obtenues par la protection implantée. Plus précisément, considérer les préfixes à partir de 18 bits ne permet pas d'obtenir suffisamment de contacts. Pour cela, les pairs partageant 17 bits avec la cible doivent également être considérés sans toutefois que ceux ci ne valident parfaitement la distribution théorique (loi hypergéométrique de paramètre (1/2). En effet, même si dans l'absolu le nombre de pairs dans la DHT partageant 17 bits et le double de celui partageant 18 bits, tous ne sont pas trouvés lors de la localisation.

La distribution théorique a donc due être adaptée d'après ce constat ce qui donne la nouvelle distribution présentée dans le tableau 8.6. Ceci peut s'expliquer par le fait que le nombre pair a varié entre les quelques mois séparant la conception et l'implantation des protections. Cependant, ce seul ajustement n'est pas suffisant pour appliquer notre méthode à la DHT de Gnutella.

```
11bit prefix: freq=0.500 (10/20 nodes,log2=-1.00000) theoric=0.500000 => K-L contrib: 0.000000
12bit prefix: freq=0.200 (4/20 nodes,log2=-2.321928) theoric=0.250000 => K-L contrib:-0.064386
13bit prefix: freq=0.150 (3/20 nodes,log2=-2.736966) theoric=0.125000 => K-L contrib: 0.039455
14bit prefix: freq=0.050 (1/20 node, log2=-4.321928) theoric=0.062500 => K-L contrib:-0.016096
15bit prefix: freq=0.050 (1/20 node, log2=-4.321928) theoric=0.031250 => K-L contrib: 0.033904
16bit prefix: freq=0.050 (1/20 node, log2=-4.321928) theoric=0.015625 => K-L contrib: 0.083904
with 20/20 nodes, K-L divergence to bce4d59bd8db868b7ffc0031ae81cca8db51937f = 0.076780
```

FIGURE 8.9 – Exécution de la détection sur une localisation de gtk-gnutella

#### 8.4.1.2 gtk-gnutella

L'implantation de notre solution dans le client gtk-gnutella a été réalisée par l'auteur de ce client, Raphaël Manfredi dont une trace d'exécution est présentée par la figure 8.9. La DHT de gnutella comporte significativement moins de pairs que celle de KAD (environ $7 * 10^4$ contre $2 * 10^6$). Cela implique un décalage important des préfixes considérés par la détection. Par ailleurs, la DHT de Gnutella a perdu brusquement un grand nombre d'utilisateurs fin 2010 lors de l'arrêt de son principalement client LimeWire. Celle-ci était composée principalement de nœuds LimeWire et de nœuds gtk-gnutella. Suite à l'arrêt du client LimeWire, la taille du réseau est passée d'environ 500000 nœuds à 70000 nœuds.

Ce fait nous indique que la population d'une DHT peut rapidement varier et ce dans de grandes proportions et nous met en garde contre l'utilisation de paramètres statiques pour la détection. En effet, une telle perte de population avec des paramètres de détection fixes aurait pour conséquence une moindre efficacité de la détection. A l'inverse, une augmentation soudaine de la population serait plus dommageable car beaucoup de pairs légitimes apparaîtraient comme suspicieux (trop proches d'une ressource) ce qui aurait pour conséquence de déclencher la contre-mesure pour tous service demandé et rendrait le réseau moins performant. Ceci nous enseigne que les paramètres de détection fixes ne sont pas fiables à long terme et doivent donc être calculés dynamiquement pour protéger au mieux le réseau.

### 8.4.2 Estimation dynamique des paramètres

Deux approches sont possibles afin de calibrer dynamiquement la détection pour la taille courante du réseau : soit en estimant préalablement le nombre de pairs duquel on dérive les paramètres de détection, soit en apprenant directement la distribution saine des préfixes. Dans les deux cas, le paramétrage se fait en émettant des recherches vers des identifiants aléatoires sur la DHT.

#### 8.4.2.1 Estimation du nombre de pairs

La distribution de référence utilisée pour détecter les pairs suspicieux dépend directement du nombre de pairs participants à la DHT. Le nombre de pair global est un paramètre qui est habituellement estimé par les clients d'une DHT. Cette fonctionnalité est ainsi présente dans KAD, Mainline DHT et gnutella-dht, entre autres. La connaissance du nombre global de pair permet un paramétrage direct et mieux adapté de notre protection.

Soit N le nombre de pairs dans la DHT estimé comme décrit précédemment, et K le facteur de réplication du réseau P2P (constante d'implantation). Le début de la fenêtre de détection est

donné par le nombre de bits "bmin" tel que :

$$bmin = E[log2(N \div K)] \tag{8.4}$$

E[X] étant la partie entière du résultat, le nombre de bit étant forcément un entier. L'application numérique à KAD et gtk-gnutella donne les résultats suivants :

- pour KAD : $bmin = E[log2(2 * 10^6 \div 10)] = 17$
- pour gtk-gnutella : $bmin = E[log2(7 * 10^4 \div 20)] = 11$

L'espérance de trouver un contact avec le préfixe "bmin" parmi les K meilleurs à l'issue de la localisation est d'environ 1/2, chaque bit supplémentaire divise cette espérance par 2. La limite supérieure "bmax" de la fenêtre doit être définie de telle sorte que les pairs au delà sont considérés trop proches et automatiquement filtrés. Nous avons choisi $bmax = bmin + \delta$ avec $\delta = 10$, ce qui limite le nombre de pairs légitimes filtrés à $1/2^{11}$, soit moins de 0.05%.

Cependant, dans le cas ou l'estimation du nombre de pairs total n'est pas fournie par le client, la distribution saine des préfixes peut facilement être apprise du réseau.

#### 8.4.2.2 Apprentissage de la distribution

Une seconde approche consiste à apprendre directement les valeurs de la distribution de référence en réalisant des recherches aléatoires sur la DHT, sans estimer au préalable le nombre de pairs y participant. Afin d'éviter les attaques potentielles, plusieurs recherches vers des identifiants aléatoires sont émises, dont on enregistre alors les 10 meilleurs pairs. L'ensemble des pairs obtenus permettent de connaître la distribution normale des préfixes pour l'état courant de la DHT. L'algorithme 5 décrit cette méthode d'apprentissage.

**Algorithme 5** : Algorithme d'estimation de la distribution des préfixes par apprentissage

```
   Input : nb_lookup
   Output : prefixes_distribution [ ]
 1 nb_contacts=0;
 2 for i=0 ; i<nb_ lookup ; i++ do
 3 |   rdm_id = rdm_id.random();
 4 |   10_best_contacts [ ] = lookup(rdm_id);
 5 |   foreach contact in 10_ best_ contacts do
 6 |   |   current_prefix = compute_prefix(rdm_id, contact);
 7 |   |   prefixes_distribution [current_prefix] ++;
 8 |   |   nb_contacts ++;
 9 |   end
10 end
11 foreach prefix in prefixes_ distribution do
12 |   prefix = prefix / nb_contacts;
13 end
```

L'estimation du nombre de pairs et l'apprentissage de la distribution peuvent être réactualisés périodiquement (par exemple toutes les 2 heures) afin d'adapter la protection aux fluctuations

journalières du nombre de pairs. Si KAD est relativement épargné par ce phénomène car ses utilisateurs sont équitablement répartis entre l'Europe et l'Asie, d'autres DHT voient quant à elles leur nombre de pairs connectés varier du simple au double selon la période de la journée. La figure 8.10 a été réalisée en enregistrant le nombre de pairs estimés pendant 4 jours dans deux DHT [TCCF11b] et illustre bien ce phénomène pour KAD et la DHT de Bittorent.

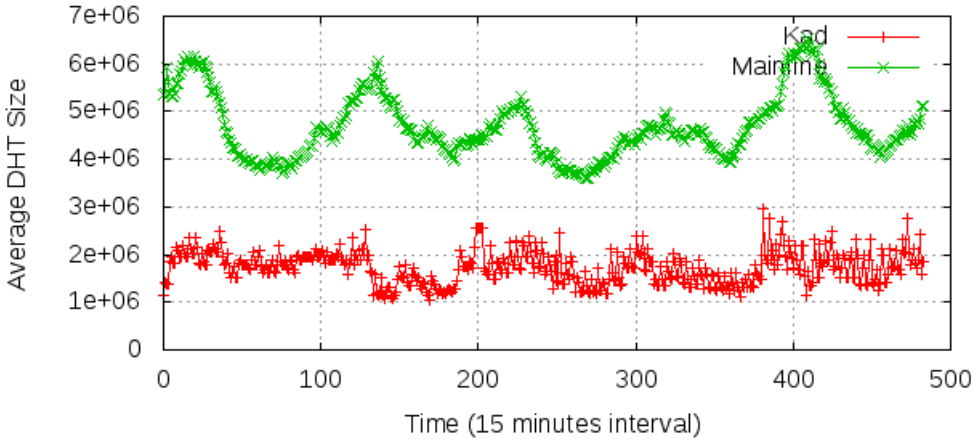


FIGURE 8.10 – Évolution de la population de deux DHT

### 8.4.3 Expérimentation contre une attaque réelle

Afin valider en pratique notre proposition de détection et de défense, nous avons déployé une attaque ciblée sur un mot-clé, impliquant 5 Sybils ayant au moins 20 bits en commun avec la cible. La figure 8.11 ci-dessous présente les résultats obtenus pour la phase de détection à l'issue d'une localisation pour le mot-clé ciblé « madonna ».

```
Detection results:
Name: madonna
Hash: A35BC8A4D252ADB3A99A46A28B275DFB

Responded: 26 21 21 21 20 20 19 18 18 17 16 16 11
Prefix between 17 and 28 from the 10 best distributions: 10 (13 total)
Ratio (responded[17-28]/responded): 0,77

KL distance: 1.093836, details:
 KL increments  -0,13 -0,13 -0,02 0,23 0,68 0,00 0,00 0,00 0,00 0,46 0,00 0,00
     Prefixes     17   18   19   20   21   22   23   24   25   26   27   28
       M        0,10 0,20 0,10 0,20 0,30 0,00 0,00 0,00 0,00 0,10 0,00 0,00
       T        0,38 0,38 0,12 0,06 0,03 0,02 0,01 0,00 0,00 0,00 0,00 0,00
```

FIGURE 8.11 – Détection d'une attaque sur le mot clé « madonna »

L'encadré bleu présente les préfixes des pairs ayant répondu aux requêtes de localisation. Ils sont au nombre de 13 et les 10 premiers constituent la distribution observée $M$. L'encadré rouge montre quant à lui la distance de Kullback-Leibler associée à cette distribution $M$. On

remarque que celle-ci est supérieure au seuil de 0,7. L'attaque est donc détectée ce qui déclenche la contre-mesure (figure 8.12).

```
Countermeasure results:
Name: madonna
Hash: A35BC8A4D252ADB3A99A46A28B275DFB

Responded: 20 20 19 18 18 17 16 16 11
Prefix between 17 and 28 from the 10 best distributions: 6 (9 total)
Ratio (responded[17-28]/responded): 0,67

KL distance: 0,431523, details:
 KL increments  -0,14 -0,04 0,05 0,56 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00
     Prefixes     17   18   19   20   21   22   23   24   25   26   27   28
       M        0,17 0,33 0,17 0,33 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00
       T        0,38 0,38 0,12 0,06 0,03 0,02 0,01 0,00 0,00 0,00 0,00 0,00
```

FIGURE 8.12 – Contre-mesure après la détection d'une attaque sur le mot clé "madonna"

On constate que l'application de la contre-mesure supprime bien les préfixes les plus dangereux. En effet, 4 pairs parmi les 5 Sybils injectés ( sur les préfixes 26 et 21) ont été successivement retirés du tableau des pairs retenus. Le seuil d'arrêt de la contre-mesure était défini à 0.7 pour cette expérience. Le contact ayant un préfixe de 26 a d'abord été retiré du tableau, mais la distance de Kullback-Leibler restant supérieure au seuil, les pairs du préfixe 21 ont été supprimés à leur tour ce qui donne la distribution finale ci-dessus, la distance ($D_{KL} = 0,431523$) étant alors inférieure à la condition d'arrêt (0.7). Une condition d'arrêt plus stricte (par exemple 0) aurait également supprimé le Sybil à 20 bit, ce préfixe contribuant à l'augmentation de la distance (0.56), mais également le contact légitime se trouvant sur ce préfixe. Nous préconisons cependant une condition d'arrêt plus stricte afin de supprimer le maximum de pairs suspects.

## Conclusion

Nous avons proposé dans ce chapitre une solution protégeant une DHT contre les attaques localisées pouvant amener à un contrôle des informations ciblées par une entité malveillante. Souhaitant créer une solution efficace et applicable à un réseau déjà déployé, celle-ci devait détecter et éviter efficacement les attaques, à un faible coût, tout en préservant la rétro-compatibilité entre clients. Deux règles préventives ont d'abord été définies, contraignant les identifiants et les adresses IP des pairs responsables d'une information visant à rendre une attaque locale plus coûteuse à réaliser mais demeurent insuffisantes pour complètement l'éviter.

Nous avons développé pour cela une méthode de détection locale à chaque pair, analysant la distribution des préfixes des pairs trouvés à l'issue du processus de localisation. En effet, nous avons observé et vérifié que, d'une part la distribution des préfixes des dix meilleurs pairs trouvés autour d'une référence non attaquée suit toujours une loi parfaitement caractérisable (géométrique de paramètre 1/2) et d'autre part, que toute attaque ciblée fait dévier la distribution des préfixes observés de cette loi, et ce, d'autant plus que l'attaque est efficace. En utilisant la divergence de Kullback-Leibler, nous pouvons ainsi détecter les attaques en comparant une distribution suspecte à la distribution de référence et étant donné un seuil de détection défini lors de simulations. Quand une attaque est détectée, une contre-mesure permet de supprimer

progressivement les pairs suspicieux des contacts trouvés afin de fournir un accès sécurisé à la DHT.

Dans un premier temps, les différentes évaluations réalisées par simulation d'attaques ont montré que notre méthode détecte les attaques avec un faible taux d'erreurs (faux-positifs et faux-négatifs). notamment pour les attaques les plus efficaces (1,56%), alors que la contre-mesure supprime ensuite la quasi-totalité des pairs malveillants impliqués, sécurisant ainsi l'accès à l'information. Les attaques non détectées insèrent trop peu de pairs, ou en concurrence avec des pairs légitimes et sont par conséquent moins dangereuses. Enfin, deux implantations dans le cadre de DHT réelles, à savoir KAD et le client gtk-gnutella, ont montré que notre méthode pouvait être facilement mise en œuvre et nécessite très peu de modifications dans le code existant de DHTs basées sur Kademlia. En outre, dans le cadre de Kademlia, notre protection ne nécessite l'échange d'aucun message et requiert une charge calculatoire négligeable. Dans la continuité de ces travaux, nous avons montré dans l'article [TCCF11a] que la principale DHT de BitTorrent (i.e. Mainline DHT, basée également sur kademlia) est également vulnérable aux attaques ciblées et l'applicabilité de notre solution dans ce contexte.

