Supervision de trac au niveau applicatif : application à la sécurit é et à l'ingénierie des réseaux

Yannick Carlinet

N° **d'ordre : 4174** **ANNÉE 2010**

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

Ecole doctorale MATISSE

présentée par

Yannick Carlinet

préparée à l'unité de recherche SSIR (EA 4039)
Sécurité des Systèmes d'Information et Réseaux
Supélec

Supervision de trafic au niveau applicatif : application à la sécurité et à l'ingénierie des réseaux.

Thèse soutenue à Issy-Les-Moulineaux le 30 juin 2010

devant le jury composé de :

César VIHO
Professeur (Rennes 1) / président

Philippe OWEZARSKI
Chargé de recherches (CNRS) / rapporteur

André-Luc BEYLOT
Professeur (INPT/ENSEEIHT) / rapporteur

Yvon GOURHANT
Chercheur (Orange Labs) / examinateur

Ludovic ME
Professeur (Supélec) / directeur de thèse

Hervé DEBAR
Professeur (Télécom SudParis) / co-directeur de thèse

Résumé

Les travaux décrits dans ce mémoire portent sur la supervision du trafic dans le cœur de réseau, au niveau de la couche applicative. Nous illustrons l'intérêt de la supervision dans la couche 7 grâce à trois études qui montrent les bénéfices obtenus pour la sécurité et pour l'évaluation de modifications d'architecture du réseau.

La première étude utilise l'épidémiologie, la science qui étudie les causes et la propagation des maladies. L'épidémiologie fournit des concepts et des méthodes pour analyser à quels risques potentiels d'infection les PC des clients ADSL sont exposés. En particulier, nous voulons analyser les risques par rapport aux applications utilisées par les clients. Grâce à la supervision applicative du trafic d'un large échantillon de clients ADSL dans le cœur de réseau, nous construisons un profil d'utilisation du réseau pour chaque client et nous détectons ceux qui génèrent du trafic malveillant. A partir de ces données, nous étudions le lien entre certaines caractéristiques des clients avec leurs risques d'être infecté. Nous mettons en évidence deux applications et un système d'exploitation qui constituent des facteurs de risque. Nous en déduisons un profil de client pour lequel existe un risque important de se voir infecter par un virus informatique.

La deuxième étude porte sur l'opportunité pour un opérateur d'installer des caches P2P dans son réseau. Mettre les contenus P2P en cache pourrait être un bon moyen de réduire la charge dans le réseau. Cependant, les performances des caches sont affectées par de nombreux facteurs, en relation avec les caches eux-mêmes, mais également avec les propriétés de l'overlay, les contenus P2P et la localisation du cache. Dans le but d'évaluer l'utilité potentielle des caches P2P, nous effectuons la supervision à grande échelle du trafic P2P, dans le réseau opérationnel de France Télécom. Après avoir étudié certaines propriétés du trafic observé, nous simulons le fonctionnement d'un cache en nous basant sur les données recueillies pendant 10 mois. Nous sommes alors en mesure d'évaluer les performances d'un cache, en termes d'économie de bande passante si un cache était réellement déployé au moment de la capture de nos traces. De plus, nous étudions l'impact sur la performance de paramètres tels que les caractéristiques du cache et de manière plus importante le nombre de clients servis par le cache. Les résultats montrent que l'on pourrait réduire le trafic P2P d'échange de fichiers de 23% avec un cache passif.

Enfin, la troisème étude porte sur l'opportunité pour un opérateur de réseau de coopérer avec les réseaux P2P à travers une interface de type P4P.

L'approche P4P permet aux clients P2P d'améliorer leur sélection de pairs dans l'overlay. Le trafic P2P représente une proportion importante du volume du trafic dans les réseaux. Cependant, les systèmes P2P les plus couramment utilisés à l'heure actuelle ne tiennent pas compte de l'infrastructure réseau sous-jacente. Grâce à la supervision applicative, nous déterminons les bénéfices de P4P, d'une part pour les applications P2P et d'autre part pour les opérateurs. Les résultats de cette expérimentation indiquent que les applications P2P ont besoin de plus d'informations que seulement l'AS de provenance des sources potentielles pour améliorer leurs performances. De plus nous montrons que le trafic P2P inter-domaines pourrait être réduit d'au moins 72% grâce à P4P.

Nous montrons donc dans ces travaux que la supervision applicative permet :

- d'analyser des phénomènes complexes liés à l'usage qui est fait du réseau, tels que la contamination par un ver ou un virus informatique ;
- d'évaluer, de manière précise et quantitative, l'impact de certaines modifications d'architecture sur le trafic opérationnel.

D'une manière plus générale, nous illustrons l'importance du rôle de l'opérateur de réseau dans le déploiement et l'exploitation des services Internet, toujours plus gourmands en bande passante, que nous ne manquerons pas de voir apparaître à l'avenir. En effet la supervision applicative est un outil essentiel pour l'évaluation des protocoles et architectures mis en œuvre dans les services Internet, complémentaire des autres outils dans ce domaine.

Table des matières

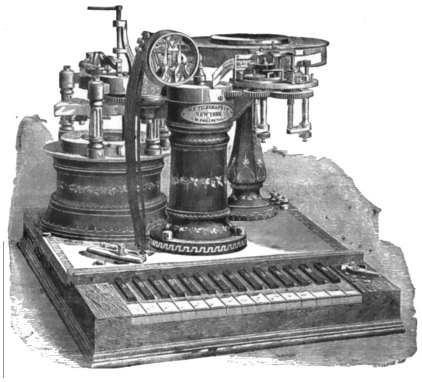
Table des figures

Liste des tableaux

Chapitre 1

Introduction

Le tout premier réseau de communication électrique commercial fut inauguré le 9 juillet 1839. Il s'agissait d'une ligne de télégraphe de 21 km, en Grande-Bretagne.