# Conclusion générale

Les réseaux pair à pair, notamment ceux utilisant les tables de hachage distribuées, sont aujourd'hui des systèmes d'information majeurs comptant des dizaines de millions d'utilisateurs et l'un des principaux usages d'Internet. Ils souffrent cependant de plusieurs problèmes de sécurité affectant les contenus stockés. D'une part, les DHTs présentent des vulnérabilités permettant l'insertion de nœuds pouvant réaliser plusieurs actions malveillantes (pollution, surveillance, suppression d'information, déni de service, ...). D'autre part, ces réseaux sont utilisés pour diffuser des contenus illégaux (contenus pédophiles, virus, ...) et hautement indésirables. Ces deux problèmes de sécurité sont intimement liés car l'attaque du mécanisme d'indexation peut servir de vecteur à la propagation de tels contenus qui peuvent alors être téléchargés inconsciemment par les utilisateurs et porter atteinte à leur sécurité. Dans le cadre de cette thèse, nous avons proposé et appliqué à KAD de nouvelles nouvelles méthodes de supervision permettant d'appréhender les problèmes de sécurité affectant les contenus.

A ce titre, nous avons réalisé trois principales contributions. Nous avons tout d'abord montré qu'une supervision précise et rigoureuse des accès aux contenus malveillants pouvait être réalisée au sein d'un grand réseau P2P et ce malgré les problèmes affectant son mécanisme d'indexation. Nous avons conçu pour cela une **architecture de supervision** capable d'observer l'ensemble des requêtes émises par un pair pour un contenu donné. Nous avons ensuite proposé des **méthodes de détection** des attaques affectant la DHT de KAD et ainsi mis en évidence la pollution par falsification d'indexation et de nombreuses insertions ciblées de nœuds. Nous avons en outre réalisé une quantification de ces attaques. Enfin, nous avons proposé une **protection contre les attaques ciblées** basée sur l'analyse de la distribution des identifiants des pairs et montré son efficacité. La solution proposée est de surcroit applicable à des DHT déjà déployées.

## 1 Travail réalisé

### Résumé

Après avoir motivé notre intérêt pour l'étude des réseaux P2P structurés, nous avons tout d'abord constaté à travers l'état de l'art que les méthodes de supervision permettant d'appréhender les accès aux contenus des réseaux P2P souffrent de plusieurs limites. Les pots de miel permettent de superviser les accès à certains fichiers mais leur vision du réseau est cependant très partielle (méconnaissance des autres sources et des fichiers connexes) et ne peut être augmentée facilement. Les sondes distribuées quant à elles permettent une connaissance exhaustive des contenus indexés mais sans pouvoir constater leurs transferts. Par ailleurs, aucune des méthodes de supervision ne considère la pollution du mécanisme d'indexation pouvant générer de nombreux faux positifs lors de la supervision des contenus, lorsqu'un utilisateur accède à un contenu partagé sous plusieurs noms trompeurs.

Pour pallier ces faiblesses, nous avons conçu et développé une architecture de supervision

performante et capable, pour un ensemble de mots-clés donné, de capturer la succession des requêtes émises par les pairs depuis la recherche des mots-clés jusqu'au téléchargement final de fichiers sur les pots de miel. Elle permet de superviser les contenus à plusieurs niveaux d'information dont la connaissance est nécessaire pour s'assurer de l'intention d'un pair et ainsi éviter les faux-positifs dus à la pollution. La conception de l'architecture a tout d'abord nécessité l'étude des vulnérabilités de KAD encore présentes malgré l'ajout de mécanismes de protection contre l'attaque Sybil dans les clients, puis, l'élaboration d'une nouvelle approche de contrôle du mécanisme d'indexation basée sur l'insertion de sondes distribuées et leur collaboration lors du processus de localisation des ressources. Nous avons en effet montré que l'efficacité du processus de localisation de KAD associé à la liberté de placement des sondes à proximité des ressources permet d'en prendre le contrôle sans nécessiter de corrompre directement les tables de routage des pairs. La modélisation de ce processus et des évaluations issues d'expériences réelles ont montré que 20 sondes sont ainsi suffisantes pour contrôler une référence. Nous avons en outre montré l'intérêt de ce contrôle en augmentant artificiellement le nombre de sources annoncées pour les fichiers appâts, ce qui a pour conséquence l'augmentation significative de l'efficacité des pots de miel. L'architecture HAMACK a finalement été entièrement implantée et déployée à grande échelle grâce à la plateforme PlanetLab. Nous avons ainsi supervisé de manière passive l'activité de contenus pédophiles sur KAD à travers 72 mots clés pendant deux semaines. L'analyse des données récoltées a permis une première caractérisation des activités pédophiles telles que l'attrait des âges ou leur répartition géographique.

Nous nous sommes ensuite intéressés à la sécurité des informations stockées dans la DHT. Les réseaux P2P tels que KAD sont en effet victimes de nombreuses attaques affectant la sécurité des contenus partagés. A travers l'état de l'art, nous avons ainsi identifié deux problèmes de sécurité majeurs que sont la pollution et l'attaque Sybil. Nous avons constaté l'absence de travaux récents sur la pollution, les précédentes études ayant été réalisées il y a plusieurs années sur des réseaux aujourd'hui obsolètes. De même, malgré les nombreux travaux de supervision dont les réseaux P2P ont fait l'objet et les nombreuses études sur leurs vulnérabilités, aucune supervision de réseau P2P réel n'a été réalisée afin de constater s'ils sont réellement attaqués.

Nous avons tout d'abord mis en évidence une nouvelle forme de pollution du mécanisme d'indexation consistant à falsifier l'indexation des fichiers. Cette pollution consiste à référencer un même fichier par de nombreuses descriptions différentes. Elle est particulièrement néfaste puisqu'elle amène un utilisateur à télécharger un contenu indésirable et contribue potentiellement à la diffusion de contenus malveillants tels que des contenus pédophiles. Nous avons proposé une métrique basée sur l'analyse des dis-similarités entre les différents noms de fichiers associés à un même contenu permettant de détecter cette pollution. La métrique a été validée par des experts puis appliquée à de nombreux contenus populaires afin d'établir une quantification de la pollution. Nous avons ainsi montré qu'au moins 2/3 des contenus populaires de KAD sont pollués dont 41% par cette nouvelle forme de pollution. Nous avons ensuite considéré une autre vulnérabilité des DHT à savoir l'insertion de noeuds malveillants. Pour découvrir ces attaques, nous avons exploré précisément le réseau puis recherché les positions suspectes des pairs pouvant traduire une attaque. Deux méthodes de détection ont ainsi été proposées, l'une basée sur l'analyse des distances entre pairs pouvant traduire des attaques groupées et l'autre basée sur l'analyse des distances entre un pair et une référence connue a priori et potentiellement attaquée. A travers cette analyse, nous avons mis en évidence que plus de 2200 ressources étaient clairement ciblées durant notre période d'observation.

Ce constat a motivé la conception d'une nouvelle protection contre les attaques ciblées sur la DHT. En effet, bien que plusieurs propositions aient été faites pour limiter l'attaque Sybil, les

difficultés de mise en œuvre des solutions existantes font que cette menace est toujours d'actualité, notamment au travers des attaques ciblées. En particulier, l'ensemble des solutions proposées nécessite des contraintes fortes devant être implantées conjointement par l'ensemble des pairs ce qui empêche leur application à un réseau déjà déployé. La solution que nous proposons utilise la divergence de Kullback-Leibler pour comparer une distribution d'identifiants de pairs mesurée à une distribution théorique saine. Une contre-mesure permet d'éviter les pairs suspects avant la réalisation du service lorsque des placements déviants sont détectés. L'évaluation par simulation et l'implantation ont montré l'efficacité de notre approche. Ses principaux avantages étant d'être parfaitement compatible et facilement applicable aux DHT actuelles à un coût négligeable.

## 2 Perspectives de recherche

Les résultats obtenus durant cette thèse ouvrent plusieurs perspectives de recherche directes.

### Collecte et traitement des données

Notre architecture de supervision, HAMACK, peut naturellement être appliquée à l'étude d'autres contenus cibles diffusés au sein de KAD. Cependant, l'ampleur et la richesse des données collectées par notre architecture rend l'exploitation des données difficile comme nous avons pu le constater lors de son application à la caractérisation des activités pédophiles. De plus, parmi l'ensemble des fonctionnalités d'HAMACK, seule la supervision passive des mots-clés était alors utilisée. Ainsi, il semble indispensable d'appliquer à l'avenir des méthodes de fouilles de données afin de pouvoir exploiter au mieux les données collectées, notamment en corrélant les différents niveaux d'observation que nous pouvons obtenir du réseau.

De tels travaux sont d'ores et déjà en cours afin de caractériser plus précisément les activités pédophiles d'après les données récoltées durant notre précédent déploiement. Une première approche non supervisée pourrait ainsi mettre en évidence des règles d'association permettant d'apprendre de nouveaux mots-clés utilisés aussitôt pour améliorer l'étiquetage. Puis, une fois les classes bien établies, une approche supervisée permettrait de dégager des caractéristiques parmi toutes les combinaisons de paramètres possibles définissant les activités pédophiles par rapport aux normales pour les différents objets considérés (recherches, publications, pairs). Nous pensons que la supervision de grands réseaux complètement distribués et exécutant des services complexes gagne à être assistée par un traitement intelligent de l'information afin d'être plus efficace.

Une seconde perspective de recherche est la comparaison de nos observations de contenus pédophiles avec celles obtenues dans le cadre du projet européen MAPAP sur le réseau eDonkey. La supervision du réseau eDonkey et celle présentée ici pour KAD sont en effet complémentaires, la première supervisant l'ensemble des requêtes de recherche sur quelques serveurs (vision partielle des recherches sur l'ensemble des mots), la seconde ne supervisant que certains mots-clés mais pour l'ensemble du réseau KAD et tous les types de requêtes (vision complète sur quelques mots). Une première étape viserait à montrer les similarités entre les données avant d'en extrapoler de nouveaux résultats. Ceci semble d'autant plus pertinent que KAD et eDonkey partagent les mêmes clients (eMule) et donc probablement les mêmes activités.

### Caractérisation des pairs malveillants

En nous intéressant à la sécurité des réseaux P2P, et de KAD en particulier, nous avons montré que de nombreuses attaques sont réellement lancées sur ces réseaux. Nous avons en outre proposé

des moyens de détection pour la pollution par falsification d'indexation et pour les attaques ciblées. En reprenant le cœur de notre protection contre les attaques ciblées, nous pourrions proposer une détection plus fine des placements suspects de pairs basée sur la divergence de Kullback-Leibler. Cependant, si nous pouvons détecter par nos travaux les conséquences de ces attaques, l'organisation, le comportement et les motivations des attaquants nous sont encore inconnues.

A partir de la détection des attaques, la première étape visant à caractériser les pairs malveillants consisterait à comprendre les moyens mis en œuvre par les attaquants pour les réaliser. Une fois les pairs attaquants identifiés, il serait intéressant de mettre en évidence les différents motifs pouvant caractériser une même attaque, par exemple : les adresses IP impliquées, les ports, le nombre de pairs malveillants insérés, la proximité entre les attaquants, les contenus ciblés, etc. Ensuite, des communications directes avec les attaquants doivent permettre de découvrir leur comportement et le but de l'attaque (éclipse, pollution, etc.). Une supervision des attaques sur le long terme permettrait de suivre leur évolution et notamment celle des contenus ciblés. Cette connaissance des attaquants est nécessaire pour améliorer encore à l'avenir la sécurité des réseaux P2P.

### Protection et amélioration des services distribués

Nous avons proposé un mécanisme de protection contre les attaques ciblées qui empêche un attaquant d'insérer des pairs à proximité d'un contenu à contrôler. Si l'insertion d'un unique pair reste difficile à détecter selon sa distance à la cible, notre solution assure en revanche qu'étant donné les 10 réplicats responsables d'une référence, la majorité de ceux contactés sera honnête. Cette propriété fournie par le mécanisme de protection peut constituer une première brique pour développer d'autres mécanismes de sécurité basés sur un consensus entre les réplicats et garantissant l'émergence de la bonne information.

Avec un stockage fiable des informations dans la DHT, de nouvelles applications sont possibles. Parmi celles-ci, nous avions déjà proposé une modification du mécanisme d'indexation permettant aux pairs responsables des fichiers de détecter la pollution par falsification d'indexation. Une autre application intéressante serait de concevoir un mécanisme de réputation globale basé sur le stockage de la réputation des pairs sur la DHT. Nous avions commencé à concevoir un tel mécanisme dans [CCF08a], mais celui-ci était rendu inefficace à cause de l'attaque Sybil. Cette réputation permettrait de lutter contre l'égoïsme des pairs par l'élaboration de mécanismes incitatifs et contre les comportements malveillants par la révocation des services entre pairs selon leur réputation.

# Productions

## Publications

Les contributions présentées dans ce mémoire ont donné lieu à diverses publications scientifiques :

- **Evaluation of Sybil Attacks Protection Schemes in KAD**, publié à AIMS 2009 [CCF09a], correspond aux résultats présentés dans le chapitre 4 quant à l'évaluation des mécanismes de protections implantés dans KAD contre l'attaque Sybil en concluant sur la faiblesse du client par rapport aux attaques localisées sur la DHT et distribuées.
- **Une architecture de honeypots distribués pour superviser le réseau P2P KAD**, publié à NOTERE 2009 [CCF09b], reprend une partie de l'évaluation des mécanismes de défense et présente les premiers résultats de l'architecture de supervision présentée dans le chapitre 5 de ce manuscrit.
- **Monitoring and Controlling Content Access in KAD**, publié à ICC 2010 [CCF10b], présente en détails notre architecture de supervision pour KAD, ses fonctionnalités et l'évaluation de son efficacité ce qui correspond également au chapitre 5.
- **Efficient DHT attack mitigation through peers' ID distribution**, publié à HotP2P 2010 : [CCF10a], présente notre solution détectant les attaques locales de la DHT par l'analyse de la distribution des identifiants des pairs ainsi que la contre-mesure associée, correspondant ainsi au chapitre 8 de ce manuscrit.
- **Détection de pairs suspects dans le réseau pair à pair KAD**, publié à SAR-SSI 2011 [CHC+11], présente notre exploration précise de la DHT et nos méthodes de détection des pairs suspects, à savoir par densité des pairs et proximités aux contenus, ayant abouti à une estimation des attaques dans KAD. Ceci correspond à la seconde partie du chapitre 7 de ce manuscrit.
- **Content Pollution Quantification in Large P2P networks : a Measurement Study on KAD**, publié à IEEE P2P 2011 [CMD+11], présente notre travail concernant la détection de la pollution par falsification de l'indexation ainsi que sa quantification dans KAD. Ceci correspond à la première partie du chapitre 7 de ce manuscrit.

Les publications suivantes ont également été réalisées pendant cette thèse. Elles présentent des travaux annexes qui ne sont pas détaillés dans ce manuscrit :

- **A Distributed and Adaptive Revocation Mechanism for P2P networks**, publié à ICN 2008 [CCF08a] à partir d'une version initiale en français : **Un mécanisme de révocation distribué et adaptatif pour les réseaux pair-à-pair** publiée à JDIR 2008 [CCF08b] ayant obtenu le prix du meilleur papier. Ces travaux présentent un mécanisme de réputation global où chaque pair dispose d'un compte distribué sur la DHT stockant sa réputation. Après consultation de la réputation d'un pair, les autres peuvent décider de révoquer tout ou partie des services à ce dernier. Un exemple d'application des comptes

distribués présenté dans ces travaux vise à lutter contre l'égoïsme des pairs en instaurant une réputation proportionnelle à la quantité de services fournis/consommés par un pair. Cependant l'attaque Sybil peut affecter notre proposition si la DHT n'en est pas protégée.

- **BitTorrent's Mainline DHT Security Assessment**, publié à NTMS2011 [TCCF11a], met en évidence la vulnérabilité de la DHT de Bittorrent (remplaçant les trackers centralisés) aux attaques ciblées et expérimente plusieurs applications de cette faiblesse. Nous montrons ensuite l'applicabilité de notre mécanisme de détection de ces attaques dans le cadre de cette DHT.
- **When KAD meets BitTorrent - Building a Stronger P2P Network**, publié à HotP2P 2011 [TCCF11b], propose une étude comparative de ces deux réseaux P2P sur des critères de sécurité, de performance et de fonctionnalité. Nous montrons qu'une architecture unifiant ces deux réseaux est possible et permettrait de cumuler leurs avantages respectifs.

### Implantations et données collectées

Les travaux réalisés durant cette thèse revêtent volontairement un aspect pratique : d'abord par l'étude et la collecte de données au sein d'un réseau P2P réel, d'autre part par l'implantation dans ce même réseau, à savoir KAD, des divers algorithmes conçus. Ainsi, l'ensemble de ces travaux a été validé expérimentalement. Cet effort de mise en œuvre a nécessité de nombreuses implantations ainsi que des collectes de données décrites dans ce paragraphe. Néanmoins, les résultats obtenus n'en demeurent pas moins génériques et facilement applicables à d'autres DHT comme l'ont montré des travaux annexes sur MainLine DHT (Bittorrent) ou Gnutella DHT.

Implantations, accessible sur demande par INRIA GForge [25]. :

- **HAMACK** : l'architecture de supervision regroupe le code source des sondes (C++), les scripts de création de la base de données (postgreSQL), les scripts d'administration et de déploiement de l'architecture (perl,bash) ainsi que l'interface web de présentation des données (php).
- **Explorateur pour KAD** : l'explorateur implanté en Java permet l'obtention d'une cartographie précise de la DHT de KAD et fournit l'ensemble des scripts d'analyse permettant de détecter les pairs suspects.
- **Algorithmes de détection d'attaques** : le code source des mécanismes de détection appliqués à KAD sera diffusé prochainement dans le cadre de la communauté d'eMule. Ils incluent la détection de la pollution et la protection des accès à la DHT par filtrage des pairs suspects. L'implantation de ce dernier mécanisme dans le cadre de gtk-gnutella a d'ores et déjà été réalisé par Raphael Manfredi et rendu disponible [26].

Données collectées, disponibles sur demande :

- **Activité de 72 mots-clés** : les données anonymisées et stockées dans une base de données relationnelle retraçant l'activité de 72 mots-clé sur KAD pendant 15 jours.
- **Cartographies de la DHT** : plusieurs cartographies complètes de la DHT de KAD, au format texte ou d'objets sérialisés.
- **Noms contradictoires de fichiers** : l'ensemble des différents noms annoncés par les sources pour 2000 fichiers populaires.
- **Résultats de localisations** : l'ensemble des pairs trouvés pour chacune des 1800 requêtes de localisation.

25. `https://gforge.inria.fr/`
26. `https://gtk-gnutella.svn.sourceforge.net/svnroot/gtk-gnutella/trunk/gtk-gnutella/`

# Liste des figures

# Liste des Algorithmes

# Liste des tableaux

# Annexe A

# Schéma de la base de données d'HAMACK

## A.1 Table keywords

La table *keywords* référence les mots-clés supervisés par l'architecture. Les champs principaux sont :

- keyword : chaîne de caractères définissant le mot-clé
- keyhash : identifiant MD4 du mot-clé dans KAD

Exemple :

```
     keyword      |             keyhash              |
------------------+----------------------------------+
  pokemon         | 05C04B540B9B97D8F587BDEF28E05BF8 |
  pthc            | 11BCFBFF2858E2D93D7D29D728B265CA |
```

## A.2 Table sybils

La table *sybils* contient des informations sur les sondes insérées dans le réseau KAD, chaque sonde activée créant une nouvelle entrée. Cette table est principalement utilisée pendant la collecte, permettant aux sondes supervisant une même référence de se connaître et de collaborer. Les champs principaux sont :

- id : identifiant du Honeypeer dans la base
- targetid : identifiant du mot-clé ou du fichier observé
- ip : adresse IP de l'ordinateur exécutant le Honeypeer
- port : port UDP utilisé pour la connexion de la sonde à KAD
- kadid : KADID du Honeypeer

Exemple :

```
 id  |    targetid    |       ip      |     kadid      | port  |       tmstmp
-----+----------------+---------------+----------------+-------+--------------------
 316 | DF72B7E2...9D7A| 193.174.67.186| DF72B7E2...0F4E| 13609 | 2010-10-25 18:58:15
 317 | F1D63EA4...622E| 138.96.116.20 | F1D63EA4...6E8E| 14449 | 2010-10-25 19:00:34
```

## A.3 Table replicat

La table « replicat » n'est utile que pendant l'exécution. Il s'agit d'une mémoire à court terme des différents messages capturés par l'architecture afin d'éviter l'enregistrement multiple d'une même information répliquée dans le cadre du protocole KAD. Avant d'enregistrer une information, une sonde consulte la table « replicat » afin de vérifier que cette information ne vient pas déjà d'être enregistrée dans la base de données par une autre. Une information est ainsi identifiée par 3 paramètres : l'adresse IP de l'émetteur, la référence visée et la date d'émission. Si ce triplet n'est pas présent dans la table réplicat, l'information peut être stockée dans la base. Cette table est vidée périodiquement (toutes les 10 minutes) pour deux raisons. D'une part car l'ensemble des réplicats d'une même information sont capturés dans un intervalle de temps très court (1 minute). D'autre part, car cette table est fortement sollicitée durant l'exécution et l'augmentation de sa taille dégrade les performances du serveur. Les champs principaux sont :

- ip : identifiant de l'adresse IP du pair émetteur dans la base
- targetid : identifiant du mot-clé ou du fichier visé par la requête
- tmstmp : date de réception de la requête par une sonde

Exemple :

```
  ip  |    targetid    |        tmstmp         |
------+----------------+-----------------------+
52632 |0ACE20E5...1A0A |2010-10-19 15:58:00+02 |
32420 |AC213377...DD35 |2010-10-19 15:58:00+02 |
```

Une fois les données collectées, cette table n'est plus utile.

## A.4 Table searchs

La table « searchs » stocke les informations obtenues pour chaque requête de recherche capturée. Les champs principaux sont :

- id : identifiant de la requête de recherche dans la base
- timestamp : date de réception de la requête
- srcid : identifiant du pair émetteur dans la base (table peers)
- targetid : identifiant MD4 de la référence (du mot-clé dans notre cas) recherché dans KAD

Exemple :

```
   id    |         tmstmp         |  srcid   |             targetid             |
---------+------------------------+----------+----------------------------------+
    3489 | 2010-10-19 15:38:00+02 |    81768 | DF72B7E2475E89BAB57A063084439D7A |
    3490 | 2010-10-19 15:39:00+02 |    81768 | DF72B7E2475E89BAB57A063084439D7A |
```

## A.5 Table searched-keywords

Les mots-clés complémentaires associés avec une recherche (pour introduire des contraintes supplémentaires) sont stockés dans la table « searched-keywords », dont les champs principaux sont :

- id : identifiant du mot-clé associé dans la base
- idsearch : identifiant dans la base de la requête de recherche incluant le mot associé (table searchs)

– keyword : chaîne de caractères définissant le mot-clé associé ;

Exemple :

```
   id   |  idsearch |      keyword     |
--------+-----------+------------------+
    783 |      3964 | eyed             |
    784 |      3964 | peas             |
```

## A.6 Table publishs

La table *publishs* stocke les informations obtenues pour chaque requête de publication capturée. Les champs principaux sont :

– id : identifiant de la requête de publication dans la base
– tmstmp : date de réception de la requête
– srcid : identifiant du pair émetteur dans la base (table peers)
– fileid : identifiant MD4 (d'un **mot-clé** ou fichier) référençant le contenu publié (un fichier ou une source) dans KAD
– targetid : identifiant MD4 du contenu publié (un **fichier** ou une source) dans KAD

Exemple :

```
   id   |         tmstmp          | srcid |     fileid      |    targetid     |
--------+-------------------------+-------+-----------------+-----------------+
 134067 | 2010-10-19 15:37:00+02 | 81742 | DF72B7E2...9D7A | 87860001...6398 |
 134068 | 2010-10-19 15:37:00+02 | 81743 | DF72B7E2...9D7A | 93600001...710A |
```

## A.7 Table published-files

La table « published-files » stocke les informations des différents fichiers partagés à travers les requêtes de publication. Les champs principaux sont :

– id : identifiant du fichier publié dans la base
– idpublish : identifiant dans la base de la requête de publication ayant annoncé le fichier (table publishs)
– fileid : identifiant MD4 du fichier sur le réseau KAD
– name : chaîne de caractères définissant le nom complet du fichier
– size : taille du fichier en Mo

Exemple :

```
  id   | idpublish |    fileid    |               name               |     size   |
-------+-----------+--------------+----------------------------------+------------+
450267 |    134067 | 878600...6398| Afro Samourai Saison 1 FR...     |  726.317078|
450268 |    134068 | 936000...710A| Friends.-.Saison.07.Complète...| 2281.513184|
 50269 |    134069 | 331F00...2BC3| New york district saison 05... |  367.048706|
```

## A.8 Table peers

La table « peers » stocke les informations concernant les pairs observés par l'architecture. Les champs principaux sont :

– id : identifiant du pair dans la base

- ipid : identifiant unique de l'adresse IP du pair dans la base pendant toute la collecte (table ips)
- port : port udp utilisé par le pair pour se connecter à KAD
- tmstmp : date lors de la première observation du pair

Exemple :

```
   id   |   ipid  | port  |         tmstmp        |
--------+---------+-------+-----------------------+
 852933 |  118370 |  4178 | 2010-10-20 02:38:00+02|
 853334 |  118371 | 59835 | 2010-10-20 02:38:00+02|
```

## A.9 Table ips

La table « ips » stocke les informations sur les adresses IP des pairs. Les champs principaux sont :

- id : identifiant de l'adresse IP
- ip : adresse IP du pair observé au format inet

Exemple :

```
   id   |      ip     |
--------+-------------+
   3241 |  152.81.X.X |
   3242 |  193.60.X.X |
```

Le champ « ip » est supprimé de la table à l'issue de la collecte de manière à anonymiser les adresses IP observées. Le champ « id » permet toujours de lier une adresse IP unique aux requêtes qu'elle a émises même si celle-ci n'est plus identifiable.

## A.10 Table geoloc

La table « geoloc » stocke les informations de géolocalisation obtenues d'après les adresses IP des pairs. Ceci est rendu possible par un post-traitement réalisé à l'issue de la collecte et avant l'anonymisation des adresses IP collectées. Leur localisation approximative (au niveau d'une ville) est ainsi obtenue grâce à la base de connaissances et aux librairies fournies par GeoLite City [27] de MaxMind.

Les champs principaux sont :

- id : identifiant de l'adresse IP dans la base
- country : pays estimé du pair, d'après son adresse IP
- city : ville estimée du pair, d'après son adresse IP

Exemple :

```
    id    |   country     |    city    | area | region |
----------+---------------+------------+------+--------+
   495613 | China         | Beijing    |    0 |  22    |
  5219843 | Mexico        | Mexico     |    0 |  09    |
  5223996 | United States | New York   |  212 |  NY    |
```

Les champs « area » et « region » sont peu utiles car ils sont surtout renseignés pour les adresses IP situées aux États-Unis.

27. http ://www.maxmind.com/app/geolitecity

# Annexe B

# Analyses supplémentaires de la base de données

## B.1 Quantification des publications par mot-clé

La matrice B.3 est légèrement différente de celle présentée pour les recherches (matrice 6.12). Les coocurences sont divisées pour chaque ligne par la somme des publications dans lesquelles le mot-clé apparaît. La matrice n'est donc plus symétrique mais permet d'identifier le pourcentage de corrélation entre deux mots publiés. Les couleurs primaires représentent les même catégories de mots-clé que pour la matrice 6.12 mais un degradé de couleur est en plus utilisé pour rendre compte de l'importance de la corrélation, les différentes intensités représentant chacune 20% supplémentaires.