Le domaine général de cette thèse est l'observation et l'analyse du trafic au niveau applicatif dans les réseaux de données informatiques. Ce chapitre décrit le contexte général de la thèse (1.1), puis la problématique traitée (1.2).

1.1 Un peu d'histoire

Pour comprendre comment vont évoluer les réseaux à l'avenir, il est utile d'observer comment ils ont évolués par le passé. Nous allons nous intéresser dans cette section aux usages des réseaux commerciaux. Nous mettons en évidence les caractéristiques des services mis en œuvre sur ces réseaux, qui vont avoir un impact sur leur ingénierie, en particulier pour leur architecture et leur dimensionnement. En effet les services sont la raison d'être de l'exploitation commerciale des réseaux et l'évolution de leurs requis conditionne l'évolution du réseau.

L'internet au sens de réseaux fondés sur les protocoles TCP/IP remonte

au début des années 80, avec la standardisation de IP (Internet Protocol [1]), ICMP (Internet Control Message protocol [2]) un protocole pour la gestion des communications IP et TCP (Transmission Control Protocol [3]) en 1981. Ces protocoles furent adoptés comme les seuls protocoles utilisés sur ARPANET (le premier réseau à transfert de paquet, développé par la DARPA) en 1983. A cette époque, les principales applications réseau qui prédominent sont le courier électronique et les groupes de discussion. Ces applications nécessitent une bande passante extrêmement réduite et n'ont aucune contrainte de qualité de service.

En 1989, Tim Berners-Lee invente le «World Wide Web» (WWW), c'est-à-dire un client et un serveur communicant via le protocole HTTP (HyperText Transfer Protocol). En 1993, Mosaic, le premier navigateur web graphique, est implémenté. Le web devient rapidement et durablement, l'application réseau la plus utilisée. Remarquons que cette application nécessite une bande passante réduite, mais a cependant une contrainte de délai, en effet après plusieurs minutes à attendre le chargement d'une page, l'utilisateur s'impatiente et considère la qualité du service insuffisante.

En 1999, apparaît Napster [4], un logiciel d'échange de fichiers musicaux de type pair-à-pair (P2P). C'est le début d'une vague de popularité phénoménale pour les applications P2P. Napster fut fermé en 2001 par la justice des USA mais ce logiciel fut bien vite remplacé par d'autres qui ne nécessitent pas un serveur central unique pour fonctionner. Parmi les plus utilisés en France on peut citer Kazaa, eDonkey, puis BitTorrent. Les applications P2P vont participer au développement accéléré des accès résidentiels à large bande. Les applications d'échange de fichier en P2P sont très consommatrices de bande passante. En revanche elles n'ont aucune contrainte de qualité de service.

En 2004, les premières offres commerciales de VoIP (Voice over IP) voient le jour. Ces offres permettent aux souscripteurs d'un accès à large bande de téléphoner avec des coûts d'exploitation réduits pour le FAI (Fournisseur d'Accès à Internet). Ce type d'application ne requiert pas une grande bande passante, mais elle a des contraintes de qualité de service très fortes. En effet le service se dégrade beaucoup en cas de pertes de paquets, de délais ou de jigue importants.

Aujourd'hui, les offres commerciales des FAI intègrent des services tels que la vidéo à la demande (VoD), ou la télévision sur IP. De plus, la diffusion de contenus vidéos en streaming HTTP est maintenant la principale application réseau, en termes de quantité de données transportées par les FAI (selon un rapport d'audit de trafic interne à France Télécom). Toutes ces nouvelles applications sont très gourmandes en bande passante et en plus ont des contraintes fortes de qualité de service. Avec les requis en bande passante

toujours plus grands des applications (par exemple avec la démocratisation prochaine de la télévision et la VoD en haute définition), les réseaux actuels arrivent bien vite à saturation et leurs successeurs devront être conçus et dimensionnés en tenant compte de tous ces nouveaux usages.

Dans ce contexte l'observation du réseau et de l'utilisation qu'on en fait est cruciale pour concevoir le réseau de demain. Pour connaître précisément les services qui génèrent le plus de trafic et donc les usages les plus exigeants, il devient nécessaire d'effectuer la supervision du réseau au niveau de la couche applicative. Ce constat est détaillé dans la section suivante, qui expose la problématique traitée dans ces travaux.

1.2 Problématique

1.2.1 Supervision et ingénierie des réseaux

La supervision est une fonction essentielle dans toutes les activités liées à la gestion d'un réseau de données. Elle est nécessaire à de nombreux titres, en premier lieu elle sert à détecter les problèmes qui pourraient se produire sur le réseau de manière à permettre une intervention rapide de l'exploitant. Les problèmes pouvant affecter le fonctionnement du réseau sont par exemple des pannes ou des congestions de trafic. Elle est essentielle également pour le comptage, la mesure des performances et pour la sécurité du réseau (détection d'anomalies, d'attaques ou d'intrusions).

La supervision de réseau sert également pour comprendre l'usage qui est fait du réseau par les clients et donc par la suite pour anticiper les besoins en termes d'utilisation. De ces besoins futurs découlent la manière dont l'opérateur va faire évoluer son réseau. La supervision est également essentielle pour la modélisation du réseau, qui va déterminer par la suite son dimensionnement. De manière plus générale, nous voyons que la supervision est un élément très important de l'ingénierie des réseaux. Ce point est mis en évidence entre autres par Owezarski [23].

Les mesures sur le trafic peuvent être de type actives ou passives. Dans le cas actif, des paquets sont envoyés dans le réseau et on observe comment le réseau fonctionne suite à cet envoi. Un exemple très simple de mesure active est l'application ping, qui envoie un paquet ICMP et mesure le temps d'aller-retour de ce paquet. L'inconvénient principal des mesures actives est qu'elles modifient l'état du réseau que l'on cherche à observer.