## B.2 Ratio recherche / publication

## B.3 Étude des fichiers publiés

## B.4 Étude des pairs

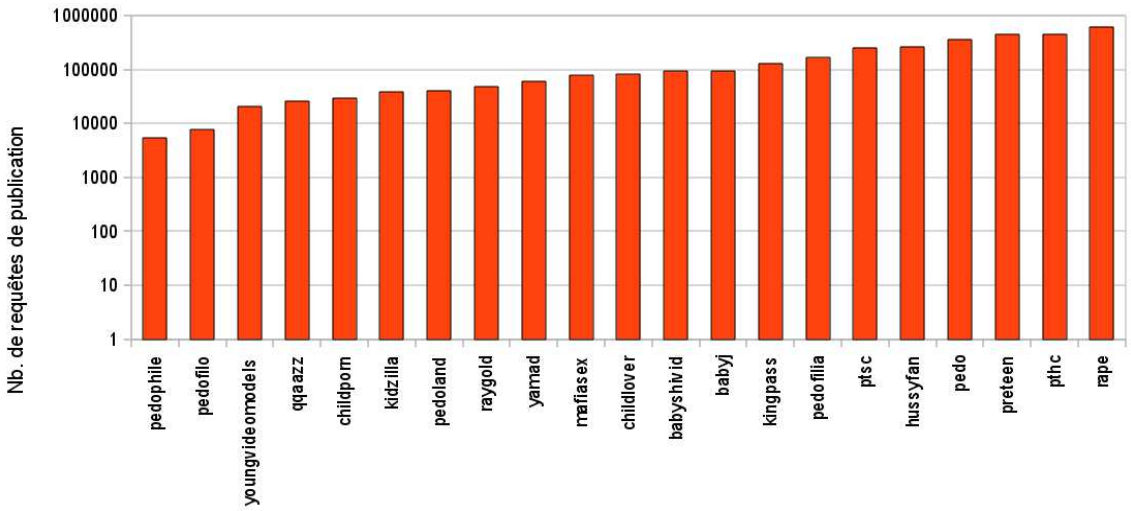


FIGURE B.1 – Nombre de requêtes de publication capturées pour les mots-clé pédo-explicites

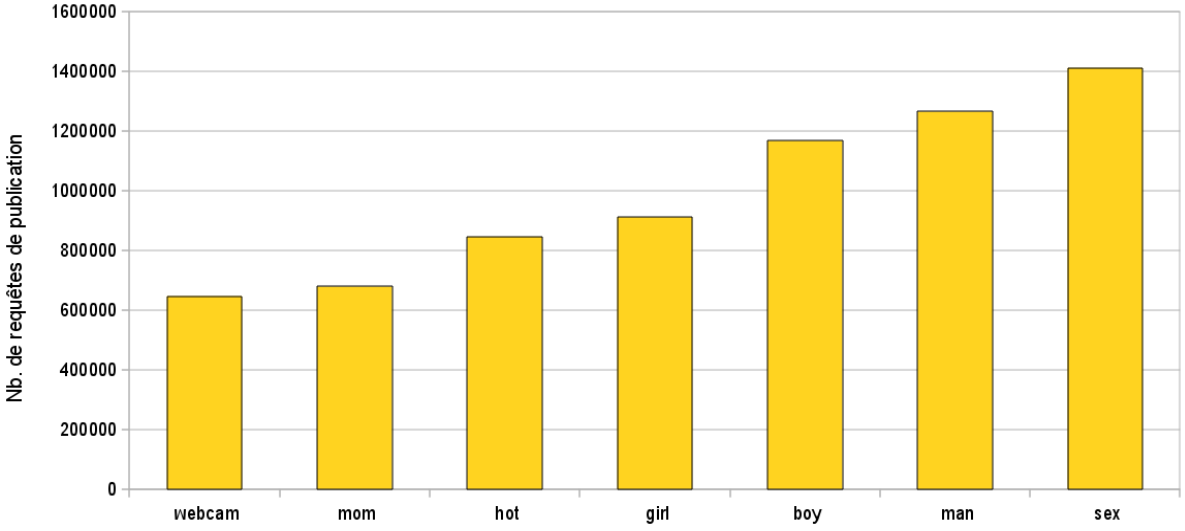


FIGURE B.2 – Nombre de requêtes de publication capturées pour les mots-clé pédo-liés

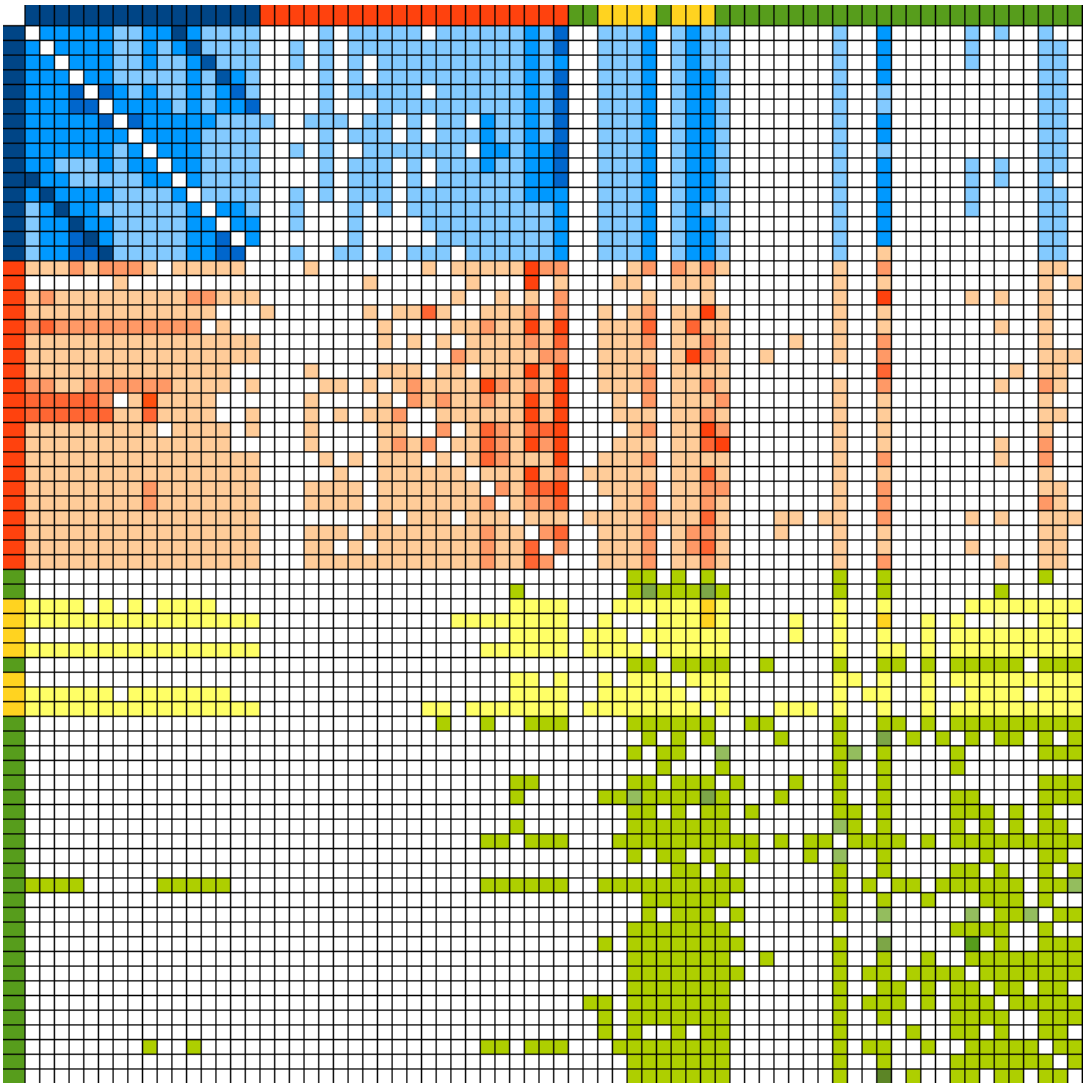

FIGURE B.3 – Matrice de corrélation des mots-clé selon les publications

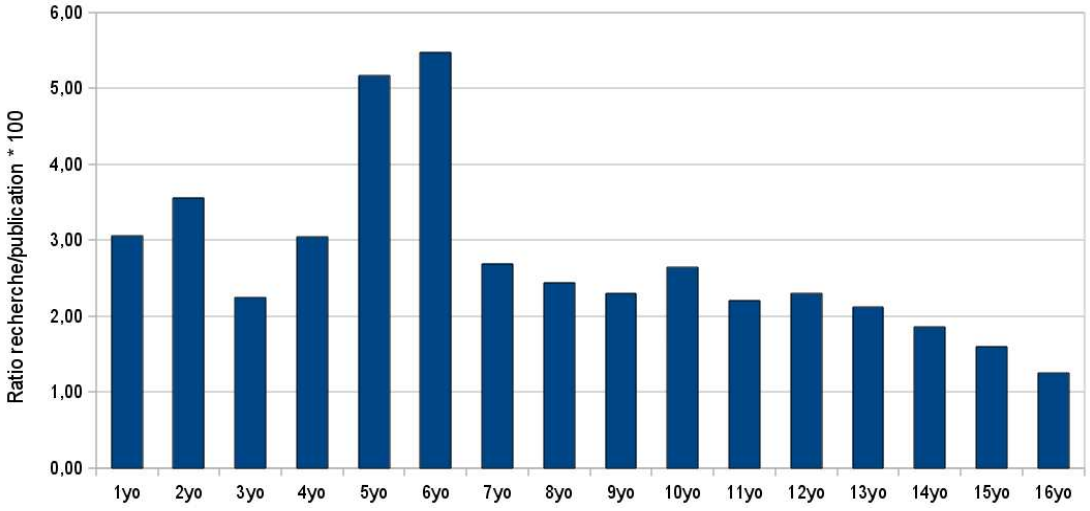


FIGURE B.4 – Ratio de recherches / publications capturées selon les âges

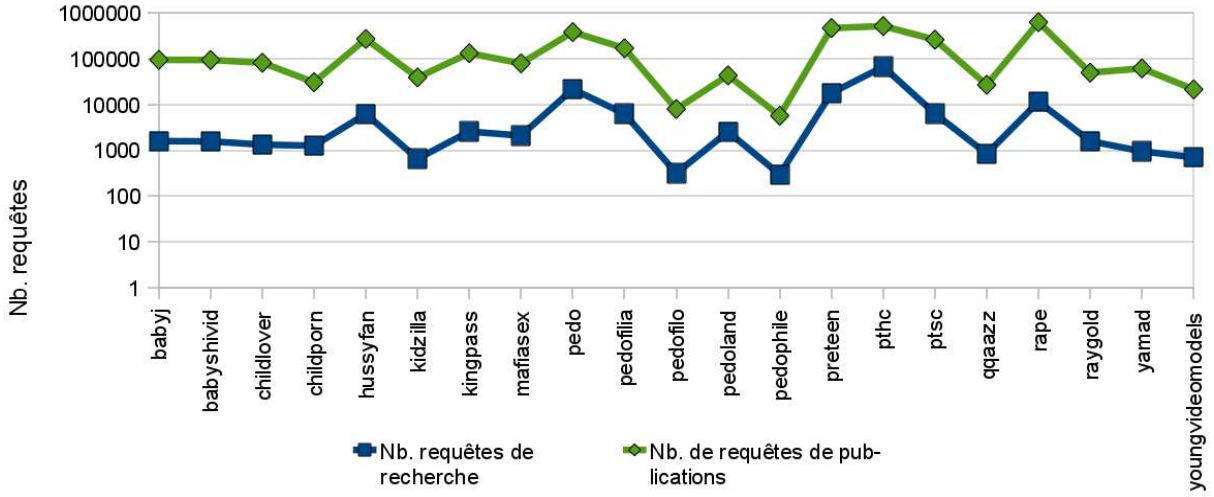


FIGURE B.5 – Nombre de requetes capturées pour les recherches et publications selon les mots-clé pédo-explicites

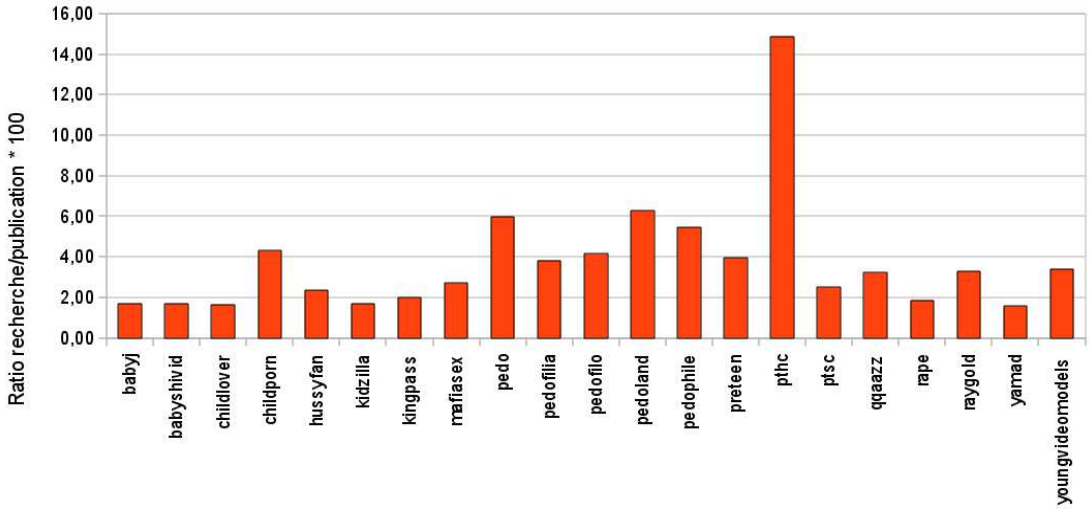


FIGURE B.6 – Ratio de recherches / publications capturées selon les mots-clé pédo-explicites

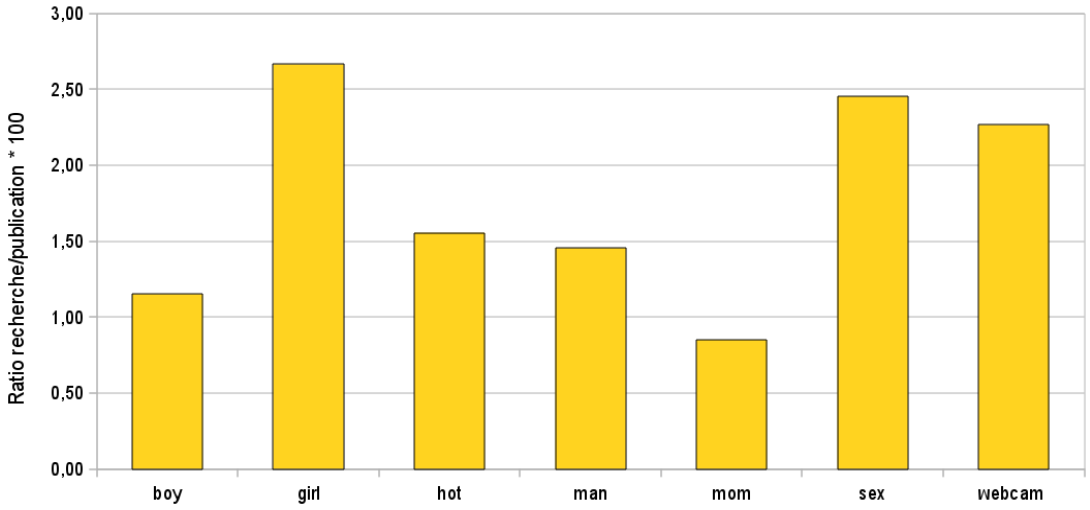


FIGURE B.7 – Ratio de recherches / publications capturées selon les mots-clé pédo-liés

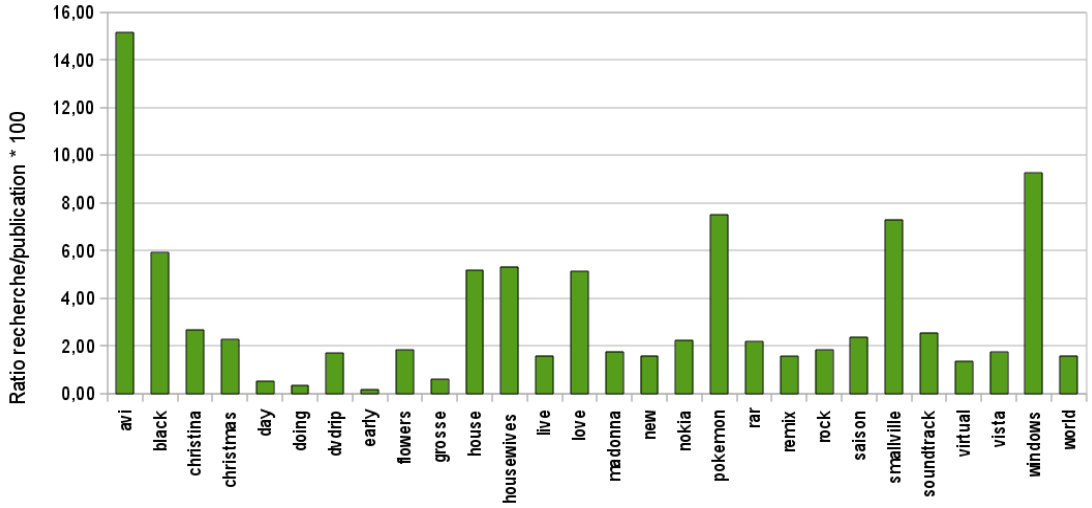


FIGURE B.8 – Ratio de recherches / publications capturées selon les mots-clé normaux

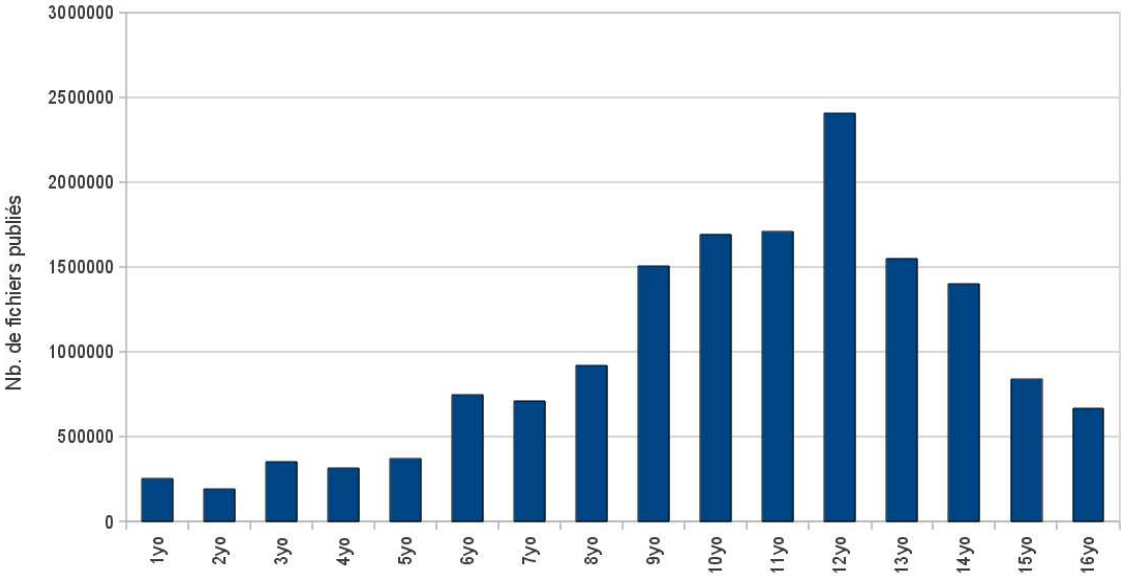


FIGURE B.9 – Nombre de fichiers publiés selon les âges

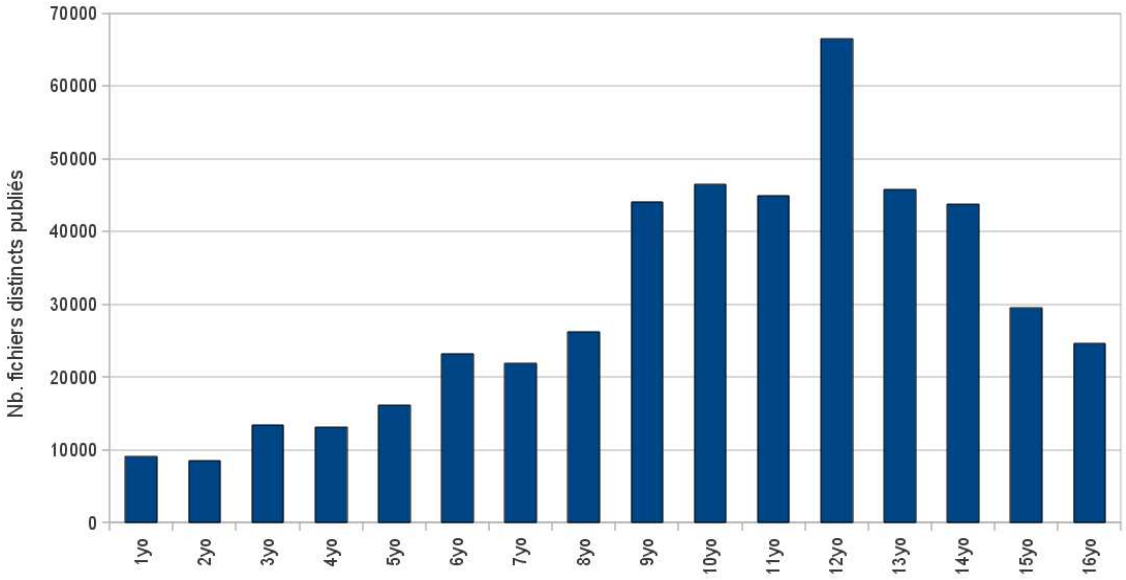


FIGURE B.10 – Nombre de fichiers distincts publiés selon les âges

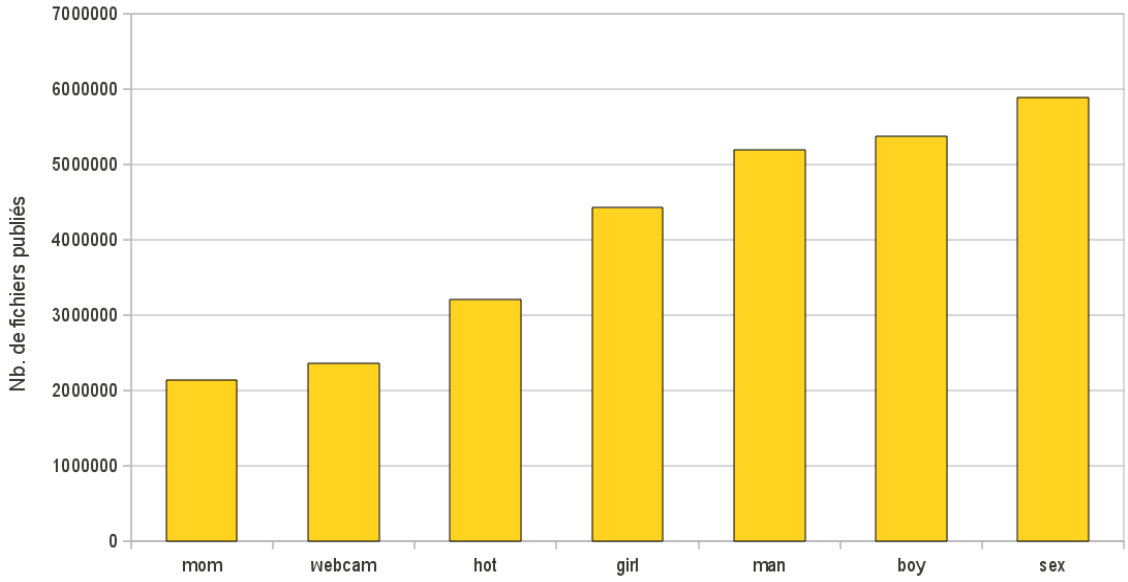


FIGURE B.11 – Nombre de fichiers publiés pour les mots-clé pédo-liés

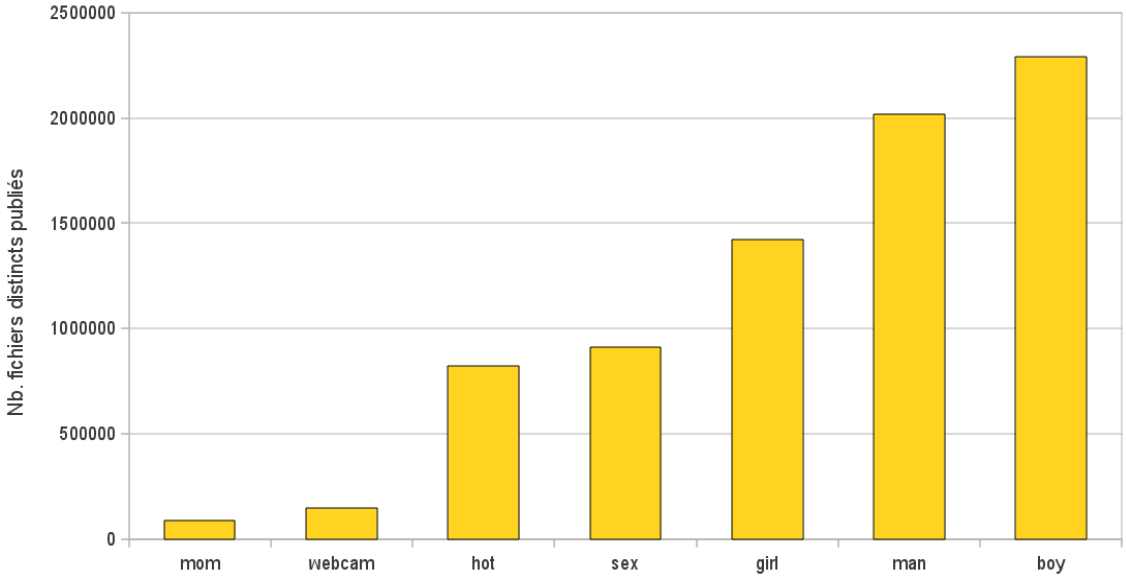


FIGURE B.12 – Nombre de fichiers distincts publiés pour les mots-clé pédo-liés

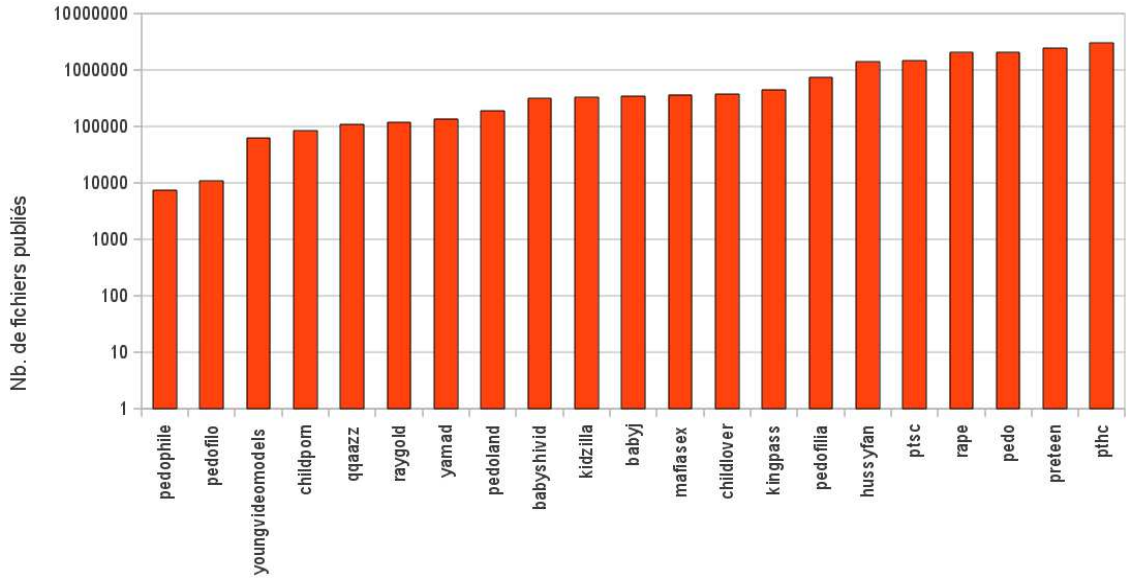


FIGURE B.13 – Nombre de fichiers publiés pour les mots-clé pédo-explicites

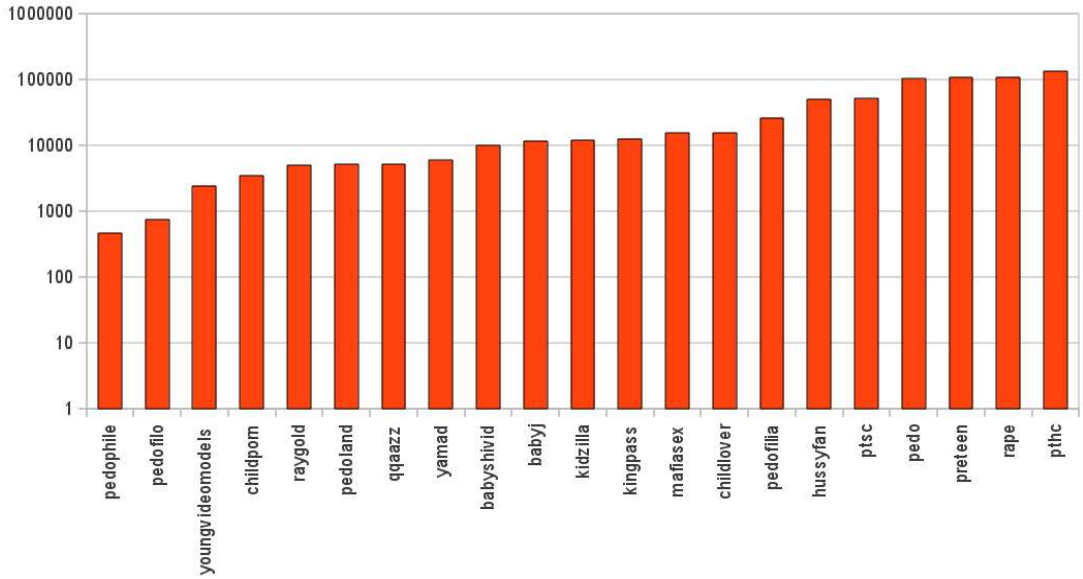


FIGURE B.14 – Nombre de fichiers distincts publiés pour les mots-clé pédo-explicites