Les mesures passives consistent à capturer une partie des paquets, ou à journaliser des informations sur le trafic dans un équipement du réseau, un routeur par exemple. Cette technique est non-intrusive car elle n'influence

pas le trafic qui est observé. Cependant ce type de mesure soulève des difficultés : observer le trafic a un coût en termes de traitements et de mémoire dans les équipements. Ainsi bien souvent on ne pourra pas mesurer tout le trafic à cause des coûts que cela entraine, il faudra donc mettre en œuvre des techniques d'échantillonage. De plus cette technique n'est applicable que par l'opérateur de réseau ou l'administrateur de domaine, car elle implique d'avoir accès aux équipements réseau.

La supervision de réseau passive peut s'effectuer sur le mode en ligne, ou bien hors ligne, sur des enregistrements de trafic. Dans la supervision en ligne, on ne peut pas réaliser une analyse très approfondie des paquets car l'analyse doit se faire en temps réel, mais l'avantage est que la mesure peut se faire sur une longue période de temps. A l'inverse, en mode hors-ligne la capture a nécessairement une durée limitée à cause de la capacité de stockage requise pour enregistrer une trace de trafic, mais on peut bien sûr réaliser des traitements sur le trafic sans contrainte de temps.

De plus, les observations peuvent s'effectuer à différents endroits dans la topologie du réseau. L'architecture des réseaux de données est composée du réseau d'accès et du réseau cœur. L'avantage d'effectuer la supervision dans le cœur est que l'on peut observer plus de clients que dans l'accès.

Il se pose également le problème du niveau, au sens de la couche du modèle OSI (Open Systems Interconnection [6]), auquel on va observer le trafic. Traditionnellement la supervision passive dans le cœur de réseau s'effectue au niveau 2 (couche lien) ou 3 (couche réseau), avec parfois des incursions dans la couche 4 (couche transport). Ainsi, par exemple, les mécanismes de supervision définis dans IPFIX [21], dont Netflow [22] est une implémentation, ne considèrent que les paquets IP qui transitent par un routeur, mais ils définissent une notion de flux. Ainsi ces mécanismes permettent de donner un aperçu sur les flux TCP, même si les entêtes TCP ne sont pas décodés. Un des obstacles à décoder les couches au delà de l'entête IP est le coût en performance que cela représente. En effet dans le cœur de réseau le trafic peut atteindre des débits rédhibitoires pour l'analyse du trafic en mode en ligne (e.g. routeurs avec des interfaces à plusieurs To/s [18]), si l'on n'effectue pas d'échantillonnage.

La supervision du trafic peut s'effectuer dans la couche 7 (couche application) dans le cœur de réseau mais l'échantillonage doit être encore plus grossier que lorsqu'on ne regarde que les entêtes IP.

Il existe en plus des limitations d'une autre nature à la supervision dans la couche 7. En effet, certains services sont soumis à des réglementations particulières, comme par exemple les couriers électroniques qui sont protégés par le secret de la correspondance, en application dans la plupart des pays. Même si un service particulier n'est pas soumis à de telles restrictions régle-

mentaires, l'éthique et le respect de la vie privée peuvent interdire de décoder le protocole de couche 7 du trafic sans précautions préalables. Le débat sur la neutralité du réseau (connu aussi sous l'appellation anglo-saxone «net neutrality») témoigne de l'importance de ces questions qui touchent aux droits des citoyens. Nous devons donc prendre toutes les précautions nécessaires avant d'effectuer le décodage d'un protocole applicatif.

1.2.2 Supervision dans la couche applicative

En dépit de ces limitations, nous allons nous attacher dans cette thèse à démontrer que ce type de supervision est possible et nous allons donner des exemples démontrant son utilité.

Tout d'abord, la supervision dans la couche 7 permet de beaucoup mieux comprendre l'usage que font les utilisateurs d'Internet, puisqu'on observe directement les applications et donc les services dont font usage les clients. Cette compréhension des usages est très importante pour l'ingénierie des réseaux. Dans la suite de ce mémoire nous allons donner trois cas d'étude où nous avons tiré parti de la supervision dans la couche 7 pour obtenir des informations d'une utilité certaine pour le FAI, qu'il aurait été impossible d'obtenir en se limitant aux couches inférieures. Les trois chapitres suivants présentent donc chacun un cas d'étude et servent d'illustration de l'importance de cette observation applicative.

Tout d'abord, nous allons montrer comment l'analyse des usages nous permet de déterminer les clients résidentiels les plus vulnérables aux virus et aux vers informatiques (chapitre 3).

Par ailleurs, la supervision du trafic issu de certains services, comme l'échange de fichiers en P2P, nous permet d'envisager des traitements particuliers à ce service, tels que la mise en cache (chapitre 4), pour le bénéfice à la fois des utilisateurs du service et de l'opérateur du réseau.

Enfin, nous allons montrer que la supervision dans la couche 7 est très utile pour la mise en œuvre de techniques, spécifiques à certains services, pour équilibrer la charge dans le réseau et optimiser les coûts d'exploitation en lien avec le trafic dans le réseau (chapitre 5).

En résumé, nous allons illustrer que la supervision dans la couche applicative peut apporter :

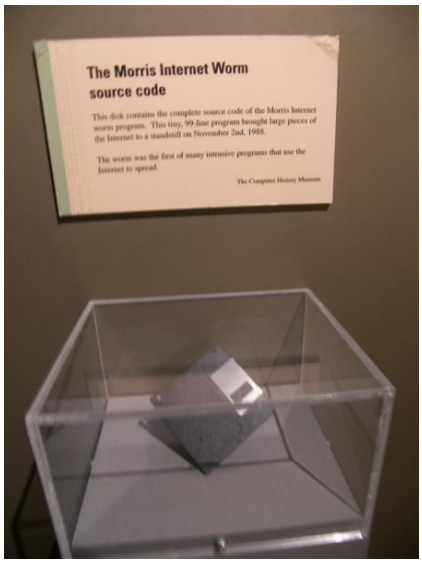
1. amélioration de la sécurité des utilisateurs de l'Internet en luttant contre les infections de machines ;
2. aide à l'ingénierie du réseau en déterminant l'opportunité de la mise en place de fonctions réseau spécialisées pour des services particuliers ;

3. optimisation de l'utilisation du réseau avec une gestion adaptée du trafic de certains services.

L'état de l'art du chapitre suivant va s'attacher à exposer les travaux effectués dans ces domaines, à savoir d'une part la détection du trafic malveillant dans le cœur de réseau, en particulier suite à la compromission des ordinateurs des utilisateurs de l'Internet. D'autre part nous allons exposer les travaux effectués dans le domaine de la reconnaissance des protocoles applicatifs, avec une attention particulière à la reconnaissance des applications P2P.

Chapitre 2

Etat de l'art



Disquette contenant le code source du ver «Morris», au musée des sciences de Boston.

Cet état de l'art concerne les domaines pour lesquels nous avons montré par des études, détaillées dans les chapitres suivants, qu'il y avait un intérêt à observer la couche applicative. Le chapitre 3 traite de l'observation des usages des utilisateurs afin d'étudier les causes de la compromission de leur ordinateur. Afin de réaliser cette étude nous avons été amenés à analyser le trafic du réseau opérationnel pour détecter le trafic malveillant. C'est pourquoi nous allons nous intéresser dans cet état de l'art à la detection du trafic malveillant à partir d'observations dans le réseau (section 2.1).

Les chapitres 4 et 5 se focalisent sur le trafic P2P, nous allons donc détailler dans cet état de l'art le problème de la reconnaissance des applications (section 2.2), avec un accent sur les applications P2P (2.2.3). Puis

nous détaillerons le décodage du protocole eDonkey (section 2.3) car c'est ce protocole que nous avons choisi pour réaliser notre expérimentation de supervision à grande échelle. Nous donnons également un bref aperçu de la mise en œuvre de cette expérimentation dans le réseau opérationnel, grâce aux sondes OTARIE (section 2.4).

2.1 Détection du trafic malveillant dans le réseau

Dans la suite nous définissons le trafic malveillant comme suit :

Définition 1 *Le trafic malveillant est le trafic qui provient des activités illégales ou contraire aux conditions d'utilisation des réseaux.*

Les logiciels malveillants constituent une menace omniprésente pour les utilisateurs de l'Internet, comme souligné dans le rapport de l'ENISA (European Network and Information Security Agency) [64] publié en novembre 2007, qui indiquait que plus de six millions d'ordinateurs dans le monde étaient compromis par des logiciels malveillants. Aujourd'hui ce nombre est certainement bien supérieur car d'importants intérêts économiques se cachent derrière le contrôle de machines à l'insu de leur propriétaire. En effet, il a été observé par Franklin et al. [38] que les machines compromises sont généralement utilisées pour des opérations rémunératrices et malveillantes, telles que du chantage au DDoS (Distributed Denial of Service) envers des entreprises, ou l'envoi massif de pourriels (spams) par exemple. Déja en 2005, une étude a déterminé que la perte dûe aux attaques par le réseau était de 130 millions de dollars pour l'année, aux USA [17].

La sécurité d'une chaîne d'information est aussi sûre que son maillon le plus faible. Or avec la démocratisation des accès large-bande, de nombreux utilisateurs de l'Internet gardent leur ordinateur connecté en permanence au réseau, sans pour autant prendre les précautions nécessaires pour assurer la sécurité de leur système informatique. Ce manque de précautions peut être dû à une certaine négligence ou tout simplement à des compétences informatiques insuffisantes. Parmi la vaste étendue des usages frauduleux, malveillants, ou simplement indésirables de l'Internet, certains sont plus facilement détectables sur les machines hôtes. C'est le cas par exemple des attaques par ingénierie sociale ou bien des virus par exemple. Nous n'aborderons pas ces attaques car les travaux présentés dans ce document se focalisent uniquement sur la détection dans le réseau.

Le trafic malveillant que l'on peut observer dans le réseau peut provenir de deux sources de natures différentes. D'une part il peut être le fait d'un pirate informatique qui cherche à exercer une activité frauduleuse ou contraire à la politique d'utilisation des réseaux. D'autre part, ce trafic peut provenir de l'ordinateur d'un utilisateur de l'Internet honnête et respectueux des lois, mais dont la machine serait contrôlée à son insu. Ce dernier type de trafic est très certainement le plus important, car les pirates vont chercher bien entendu à envoyer le moins possible de trafic malveillant depuis leur propre machine, pour la raison évidente que cela risque de les trahir et par la suite de les exposer à des poursuites judiciaires.

L'observation du trafic dans le réseau permet de détecter certains types d'attaques qui passeraient inaperçues au niveau des machines hôtes. C'est le cas des types d'attaque énumérés ci-dessous :

- propagation de vers ;
- attaques de déni de service distribuées ;
- spams ;
- botnets.

Ces exemples de trafic malveillant sont détaillés dans la sous-section suivante.

2.1.1 Quelques exemples de trafic malveillant

Nous donnons dans cette sous-section un aperçu de la diversité des types de trafic malveillant qui transite dans les réseaux de cœur.

Pour se propager, les vers doivent chercher des machines vulnérables sur le réseau. Puis lorsqu'une telle machine a été trouvée, le ver exploite la faille et se télécharge sur la machine. Toutes ces actions génèrent une activité réseau qui peut donner lieu à la détection du ver. De la même manière les attaques de déni de service distribuées génèrent un trafic réduit sur les machines hôtes mais le nombre de ces dernières rend le trafic vers la victime beaucoup plus facile à détecter. Les spams et les botnets également génèrent une activité réseau qui peut être détectée. Le principal avantage de la détection dans le réseau par rapport à sur la machine hôte est que l'on a une vision d'ensemble qui permet d'observer des comportements synchronisés, ou au moins coordonnés, de plusieurs machines sur le réseau. Ces quatres types de violations des politiques de sécurité sont responsables de l'essentiel du trafic malveillant circulant dans les réseaux à l'heure actuelle. Nous allons donc les détailler dans cette sous-section.