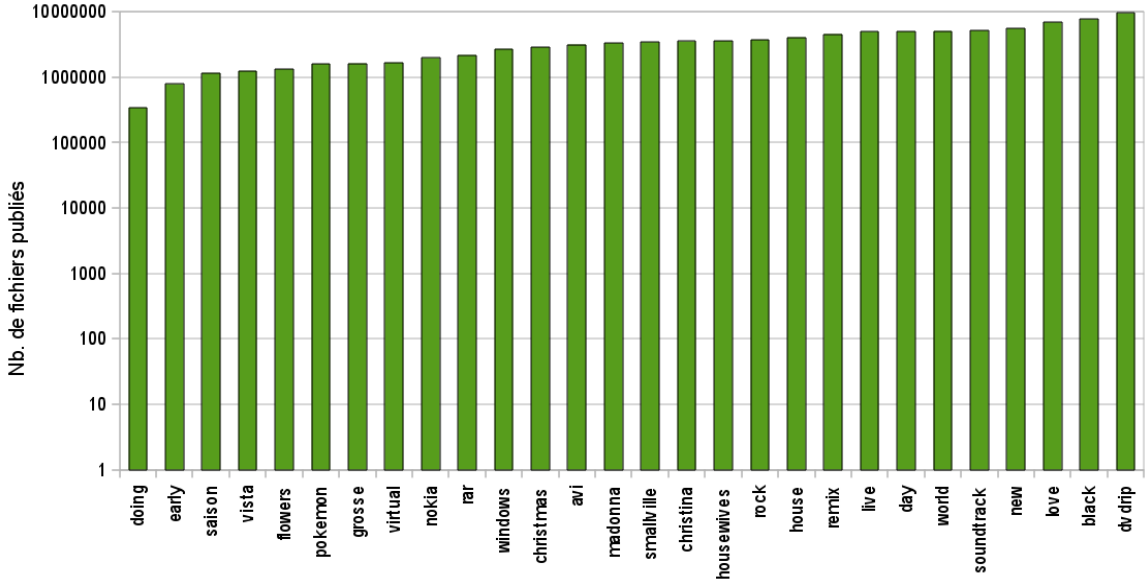


FIGURE B.15 – Nombre de fichiers publiés pour les mots-clé normaux

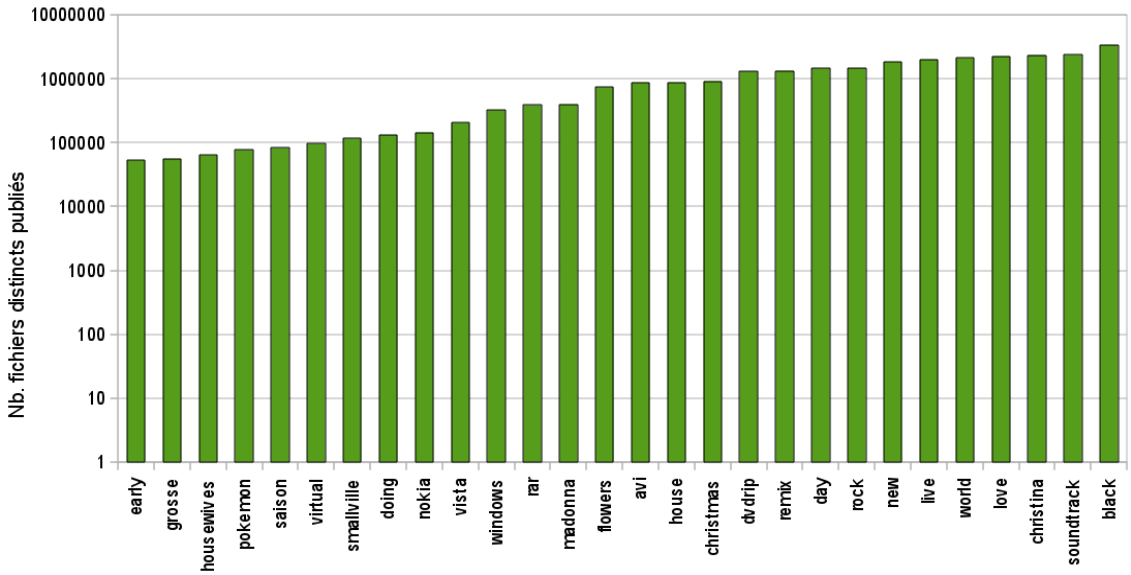


FIGURE B.16 – Nombre de fichiers distincts publiés pour les mots-clé normaux

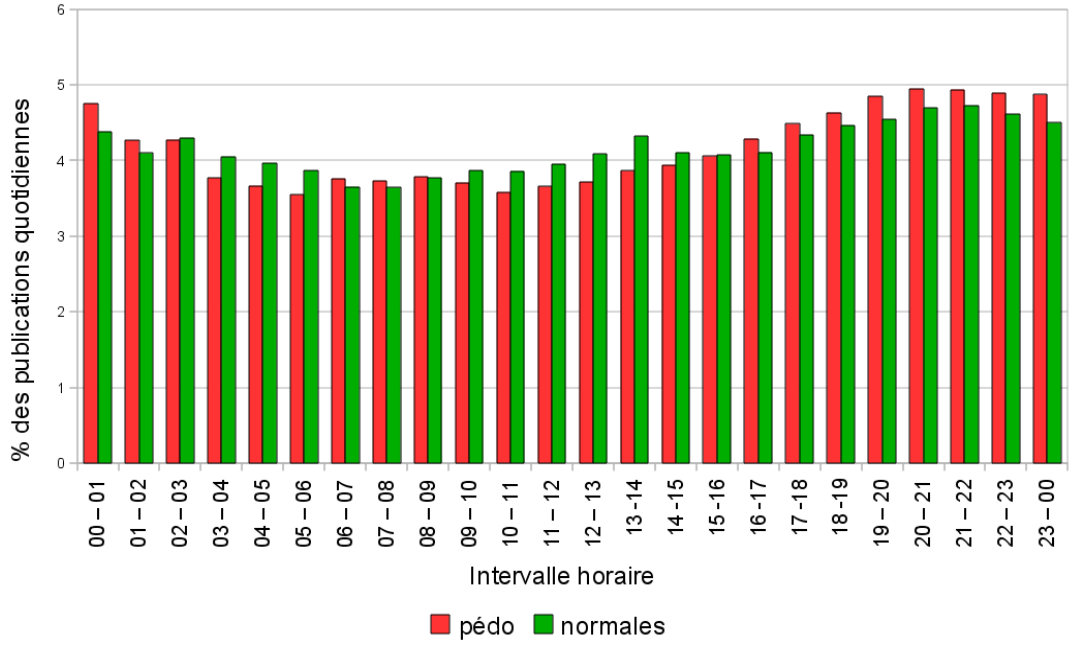


FIGURE B.17 – Distribution des requêtes de publication reçues par intervalle horaire

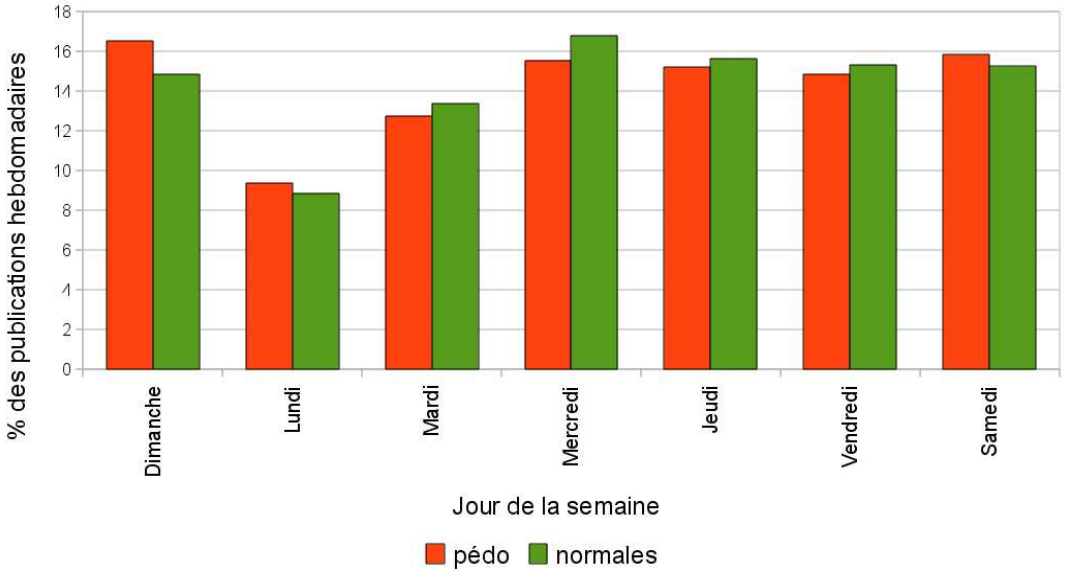


FIGURE B.18 – Distribution journalière des requêtes de publication capturées

# Annexe C

# Liste de 100 contenus populaires téléchargés en 2010

```
1  inception
2  iron man 2
3  2010
4  xxx
5  french
6  avatar
7  dvdrip
8  despicable me
9  porn
10 clash of the titans
11 toy story 3
12 glee
13 salt
14 twilight eclipse
15 dexter
16 apprentice sorcerer
17 axxo
18 robin hood
19 prince of persia
20 windows 7
21 greek get him
22 predators
23 airbender last
24 shutter island
25 knight and day
26 expendables
27 takers
28 dinner for schmucks
29 unstoppable
30 eli book
31 grown ups
32 true blood
33 alice in wonderland
34 movies
35 shrek forever after
36 supernatural
37 hindi
38 house
39 devil
40 step up 3d
41 megamind
42 harry potter and the deathly hallows
43 skyline
44 green zone
45 naughty america
46 eminem
47 lost
48 town
49 date night
50 wolfman
```

FIGURE C.1 – Liste remaniée de 100 contenus populaires téléchargés en 2010 d'après un site populaire d'indexation de torrents (1/2)

```
1  smallville
2  last song
3  torrents
4  dragon how to train your
5  fringe
6  dear john
7  red
8  social network
9  weeds
10 noir
11 pc games
12 vampire diaries
13 2012
14 twilight
15 cop out
16 tamil
17 city 2 sex and the
18 remember me
19 walking dead
20 eclipse
21 due date
22 fxg
23 grown ups
24 entourage
25 sherlock holmes
26 how i met your mother
27 sex
28 microsoft office 2010
29 spartacus
30 pacific
31 karate kid
32 other guys the
33 call of duty black ops
34 chuck
35 ita
36 resident evil
37 wii
38 hot tub time machine
39 ubuntu
40 nero
41 ncis
42 theory big bang
43 indiana jones
44 tron 2
45 lady gaga
46 raiponce
47 photoshop
48 black swan
49 tourist the
```

FIGURE C.2 – Liste remaniée de 100 contenus populaires téléchargés en 2010 d'après un site populaire d'indexation de torrents (2/2)

# Bibliographie


[AC07] 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.

[AG09] Benjamin Bayart Astrid Girardeau. Tout le monde a intérêt à transformer internet en minitel. `http://www.ecrans.fr/Tout-le-monde-a-interet-a,5762.html`, 2009.

[AH00] Eytan Adar and Bernardo A. Huberman. Free riding on Gnutella. *First Monday*, 5(10), October 2000.

[AHIZ06] Kostas G. Anagnostakis, Fotios C. Harmantzis, Sotiris Ioannidis, and Manaf Zghaibeh. On the impact of practical p2p incentive mechanisms on user behavior. Technical report, NET Institute, September 2006.

[ALM09a] Frederic Aidouni, Matthieu Latapy, and Clémence Magnien. Ten weeks in the life of an edonkey server. In *IPDPS* [IEE09], pages 1–5.

[ALM09b] Oussama Allali, Matthieu Latapy, and Clémence Magnien. Measurement of edonkey activity with distributed honeypots. In *IPDPS* [IEE09], pages 1–8.

[AMRS08] Luca M. Aiello, Marco Milanesio, Giancarlo Ruffo, and Rossano Schifanella. Tempering kademlia with a robust identity based system. In *P2P '08 : Proceedings of the 2008 Eighth International Conference on Peer-to-Peer Computing*, pages 30–39, Washington, DC, USA, 2008. IEEE Computer Society.

[And01] Kelsey Anderson. An analysis of the traffic on the gnutella network. Technical report, University of California, San Diego, CA, USA, March 2001.

[BFB08] Anirban Banerjee, Michalis Faloutsos, and Laxmi Bhuyan. The p2p war : Someone is monitoring your activities. *Comput. Netw.*, 52(6) :1272–1280, 2008.

[BMGS09] 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.

[BP04] Chadi Barakat and Ian Pratt, editors. *Passive and Active Network Measurement, 5th International Workshop, PAM 2004, Antibes Juan-les-Pins, France, April 19-20, 2004, Proceedings*, volume 3015 of *Lecture Notes in Computer Science*. Springer, 2004.

[Bru06] Rene Brunner. A performance evaluation of the Kad-protocol. Master's thesis, University of Mannheim and Institut Eurecom, 2006.

[BSCF07] 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.

[CA07] 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.

[CB07] Damiano Carra and Ernst W Biersack. Building a reliable p2p system out of unreliable p2p clients : the case of kad. In *CoNEXT 2007, 3rd International Conference on emerging Networking Experiments and Technologies, December 10-13, 2007, New York, NY, USA*, 12 2007.

[CCF08a] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. A Distributed and Adaptive Revocation Mechanism for P2P networks. In *Seventh International Conference on Networking - ICN 2008*, pages pp. 290–295, Cancun Mexique, 04 2008.

[CCF08b] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Un mécanisme de révocation distribué et adaptatif pour les réseaux pair-à-pair. In *9ème journées Doctorales en Informatique et Réseaux - JDIR 2008*, page 87, Villeneuve d'Ascq France, 01 2008. Prix du meilleur papier.

[CCF09a] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Evaluation of Sybil Attacks Protection Schemes in KAD. In Ramin Sadre and Aiko Pras, editors, *3rd International Conference on Autonomous Infrastructure, Management and Security - AIMS 2009 Scalability of Networks and Services*, volume 5637 of *Lecture Notes in Computer Science*, pages 70–82, Enschede Pays-Bas, 2009. University of Twente, Springer.

[CCF09b] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Une architecture de honeypots distribués pour superviser le réseau P2P KAD. In Abdel Obaïd, editor, *9e Conférence Internationale sur Les NOuvelles TEchnologies de la REpartition*, pages 76–82, Montréal Canada, 2009. Université du Québec à Montréal.