Propagation des vers

Un ver est un programme qui se reproduit par le réseau en exploitant des failles de sécurité dans les machines connectées au réseau.

Il faut distinguer les vers des virus, car un virus est un programme ajouté à un logiciel légitime (appelé programme hôte selon l'analogie biologique), de manière à s'exécuter lorsque le programme hôte est lancé. Les vers sont pourvus de trois fonctions principales, une fonction de découverte de machines où il pourra potentiellement se dupliquer (e.g. scan d'adresses IP), une fonction d'exploitation de failles pour se reproduire (e.g. une faille dans un service réseau) et une fonction d'action malveillante une fois qu'une nouvelle victime est infectée (e.g. installation d'un cheval de Troie).

Göldi et Hiestand [51] présentent une taxonomie des vers. Khiat et al. [37] mettent en évidence l'émergence de vers qui tirent parti des applications des utilisateurs pour se propager de manière furtive. Franklin et al. [38] montrent que les vers sont un outil de plus en plus utilisé par le crime organisé sur Internet.

Les attaques de déni de service distribuées

Un DDoS (Distributed Denial of Service) est une attaque qui consiste à perturber le service offert par une machine sur un réseau en lui envoyant des données depuis de multiples machines attaquantes.

Mirkovic et Reiher ont présenté dans [61] plusieurs taxonomies pour les attaques par DDoS. Ils proposent de classer les attaques selon différents critères : le degré d'automatisme (automatique, semi-automatique, manuel), la faille exploitée (exploit sémantique, inondation), la validité de l'adresse source (valide ou usurpée), le débit du trafic d'inondation (constant ou variable), le type de victime (application, machine hôte, resource, réseau, infrastructure) et enfin l'impact sur la victime (dégradation du service ou déni de service).

Moore et al. [62] ont réussi à quantifier la prévalence des attaques DDoS dans l'Internet par une technique très astucieuse, appelée «backscatter analysis». Cette technique permet d'évaluer quantativement le nombre d'attaques par DDoS dans le monde entier. Cela consiste à observer les réponses des victimes des attaques par DDoS avec des paquets IP dont l'adresse est usurpée. Lorsque l'adresse source des paquets d'attaque est choisie aléatoirement par les outils d'attaque, on va pouvoir observer des réponses des victimes vers des adresses aléatoires et donc en particulier vers des plages d'adresses inutilisées. On peut donc ainsi distinguer les réponses dans le trafic total d'une trace. Les auteurs ont mis en évidence 68 000 ataques de DDoS vers 34 000 adresses distinctes, entre 2001 et 2004.

Enfin, Zhou et al. [63] ont réalisé un état de l'art des systèmes distribués de détection d'attaques. Les auteurs constatent qu'il est plus facile de détecter une attaque distribuée en supervisant plusieurs points du réseau plutôt qu'en se limitant à un seul IDS (Intrusion Detection System). En effet une attaque peut passer inaperçue proche d'une machine attaquante car le trafic est réduit. En revanche l'attaque est facilement détectable proche de la victime, car c'est là que se concentrent tous les trafics d'attaque.

Les spams

Un spam (ou pourriel) est un couriel non sollicité. Il existe deux catégories de systèmes pour la détection des spams :

- les filtres sur le contenu, essentiellement mis en œuvre au niveau des serveurs de messagerie ou même des postes clients ;
- les systèmes de réputation de l'envoyeur, qui peuvent être appliqués sur les serveurs ou dans le réseau.

Les systèmes de réputation déployés actuellement sont de type DNSBL (DNS Black List), où une liste noire des adresses des envoyeurs de spams est tenue à jour. Spamhaus [32] et Spamcop [33] sont des exemples d'un tel système.

Les botnets

Un botnet est un ensemble de machines compromises (appelées bots) sous le contrôle d'un pirate, à l'insu de leurs propriétaires respectifs. Ces machines sont ensuite utilisées de manière coordonnée pour lancer des attaques de type DDoS, ou bien pour envoyer des spams par exemple [65].

Les botnets peuvent avoir une architecture centralisée ou bien distribuée. Dans le mode centralisé une seule machine (appelée botmaster) contrôlée par l'attaquant sert à envoyer les commandes qui seront exécutées par les bots. A l'inverse dans le mode distribué, les ordres du botmaster sont répercutés de bots en bots par des moyens de communication de type pair-à-pair.

Le canal de communication entre le botmaster et les bots est appelé canal de Commande et Contrôle (canal C&C). Le canal C&C peut utiliser une infrastructure publique, telle qu'un canal d'un serveur de discussion IRC (Internet Relay Chat) par exemple, ou bien utiliser un protocole de communication privé, c'est-à-dire utilisé uniquement par ce botnet.

Les communications au travers d'un serveur IRC restent le cas le plus fréquent c'est pourquoi beaucoup de méthodes de détection reposent sur l'analyse du trafic IRC. Cependant il existe d'autres canaux C&C. Des bots tels que PhatBot utilisent ainsi un protocole de communication pair-à-pair

nommé WASTE [66]. Ce protocole permet de créer un réseau overlay pour des communications anonymes, chiffrées et pair-à-pair, donc très difficile à détecter. Cependant d'après [67] les canaux IRC sont les plus communément employés.

En effet dans [67], Rajab et al. cherchent à observer le fonctionnement de botnets et à estimer la prévalence de ces botnets dans le trafic global. L'étude s'est déroulée en trois phases : une phase de récolte de binaires de bots, une phase d'analyse de ces binaires et enfin une phase d'observation de botnets au moyen de bots espions. La récolte de binaire fut effectuée au moyen d'un pot de miel (honeypot) mettant en œuvre la plate-forme Nepenthes [68] sur le réseau PlanetLab [69]. Ensuite ces binaires furent analysés de manière à pouvoir observer les tentatives de connexion du binaire à un serveur IRC. Puis ils ont observé les réponses du binaire de bot aux requêtes du serveur IRC (mis en place exprès pour ce binaire). Forts de ces observations, les auteurs ont conçu un bot qui puisse se connecter réellement à un botnet et se conformer au comportement attendu d'un bot par le botmaster. Ils ont pu ainsi observer plus de 100 botnets sur une période de 3 mois. Les conclusions de l'étude indiquent que le trafic lié aux botnets constitue plus de 27% du trafic global et que 11% des serveurs DNS sont utilisés par des bots pour résoudre le nom d'un serveur IRC pour se connecter au canal C&C.

Nous voyons donc que le trafic malveillant peut prendre de nombreuses formes et reste un sujet d'étude d'actualité. L'étude des différentes formes des trafics d'attaque est utile afin d'aider à la conception de systèmes de détection plus performants et efficaces. Nous allons maintenant nous intéresser au système de détection que nous avons mis en œuvre dans le chapitre 3, à savoir Snort [8].

2.1.2 Snort

Snort est un NIDS (Network Intrusion Detection System) libre, qui bénéficie d'un déploiement très large, avec 3,7 millions de téléchargements à ce jour, selon ses dévelopeurs [9].

Snort effectue une détection du trafic malveillant par signature, même si le logiciel inclut des modules qui implémentent des mécanismes de détection par anomalie. Les systèmes de détection par signature recherchent dans un paquet ou un flux une suite d'octets caractéristiques de trafic malveillant. Les signatures sont stockées dans une base de données appelée «ensemble de règles» (ou «ruleset»).

Le fonctionnement de Snort consiste à lire les paquets de l'interface réseau, puis les pré-processeurs sont exécutés et enfin le moteur de détection par règles est appelé. Voyons maintenant plus en détail les mécanismes mis en

œuvre à chaque étape.

Les pré-processeurs Snort

Les pré-processeurs servent à effectuer des traitements avant l'éxécution des règles de détection. Ainsi par exemple le pré-processeur *frag3* réalise la défragmentation des paquets IP, de manière à empêcher que des paquets d'attaque fragmentés intentionnellement puissent échapper à la détection.

Le pré-processeur *stream5* reconstitue les flux TCP. Il est également capable de reconstruire les sessions UDP. Il permet que des règles soient exécutée sur le flux de données. Ce module peut réaliser une détection par anomalies et lancer une alerte en cas de trafic TCP ou UDP anormal.

Le pré-processeur *sfPortscan* vise à détecter les balayages automatiques de ports (portscans) effectués par un ver lors de sa phase de reconnaissance de victimes potentielles.

Le pré-processeur *performance* sert à superviser les performances de l'application Snort en temps réel.

Le pré-processeur *ARP spoof* décode les paquets ARP (Address Resolution Protocol) et détecte les attaques ARP et les incohérences dans les correspondances entre les adresses MAC (Media Access Control) et les adresses IP.

Les pré-processeurs servent également à inspecter la charge utile des paquets et décoder certains protocoles applicatifs. Ainsi, le module *HTTP inspect* décode les différents champs de l'entête HTTP. Il est utile pour détecter des tentatives d'exploitation de vulnérabilités de serveurs web. Le module *SMTP* est capable de décoder les échanges SMTP (Simple Mail Transfer Protocol). Il existe également des pré-processeurs pour les protocoles de la couche applicative suivants : FTP, Telnet, SSH, SSL/TLS, DNS.

Le moteur de détection par règles

Les règles sont écrites dans un langage spécifique à Snort. Elles sont composées d'un entête et d'options. L'entête contient l'action de la règle, les adresses source et destination avec leur masque de réseau et les numéros de port source et destination. Les options contiennent les messages d'alerte et des indications sur quelles parties du paquet doivent être inspectées.

Les options des règles sont le cœur du moteur de détection. Elles permettent tout d'abord de définir les caractéristiques de l'alerte : message pour l'administrateur, niveau, priorité, identifiant de l'alerte. Ensuite elles permettent de définir des chaînes de caractères à chercher dans la charge utile du paquet. Les options peuvent définir le nombre d'octets à inspecter, ainsi

que l'offset pour le début de l'inspection. Certaines options sont spécifiques à des protocoles applicatifs et doivent donc être utilisées en conjonction avec le pré-processeur correspondant. Par exemple l'option *urilen* permet de spécifier une longueur de l'URI (Uniform Resource Identifier) dans une requête HTTP. Bien entendu les options des règles peuvent porter sur les champs des entêtes IP, UDP ou TCP.

Considérons l'exemple de la règle suivante :

```
alert tcp any any -> 192.168.1.0/24 111 \
(content:"|00 01 86 a5|"; msg:"mountd access";)
```

Le premier mot-clé *alert* indique l'action à effectuer, ici il faut lancer une alerte. D'autres types d'action peuvent être *log* ou *pass* par exemple. Le deuxième mot-clé indique le protocole qui peut être l'un parmi les quatre protocoles supportés par Snort : TCP, UDP, ICMP et IP. Ensuite les deux *any* indiquent que toutes les adresses et tous les ports source sont concernés par la règle. Puis viennent l'adresse et le port destination. Ensuite entre parenthèses viennent les options de la règle. Ici nous avons deux options. La première (*content*) donne une suite de caractères hexadécimaux à rechercher et la deuxième option (*msg*) définit le message d'alerte.

Dans le chapitre 3 nous avons utilisé Snort pour détecter le trafic malveillant dans nos traces de trafic. Notre choix s'est porté sur cet outil car il dispose d'un ensemble de règles très riche et il est disponible librement.