[CCF10a] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Efficient DHT attack mitigation through peers' ID distribution. In *Seventh International Workshop on Hot Topics in Peer-to-Peer Systems - HotP2P 2010*, Atlanta États-Unis, 04 2010. IEEE International Parallel & Distributed Processing Symposium.

[CCF10b] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Monitoring and Controlling Content Access in KAD. In *International Conference on Communications - ICC 2010*, Capetown Afrique Du Sud, 05 2010. IEEE.

[CCR05] Miguel Castro, Manuel Costa, and Antony Rowstron. Debunking some myths about structured and unstructured overlays. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 85–98, Berkeley, CA, USA, 2005. USENIX Association.

[CDG+02] 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.

[CF05] Alice Cheng and Eric Friedman. Sybilproof reputation mechanisms. In *Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, P2PECON '05, pages 128–132, New York, NY, USA, 2005. ACM.

[Cha08] Guillaume Champeau. emule : tous les serveurs edonkey razorback fermés. http://www.numerama.com/magazine/10250-emule-tous-les-serveurs-edonkey-razorback-fermes.html, 2008.

[Cha09a] Guillaume Champeau. Filtrage : tous les gros fai brident le p2p au canada. http://www.numerama.com/magazine/11763-filtrage-tous-les-gros-fai-brident-le-p2p-au-canada.html, 2009.

[Cha09b] 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.

[Cha10] 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.

[CHC+11] Thibault Cholez, Christopher Hénard, Isabelle Chrisment, Olivier Festor, Guillaume Doyen, and Rida Khatoun. Détection de pairs suspects dans le réseau pair à pair KAD. In *SAR-SSI 2011 : 6ème Conf. sur la Sécurité des Architectures Réseaux et Systèmes d'Information*, La Rochelle, France, May 2011. IEEE. Financement GIS - 3SGS - Projet ACDAP2P.

[CJ05] Kenjiro Cho and Philippe Jacquet, editors. *Technologies for Advanced Heterogeneous Networks, First Asian Internet Engineering Conference, AINTEC 2005, Bangkok, Thailand, December 13-15, 2005, Proceedings*, volume 3837 of *Lecture Notes in Computer Science*. Springer, 2005.

[CKS+06] Tyson Condie, Varun Kacholia, Sriram Sank, Joseph M. Hellerstein, and Petros Maniatis. Induced churn as shelter from routing-table poisoning. In *NDSS* [Int06].

[CMD+11] Thibault Cholez, Guillaume Montassier, Guillaume Doyen, Rida Khatoun, Isabelle Chrisment, and Olivier Festor. Content pollution quantification in large p2p networks : a measurement study on KAD. In *P2P '11 : Proceedings of the Eleventh IEEE International Conference on Peer-to-Peer Computing*, Kyoto, Japon, August 2011. IEEE.

[CRB+03] 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.

[CW07] Scott A Crosby and Dan S. Wallach. An analysis of bittorrent's two kademlia-based dhts. Technical Report TR-07-04, Department of Computer Science, Rice University, June 2007.

[CWC05] 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.

[Dav09] Julia Davidson. Internet child abuse : Understanding offender online grooming behaviour. In *Proceedings of the 1st international conference on Advances in the Analysis of Online Paedophile Activity*, pages 39–55, Paris, France, 2009.

[DH06] Jochen Dinger and Hannes Hartenstein. Defending the sybil attack in P2P networks : taxonomy, challenges, and a proposal for self-registration. In *First International Conference on Availability, Reliability and Security (ARES 2006)*, pages 756–763, April 2006.

[DHX09] Lingli Deng, Yeping He, and Ziyao Xu. Combating index poisoning in p2p file sharing. In *Proceedings of the 3rd International Conference and Workshops on Advances in Information Security and Assurance*, ISA '09, pages 358–367, Berlin, Heidelberg, 2009. Springer-Verlag.

[DKK+05] 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.

[DLLKA05] George Danezis, Chris Lesniewski-Laas, M. Frans Kaashoek, and Ross J. Anderson. Sybil-resistant dht routing. In di Vimercati et al. [dVSG05], pages 305–318.

[Dou02] 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.

[Doy05] Guillaume Doyen. *Supervision des réseaux et services pair à pair.* PhD thesis, Université Henri Poincaré - Nancy I, 12 2005.

[DR04] Peter Dadam and Manfred Reichert, editors. *INFORMATIK 2004 - Informatik verbindet, Band 2, Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Ulm, 20.-24. September 2004*, volume 51 of *LNI*. GI, 2004.

[dVSG05] Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors. *Computer Security - ESORICS 2005, 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005, Proceedings*, volume 3679 of *Lecture Notes in Computer Science*. Springer, 2005.

[FC05] Michal Feldman and John Chuang. Overcoming free-riding behavior in peer-to-peer systems. *SIGecom Exch.*, 5(4) :41–50, 2005.

[FMR+09] Romano Fantacci, Leonardo Maccari, Matteo Rosi, Luigi Chisci, Luca Maria Aiello, and Marco Milanesio. Avoiding eclipse attacks on kad/kademlia : an identity based approach. In *Proceedings of the 2009 IEEE international conference on Communications*, ICC'09, pages 983–987, Piscataway, NJ, USA, 2009. IEEE Press.

[For00] The Gnutella Developer Forum. Gnutella 0.4 specifications. `http://rfc-gnutella.sourceforge.net/developer/stable/index.html`, 2000.

[For02] The Gnutella Developer Forum. Gnutella 0.6 specifications. `http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html`, 2002.

[FPJ+07] 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.

[FSY05] Amos Fiat, Jared Saia, and Maxwell Young. Making chord robust to byzantine attacks. In *In Proc. of the European Symposium on Algorithms (ESA*, pages 803–814. Springer, 2005.

[GC09] Anthony Hémond Guillaume Champeau. Filtrage du p2p au canada. `http://www.numerama.com/magazine/`

11794-Filtrage-du-P2P-au-Canada-cette-vision-d-Internet-est-inacceptable.html, 2009.

[gdtedn06] Commission générale de terminologie et de néologie. Vocabulaire de l'informatique, jorf n°111 du 13 mai 2006 (p. 7072, texte n° 130). http://legifrance.gouv.fr/affichTexte.do?cidTexte=JORFTEXT000000608830&dateTexte=&fastPos=1, 2006.

[Gir09] Astrid Girardeau. Mininova doit faire le maxi ménage. http://ecrans.fr/Mininova-doit-faire-le-maxi-menage,7939.html, 2009.

[GLLB04] 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.

[GSR+09] Kalman Graffi, Dominik Stingl, Julius Rueckert, Aleksandra Kovacevic, and Ralf Steinmetz. Monitoring and management of structured peer-to-peer systems. In Schulzrinne et al. [SAD09], pages 311–320.

[Har68] Garrett Hardin. The tragedy of the commons. *Science*, 162(3859) :1243–1248, December 1968.

[HBMS04] Oliver Heckmann, Axel Bock, Andreas Mauthe, and Ralf Steinmetz. The edonkey file-sharing network. In Dadam and Reichert [DR04], pages 224–228.

[HCW05] Daniel Hughes, Geoff Coulson, and James Walkerdine. Free riding on gnutella revisited : The bell tolls ? *IEEE Distributed Systems Online*, 6(6) :1, 2005.

[HKL+06] 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.

[HSD+08] Thorsten Holz, Moritz Steiner, Frederic Dahl, Ernst Biersack, and Felix Freiling. Measurements and mitigation of peer-to-peer-based botnets : a case study on storm worm. In *LEET'08 : Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, pages 1–9, Berkeley, CA, USA, 2008. USENIX Association.

[HWCG06] Daniel Hughes, James Walkerdine, Geoff Coulson, and Stephen Gibson. Peer-to-peer : Is deviant behavior the norm on p2p file-sharing networks ? *IEEE Distributed Systems Online*, 7, 2006.

[HZNR09] Kevin Hoffman, David Zage, and Cristina Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Comput. Surv.*, 42(1) :1–31, 2009.

[IEE06] IEEE Computer Society. *INFOCOM 2006. 25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 23-29 April 2006, Barcelona, Catalunya, Spain*. IEEE, 2006.

[IEE07] IEEE Computer Society. *INFOCOM 2007. 26th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 6-12 May 2007, Anchorage, Alaska, USA*. IEEE, 2007.

[IEE09] IEEE Computer Society. *23rd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2009, Rome, Italy, May 23-29, 2009*. IEEE, 2009.

[Int06] The Internet Society. *Proceedings of the Network and Distributed System Security Symposium, NDSS 2006, San Diego, California, USA*. The Internet Society, 2006.

[Ipo09] Ipoque. Internet study 2008/2009. `http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009`, 2009.

[IUKB+04] 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*, volume 3015, pages 1–11. LNCS, April 2004.

[JJ09] B. S. Jyothi and D. Janakiram. Symon : Defending large structured p2p systems against sybil attack. In Schulzrinne et al. [SAD09], pages 21–30.

[JOK09] Raúl Jiménez, Flutra Osmani, and Björn Knutsson. Connectivity properties of mainline bittorrent dht nodes. In Schulzrinne et al. [SAD09], pages 262–270.

[KBFc04] 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.

[KBM07] 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.

[KCTHK09] Hun Jeong Kang, Eric Chan-Tin, Nicholas Hopper, and Yongdae Kim. Why kad lookup fails. In Schulzrinne et al. [SAD09], pages 121–130.

[KGG02] Stefan S. Krishna, P. Krishna Gummadi, and Steven D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Multimedia Computing and Networking (MMCN)*, January 2002.

[KLR09] 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.

[KLVW04] 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.

[KMT03] Yongdae Kim, Daniele Mazzocchi, and Gene Tsudik. Admission control in peer groups. In *Proceedings of the Second IEEE International Symposium on Network Computing and Applications*, NCA '03, pages 131–, Washington, DC, USA, 2003. IEEE Computer Society.

[KSGM03] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 640–651, New York, NY, USA, 2003. ACM.

[Lat09] Matthieu Latapy. Measurement and analysis of p2p activity against paedophile content : Results. `http://antipaedo.lip6.fr/results.htm`, 2009.

[LBGL05] 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.

[LC08] Zhoujun Li and Xiaoming Chen. Misusing kademlia protocol to perform ddos attacks. *Parallel and Distributed Processing with Applications, International Symposium on*, 0 :80–86, 2008.

[LCC+06] 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.

[Leg06] Legifrance. Code de la propriété intellectuelle, article l335-2-1 créé par loi n°2006-961 du 1 août 2006 - art. 21 jorf 3 août 2006. `http://www.legifrance.gouv.fr/affichCodeArticle.do;jsessionid=CED3FB2034C5477BB1F784BC9EE55823.tpdjo15v_1?cidTexte=LEGITEXT000006069414&idArticle=LEGIARTI000006279235&dateTexte=20100815&categorieLien=id#LEGIARTI000006279235`, 2006.

[LGC+09] R. Landa, D. Griffin, R. G. Clegg, E. Mykoniati, and M. Rio. A sybilproof indirect reciprocity mechanism for Peer-to-Peer networks. In *INFOCOM 2009. The 28th Conference on Computer Communications. IEEE*, pages 343–351, 2009.

[lGLMV09] Jean loup Guillaume, Matthieu Latapy, Clémence Magnien, and Guillaume Valadon. Content rating and fake detection system. Technical report, Measurement and Analysis of P2P Activity Against Paedophile Content Project, Laboratoire d'Informatique de Paris 6 - CNRS and Université Pierre et Marie Curie, 2009.

[LHKM04] 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.

[LKR04] J. Liang, R. Kumar, and K. Ross. Understanding KaZaA. Technical report, Department of Computer and Information Science, Polytechnic University, Brooklyn, New York, USA, 2004.

[LKXR05] Jian Liang, Rakesh Kumar, Yonjian Xi, and Keith W Ross. Pollution in p2p file sharing systems. In *IN IEEE INFOCOM*, pages 1174–1185, 2005.

[LL08] Chris Lesniewski-Laas. A sybil-proof one-hop dht. In *SocialNets '08 : Proceedings of the 1st Workshop on Social Network Systems*, pages 19–24, New York, NY, USA, 2008. ACM.

[LMF10] Matthieu Latapy, Clémence Magnien, and Raphael Fournier. Quantifying paedophile activity in a large p2p system. Technical report, Measurement and Analysis of P2P Activity Against Paedophile Content Project, Laboratoire d'Informatique de Paris 6 - CNRS and Université Pierre et Marie Curie, 2010.

[LMFS09] Matthieu Latapy, Clémence Magnien, Raphael Fournier, and Massoud Seifi. Maps of paedophile activity. Technical report, Measurement and Analysis of P2P Activity Against Paedophile Content Project, Laboratoire d'Informatique de Paris 6 - CNRS and Université Pierre et Marie Curie, 2009.

[LMSW09] Thomas Locher, David Mysicka, Stefan Schmid, and Roger Wattenhofer. A Peer Activity Study in eDonkey and Kad. In *International Workshop on Dynamic Networks : Algorithms and Security (DYNAS), Wroclaw, Poland*, September 2009.

[LMSW10] Thomas Locher, David Mysicka, Stefan Schmid, and Roger Wattenhofer. Poisoning the Kad Network. In *11th International Conference on Distributed Computing and Networking (ICDCN), Kolkata, India*, January 2010.

[LMT08a] Francois Lesueur, Ludovic Mé, and Valérie Viet Triem Tong. Detecting and excluding misbehaving nodes in a P2P network. In *Proceedings of the 8th International Conference on Innovative Internet Community Systems (I2CS)*, Schoelcher, Martinique, jun 2008.

[LMT08b] Francois Lesueur, Ludovic Mé, and Valérie Viet Triem Tong. A sybil-resistant admission control coupling SybilGuard with distributed certification. In *Proceedings of the 4th International Workshop on Collaborative Peer-to-Peer Systems (COPS)*, Rome, Italy, jun 2008. IEEE Computer Society.

[LMT09] Franois Lesueur, Ludovic Mé, and Valérie Viet Triem Tong. An efficient distributed pki for structured p2p networks. In Schulzrinne et al. [SAD09], pages 1–10.

[LMV08] Matthieu Latapy, Clémence Magnien, and Guillaume Valadon. First report on database specification and access including content rating and fake detection system. Technical report, Measurement and Analysis of P2P Activity Against Paedophile Content Project, Laboratoire d'Informatique de Paris 6 - CNRS and Université Pierre et Marie Curie, 2008.

[LN07] Hogyun Lee and Taekyong Nam. P2p honeypot to prevent illegal or harmful contents from spreading in p2p network. In *The 9th International Conference on Advanced Communication Technology*, volume 3, pages 497 –501, feb. 2007.

[LNR05] Jian Liang, Naoum Naoumov, and Keith W. Ross. Efficient blacklisting and pollution-level estimation in p2p file-sharing systems. In Cho and Jacquet [CJ05], pages 1–21.