2.2 Reconnaissance des protocoles applicatifs

La reconnaissance des protocoles applicatifs est utilisée par les administrateurs de réseau d'entreprises ou de campus afin d'appliquer les politiques d'usage du réseau. Ainsi, par exemple, certains types d'usages sont restreints ou prohibés en environnement professionel, tels que les jeux ou certains contenus vidéos. D'autre part les administrateurs veulent identifier le trafic malveillant, c'est-à-dire correspondant à des activités ou des individus non autorisés. Bien entendu, la reconnaissance des protocoles de couche 7 sert également pour la supervision de réseau, comme expliqué dans la section 1.2.

Pendant longtemps, la reconnaissance s'est effectuée à partir du numéro de port de l'entête TCP. En effet à chaque numéro de port est associé un service par l'IANA (Internet Assigned Numbers Authority [16]), l'organisme international chargé de gérer la standardisation de ces numéros. Cependant de nombreuses applications négocient le numéro de port à l'établissement de la communication. De plus, certaines applications utilisent délibérément un

numéro de port attribué à un autre service. Par exemple le port 80, correspondant au service WWW (World Wide Web), est souvent utilisé indûment car les dévelopeurs des applications savent que les pare-feux laissent généralement passer le trafic sur ce port.

La reconnaissance des protocoles s'effectue selon deux grandes catégories de techniques : la recherche de signatures dans la charge utile (i.e. payload) des paquets IP (sous-section 2.2.1) et la classification des flux dans des grandes catégories d'applications selon certaines statistiques sur les paramètres des flux (sous-section 2.2.2)

Dans le domaine de la reconnaissance des applications, une attention particulière a été portée sur les applications P2P. La sous-section 2.2.3 est donc consacrée aux travaux de recherche effectués sur la reconnaissance des applications P2P.

2.2.1 Signature dans la charge utile

Cette technique consiste à observer les premiers octets de la charge utile des paquets IP. Chaque protocole applicatif définit en principe un format d'entête caractéristique, qui permet de le reconnaître. Ainsi, par exemple, on peut reconnaître un entête d'un message eMule car ce dernier est constitué de trois champs caractéristiques comme illustré sur la figure 2.1.

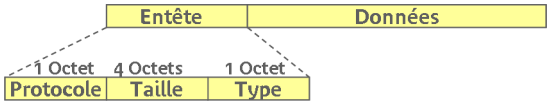


FIGURE 2.1 – Exemple d'analyse de l'entête d'un message eDonkey (couche 7)

Le premier champ est toujours à la valeur 0xe3 pour eDonkey, ou bien 0xc5 pour eMule (qui est une extension compatible de eDonkey). Si on observe un paquet dont les 6 premiers octets correspondent à la structure montrée dans la figure, alors nous pouvons avoir la quasi-certitude qu'il s'agit d'un paquet eMule.

Il faut donc définir des heuristiques pour chaque protocole applicatif que l'on souhaite reconnaître.

Cette technique permet d'obtenir un faible taux de mauvaise classification. Cependant les limitations de cette technique sont :

- un coût élevé de traitement, en effet il ne sera pas possible d'appliquer une analyse de la charge utile sur tous les paquets dans le cœur de réseau, car les performances du réseau en seraient trop dégradées ;

- on ne pourra pas reconnaître une application qui chiffrerait ses échanges, car la charge utile des paquets est alors inaccessible ;
- cela requiert un coût élevé en dévelopements car il faut suivre la grande diversité des protocoles et des logiciels et leurs évolutions ;
- selon les législations en vigeur, des contraintes légales peuvent empêcher d'effectuer une analyse syntaxique de la charge utile des paquets si certaines conditions ne sont pas remplies.

A cause de ces limitations, d'autres mécanismes furent étudiés, comme détaillé ci-dessous.

2.2.2 Classification des flux

La classification de flux par des méthodes statistiques consiste à observer les statistiques de paramètres en relation avec le comportement des applications, comme par exemple la taille, le sens, ou le temps inter-arrivée des paquets d'un flux TCP. Ensuite chaque flux peut être classé dans une catégorie qui contient tous les flux avec des caractéristiques similaires.

Ainsi, dans [84], McGregor et al. observent des caractéristiques des flux telles que la durée inter-arrivée des paquets, la taille, la durée du flux, etc... Puis ils appliquent un algorithme de clustering sur chaque flux afin de les regrouper en un petit nombre de classes. L'algorithme mis en œuvre ici est EM (Expectation-Maximisation) et des techniques d'apprentissage sont appliquées afin d'effectuer la classification. Cependant les auteurs reconnaissent l'une des principales limitations des résultats de cette étude : il est difficile d'associer les grandes classes de flux obtenues en sortie de l'algorithme avec des classes d'application réseau, ce qui limite considérablement l'interprétation des résultats obtenus.

Une autre approche, décrite par Roughan et al. dans [85], consiste à définir les classes à l'avance. Les auteurs définissent donc quatre classes d'applications selon les besoins en QoS (Quality of Service) de chacune. Ensuite les auteurs cherchent à déterminer les paramètres qui vont servir à classer chaque flux dans l'une ou l'autre des quatre classes. Ils constatent que les paramètres les plus utiles parmi ceux qui sont faciles à obtenir sont la taille des paquets et la durée de flux. Grâce à l'observation de ces données, les auteurs ont pu tester deux algorithmes de classification, à savoir NN (Nearest Neighbor) et LDA (Linear Discriminant Analysis). Une des limitations de cette approche est que l'on doit attendre la fin d'un flux pour le classifier, ce qui est utile pour effectuer du comptage, mais cela n'est pas adapté pour appliquer des politiques aux flux selon leur classe.

Par la suite, Zuev et Moore [86] approfondissent cette approche en définissant un plus grand nombre de classes d'application, de manière à obtenir

des résultats plus utiles. Ils mettent également en œuvre des algorithmes plus sophistiqués, basés sur des techniques d'apprentissage et un classifieur naïf de Bayes. La classification s'effectue au niveau de la couche 4 (entête TCP) mais l'apprentissage nécessite une trace complète puisqu'il s'effectue au niveau de la couche 7.