[LNR06] Jian Liang, Naoum Naoumov, and Keith W. Ross. The index poisoning attack in p2p file sharing systems. In *INFOCOM* [IEE06].

[LRW03] 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.

[LSM06] Brian Neil Levine, Clay Shields, and N. Boris Margolin. A Survey of Solutions to the Sybil Attack. Tech report 2006-052, University of Massachusetts Amherst, Amherst, MA, October 2006.

[LUKM05] Arnaud Legout, Guillaume Urvoy Keller, and Pietro Michiardi. Understanding BitTorrent : An Experimental Perspective. Technical Report, PLANETE - INRIA / Institut Eurecom - CNRS : FRE2660 - EURECOM, 2005.

[Mar02] Evangelos P. Markatos. Tracing a large-scale peer to peer system : An hour in the life of gnutella. *IEEE International Symposium on Cluster Computing and the Grid*, 0 :65, 2002.

[Mem08] Ghulam Memon. Characterizing traffic in widely-deployed dht. Technical Report CIS-TR-2009-01, Computer and Information Science Department, University of Oregon, 2008.

[MLlGG08] Clémence Magnien, Matthieu Latapy, Jean loup Guillaume, and Bénédicte Le Grand. First report on paedophile keywords observed in edonkey. Technical report, Lip 6 Cnrs and Université Pierre and Marie Curie, 2008.

[MM02] 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.

[MM03] Sergio Marti and Hector G. Molina. Identity crisis : Anonymity vs. reputation in P2P systems. In *3rd International Conference on Peer-to-Peer Computing (P2P'03)*, pages 134–141, 2003.

[MRGS09] 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.

[MS99] Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA, 1999.

[NR06] Naoum Naoumov and Keith Ross. Exploiting p2p systems for ddos attacks. In *InfoScale '06 : Proceedings of the 1st international conference on Scalable information systems*, page 47, New York, NY, USA, 2006. ACM.

[PKK08] Michael Piatek, Tadayoshi Kohno, and Arvind Krishnamurthy. Challenges and directions for monitoring p2p file sharing networks - or - why my printer received a dmca takedown notice. In Provos [Pro08].

[Pro08] Niels Provos, editor. *3rd USENIX Workshop on Hot Topics in Security, July 29, 2008, San Jose, CA, USA, Proceedings*. USENIX Association, 2008.

[qdllf06] Office québécois de la langue française. Entrée 'poste-à-poste' du grand dictionnaire terminologique, oqlf. `http://liendex.ptaff.ca/v2/fr_CA/francais/grand_dictionnaire_terminologique/poste-à-poste`, 2006.

[RD01] 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 :329–350, 2001.

[REMP07] Hosam Rowaihy, William Enck, Patrick McDaniel, and Tom La Porta. Limiting sybil attacks in structured p2p networks. In *INFOCOM* [IEE07], pages 2596–2600.

[RFH+01] 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.

[RIF02] Matei Ripeanu, Adriana Iamnitchi, and Ian Foster. Mapping the gnutella network. *IEEE Internet Computing*, 6 :50–57, 2002.

[SAD09] Henning Schulzrinne, Karl Aberer, and Anwitaman Datta, editors. *Proceedings P2P 2009, Ninth International Conference on Peer-to-Peer Computing, 9-11 September 2009, Seattle, Washington, USA*. IEEE, 2009.

[SBEN07] 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.

[SCB09] Moritz Steiner, Damiano Carra, and Ernst W Biersack. Evaluating and improving the content access in KAD. *Springer "Journal of Peer-to-Peer Networks and Applications", Vol 3 N°2, June*, 2009.

[SCDR04] Atul Singh, Miguel Castro, Peter Druschel, and Antony Rowstron. Defending against eclipse attacks on overlay networks. In *EW 11 : Proceedings of the 11th workshop on ACM SIGOPS European workshop*, page 21, New York, NY, USA, 2004. ACM.

[Sch05] Christian Scheideler. How to spread adversarial nodes ? : rotate ! In *STOC '05 : Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 704–713, New York, NY, USA, 2005. ACM.

[SEENB07] Moritz Steiner, Wolfgang Effelsberg, Taoufik En-Najjary, and Ernst W Biersack. Load reduction in the KAD peer-to-peer system. In *DBISP2P 2007, 5th International Workshop on Databases, Information Systems and Peer-to-Peer Computing, September, 24, 2007, Vienna, Austria*, 09 2007.

[SENB07a] Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack. Exploiting kad : possible uses and misuses. *SIGCOMM Comput. Commun. Rev.*, 37(5) :65–70, 2007.

[SENB07b] 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.

[SENB09] Moritz Steiner, Taoufik En-Najjary, and Ernst W Biersack. Long term study of peer behavior in the KAD DHT. *IEEE/ACM Transactions on Networking, Vol. 17, N°5, October 2009, ISSN :1063-6692*, 2009.

[SG07] 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.

[SL04] Mudhakar Srivatsa and Ling Liu. Vulnerabilities and security threats in structured overlay networks : A quantitative analysis. *Computer Security Applications Conference, Annual*, 0 :252–261, 2004.

[SM02] Emil Sit and Robert Morris. Security considerations for peer-to-peer distributed hash tables. In *IPTPS '01 : Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 261–269, London, UK, 2002. Springer-Verlag.

[SMK+01] 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.

[SNDW06] 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.

[SR94] R. R. Sokal and F. J. Rohlf. *Biometry : the principles and practice of statistics in biological research.* New York : Freeman, 3rd edition, 1994.

[SR06] Daniel Stutzbach and Reza Rejaie. Improving lookup performance over a widely-deployed dht. In *INFOCOM* [IEE06].

[Sri01] K. Sripanidkulchai. The popularity of Gnutella queries and its implications on scalability. Technical report, Carnegie Mellon University, February 2001. Available from `http://www.cs.cmu.edu/~kunwadee/research/p2p/gnutella.html`.

[SRRS09] Kyuyong Shin, Douglas S. Reeves, Injong Rhee, and Yoonki Song. WINNOWING : Protecting P2P Systems Against Pollution By Cooperative Index Filtering. Tech report TR-2009-2, Department of Computer Science, North Carolina State University, 2009.

[Ste08] Moritz Steiner. *Structures and algorithms for peer-to-peer cooperation.* PhD thesis, University of Nice, EURECOM, 12 2008.

[STR07] Xin Sun, Ruben Torres, and Sanjay Rao. Ddos attacks by subverting membership management in p2p systems. In *NPSEC '07 : Proceedings of the 2007 3rd IEEE Workshop on Secure Network Protocols*, pages 1–6, Washington, DC, USA, 2007. IEEE Computer Society.

[STR10] Xin Sun, Ruben Torres, and Sanjay Rao. Preventing ddos attacks on internet servers exploiting p2p systems. *Comput. Netw.*, 54 :2756–2774, October 2010.

[STY03] Nitesh Saxena, Gene Tsudik, and Jeong H. Yi. Admission control in Peer-to-Peer : Design and performance evaluation. In *1st ACM Workshop Security of Ad Hoc and Sensor Networks*, 2003.

[SW04] S. Sen and Jia Wang. Analyzing peer-to-peer traffic across large networks. *Networking, IEEE/ACM Transactions on*, 12(2) :219–232, April 2004.

[TCCF11a] Juan Pablo Timpanaro, Thibault Cholez, Isabelle Chrisment, and Olivier Festor. BitTorrent's Mainline DHT Security Assessment. In *4th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Paris France, 02 2011. IEEE. Projet GIS 3SGS ACDAP2P (Approche collaborative pour la détection d'attaques dans les réseaux pair à pair).

[TCCF11b] Juan Pablo Timpanaro, Thibault Cholez, Isabelle Chrisment, and Olivier Festor. When kad meets bittorrent - building a stronger p2p network. In *Eighth International Workshop on Hot Topics in Peer-to-Peer Systems - HotP2P 2011*, Anchorage, États-Unis, April 2011. IEEE International Parallel & Distributed Processing Symposium.

[Tor09] TorrentFreak. Bittorrent's future ? dht, pex and magnet links explained. `http://torrentfreak.com/bittorrents-future-dht-pex-and-magnet-links-explained-091120/`, 2009.

[Tor10] TorrentFreak. Bittorrent zeitgeist : What people searched for in 2010. `http://torrentfreak.com/bittorrent-zeitgeist-what-people-searched-for-in-2010-101227/`, 2010.

[TP10] Uzi Tuvian and Lital Porat. Hash collision attack vectors on the ed2k p2p network. Technical report, Interdisciplinary Center Herzliya, 2010.

[Tut04] Kurt Tutschku. A measurement-based traffic profile of the edonkey filesharing service. In Barakat and Pratt [BP04], pages 12–21.

[Tve77] Amos Tversky. Features of similarity. In *Psychological Review*, volume 84, pages 327–352, 1977.

[UPvS11] Guido Urdaneta, Guillaume Pierre, and Maarten van Steen. A survey of DHT security techniques. *ACM Computing Surveys*, 43(2), June 2011. `http://www.globule.org/publi/SDST_acmcs2009.html`.

[VV] Jochem Van Vroonhoven. Peer to peer security.

[Wal03] Dan S. Wallach. A survey of peer-to-peer security issues. In *ISSS'02 : Proceedings of the 2002 Mext-NSF-JSPS international conference on Software security*, pages 42–57, Berlin, Heidelberg, 2003. Springer-Verlag.

[WH10] Scott Wolchok and J. Alex Halderman. Crawling BitTorrent DHTs for fun and profit. In *Proc. 4th USENIX Workshop on Offensive Technologies (WOOT)*, August 2010.

[WTCT+08] 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.

[YFX+09] Jie Yu, Chengfang Fang, Jia Xu, Ee-Chien Chang, and Zhoujun Li. Id repetition in KAD. In Schulzrinne et al. [SAD09], pages 111–120.

[YGKX08] Haifeng Yu, Phillip B. Gibbons, Michael Kaminsky, and Feng Xiao. Sybillimit : A near-optimal social network defense against sybil attacks. In *SP '08 : Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pages 3–17, Washington, DC, USA, 2008. IEEE Computer Society.

[YKGF06] Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham Flaxman. Sybilguard : defending against sybil attacks via social networks. In *SIGCOMM '06 : Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 267–278, New York, NY, USA, 2006. ACM.

[YL09] Jie Yu and Zhoujun Li. Active measurement of routing table in kad. In *Proceedings of the 6th IEEE Conference on Consumer Communications and Networking Conference*, CCNC'09, pages 1252–1256, Piscataway, NJ, USA, 2009. IEEE Press.

[YMS+06] 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.

[ZDL07] Mo Zhou, Yafei Dai, and Xiaoming Li. A measurement study of the structured overlay network in p2p file-sharing systems. *Adv. MultiMedia*, 2007(1) :10–10, 2007.

# Abstract

**Monitoring of structured P2P networks applied to the security of contents**

The purpose of this thesis is to design and implement new monitoring solutions which are able to deal with the security issues affecting data within large structured P2P networks (DHT). There are two major types of issues. First, P2P networks are used to spread illegal contents whose access from users are difficult to monitor accurately. Second, regular contents can be removed from the P2P network or polluted if the indexation mechanism is corrupted (Sybil attack). All the proposed monitoring techniques are applied to the KAD P2P network which is currently one of the most widely deployed.

We first designed a new approach to monitor contents based on the insertion of distributed probes in the network to take control of the indexation mechanism. The probes can attract all the related requests for a given content with a high probability and assess the peers intent to access it by generating very attractive honeypots with few resources. We describe the weaknesses of the network allowing our solution to be effective despite recent protection mechanisms. We then present the services offered by our monitoring architecture and we evaluate its efficiency on KAD. We also present a real deployment whose purpose is to quantify the activity of contents related to paedophilia on this network.

In a second step, we focus on data integrity in distributed hash tables. We performed large scale monitoring campaigns on the KAD network. Our observations show that it suffers both from a new and very harmful form of pollution of its indexation mechanism affecting 2/3 of the shared files and also from a large number of localized attacks targeting contents. To mitigate these threats, we propose a new efficient way to detect attacks by analysing the distribution of the peers' ID found around an entry after a DHT lookup. We present our detection model and the a counter-measure which can protect the peers from attackers at a negligible cost. Finally, we evaluate our solution and implement it in real P2P networks.



**Keywords:** P2P networks, Distributed Hash Table, KAD, monitoring, honeypot, content indexation, security, Sybil attack, attack detection, defense, content pollution

## Résumé

**Supervision des réseaux pair à pair structurés appliquée à la sécurité des contenus**

L'objectif de cette thèse est de concevoir et d'appliquer de nouvelles méthodes de supervision capables d'appréhender les problèmes de sécurité affectant les données stockées dans les réseaux P2P structurés (DHT). Ceux-ci sont de deux types, d'une part les réseaux P2P sont utilisés pour diffuser des contenus illégaux dont il est difficile d'identifier précisément les accès, et d'autre part des contenus légitimes peuvent disparaître ou être pollués si le mécanisme d'indexation du réseau P2P est corrompu (attaque Sybil). L'ensemble des méthodes de supervision développées dans cette thèse sont appliquées au réseau P2P KAD qui est à ce jour l'un des plus déployé.

Dans un premier temps, nous proposons une nouvelle méthode de supervision des contenus basée sur l'insertion de sondes et le contrôle du mécanisme d'indexation du réseau. Celle-ci permet d'attirer avec une grande probabilité l'ensemble des requêtes des pairs pour un contenu donné, puis de vérifier leur intention en générant des appâts très attractifs avec peu de ressources. Nous décrivons ainsi les faiblesses du réseau permettant la mise en œuvre de notre méthode, en dépit des protections existantes. Nous présentons les fonctionnalités de l'architecture de supervision développée et en évaluons l'efficacité sur le réseau P2P KAD avant de présenter un déploiement réel ayant pour but l'étude des activités liées aux contenus pédophiles.

Dans un second temps, nous nous intéressons à la sécurité des données indexées dans une DHT. Nous supervisons le réseau KAD et montrons que celui-ci est victime, d'une part d'une nouvelle forme de pollution particulièrement néfaste appelée falsification de l'indexation et affectant 2/3 des fichiers populaires partagés, et d'autre part de nombreuses attaques ciblées qui affectent la sécurité des contenus stockés en son sein. Nous proposons un moyen de détecter efficacement cette dernière attaque en analysant la distribution des identifiants des pairs autour d'une référence recherchée. Nous présentons notre modèle de détection et la contre-mesure appliquée en cas d'attaque qui permettent de protéger un client des pairs suspects tout en présentant un coût d'application négligeable. Nous terminons par l'évaluation de ces protections et leur mise en œuvre dans des réseaux P2P réels.



**Mots-clés:** réseaux P2P, table de hachage distribuée, KAD, supervision, pots de miel, indexation des contenus, sécurité, attaque Sybil, détection d'attaques, défense, pollution des contenus