Karagiannis et al. [87] adoptent une approche fondamentalement différente puisqu'ils observent les flux selon trois critères : «social», «fonctionnel» et «application». Le critère «social» consiste à observer avec quelles machines chaque hôte établit des communications. Le critère «fonctionel» définit si l'hôte agit plutôt en fournisseur ou en consommateur de resources, i.e. s'il est plutôt client ou plutôt serveur. Enfin le critère «application» consiste à observer les caractéristiques de niveau 4 des flux. Cette approche complète donc les paramètres d'observation afin d'affiner la classification.

Wright et al. [89] constatent que les flux chiffrés ne peuvent pas être classifiés par les techniques décrites précédemment. Ils proposent donc une méthode fonctionnant avec le trafic chiffré. Les paramètres pris en compte sont la taille des paquets, leur horodatage et leur sens de circulation.

Un des problèmes des approches décrites précédemment est que l'on doit attendre la fin du flux pour pouvoir le classifier, puisque la durée du flux est un des paramètres pris en compte. Bernaille et al. [88] surpassent cette limitation en ne regardant que la taille et le sens des premiers paquets dans un flux avant d'effectuer la classification de ce flux. Ils déterminent que les quatre premiers paquets d'une connexion TCP suffisent à discriminer les flux. Les techniques de classification employées sont la moyenne K (K-means), le modèle de mélange gaussien (Gaussian Mixture Model) et le clustering spectral. Un avantage certain de cette approche est qu'elle peut fonctionner en mode en ligne. Par contre, il existe des limitations : si des paquets sont perdus, ou arrivent dans le désordre, le système ne va pas pouvoir reconnaître l'application. De même, si l'application réalise un brouillage, avec du bourrage de taille aléatoire dans les paquets par exemple, elle pourra passer inaperçue. Enfin, pour être reconnue, une application doit avoir été vue lors de l'apprentissage du système.

Récemment, Finamore et al. ont publié des travaux sur une nouvelle technique pour reconnaître l'application à partir du trafic UDP [93], puis par la suite TCP [94]. Les auteurs observent les adresses et numéros de port source et destination, ainsi que les premiers octets de la charge utile des paquets. La figure 2.2, extraite de [93], donne un aperçu du principe de fonctionnement du système.

Les signatures des paquets sont extraites par une méthode inspirée du test du χ^2. Ensuite pour chaque application, un modèle est créé grâce à la signature χ^2 des paquets de cette application. Selon les auteurs, les résultats

sont étonnamment bons, avec 99,6 % du trafic qui est correctement classifié.

Szabo et al. soulignent dans [90] un problème récurrent dans le domaine de la classification : l'évaluation pertinente des solutions techniques proposées. Bien souvent les évaluations sont effectuées à partir de traces dont on ne sait pas exactement ce qu'elles contiennent, ni si elles sont représentatives du trafic réel. En l'absence de «ground truth» (la véritables nature des paquets ou flux que l'on cherche à classifier), les solutions techniques sont évaluées à partir d'autres techniques de classification, ou bien avec la même trace qui a servi à l'apprentissage. Dans ces cas de figure, on ne sait pas exactement ce que les résultats d'évaluation signifient. Les auteurs proposent donc une méthode de validation d'un classifieur, fondée sur une trace générée, de manière à connaître parfaitement son contenu. La figure 2.3 illustre un exemple de résultat d'évaluation d'un classifieur.

Devant la prolifération vertigineuse des études et des publications sur le sujet de la classification des flux, Kim et al. [91] ont revisité les trois principales approches dans le domaine. Ils ont évalué chacune sur les mêmes traces, de manière à obtenir des résultats que l'on puisse comparer entre les différentes approches. Les trois approches principales en question sont : les approches fondées sur l'observation des numéros de port de la couche transport (couche 4), celles fondées sur le comportement des hôtes et celles fondées sur les caractéristiques des flux. Les résultats indiquent que les numéros de port peuvent permettre de correctement classifier les flux d'un certain nombre de protocoles, cependant il est évident qu'une application ne souhaitant pas être détectée ne pourra pas être reconnue par cette technique. L'observation du comportement des machines, telle que mise en œuvre dans Blinc [87] ou Kiss [93, 94] par exemple, est efficace lorsque le point d'observation est bien situé dans la topologie du réseau, mais ce n'est pas toujours le cas. En effet, dans le cœur de réseau, on ne voit pas forcément toutes les connexions d'un hôte particulier, de plus il est possible que l'on ne puisse observer les paquets que

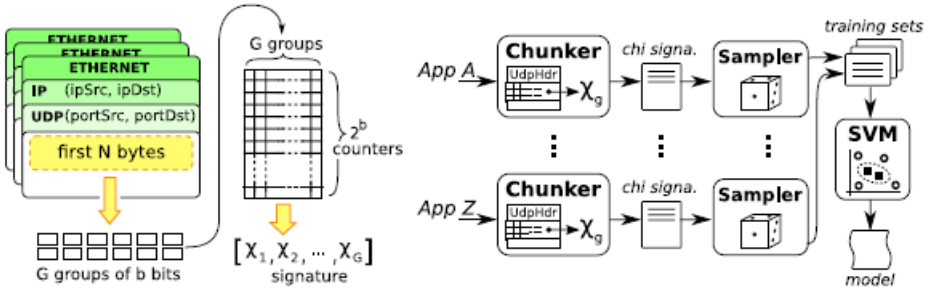


FIGURE 2.2 – Principe de fonctionnement de KISS [93, 94]

dans un sens, ce qui va nécessairement diminuer l'efficacité de la technique de classification.

Dans le domaine de la reconnaissance de protocoles applicatifs, une attention particulière a été consacrée, aussi bien par le monde de la recherche que par le monde industriel, aux protocoles P2P. Nous détaillons les travaux dans ce domaine spécifique dans la sous-section suivante.

2.2.3 Reconnaissance du trafic P2P

L'intérêt suscité pour la reconnaissance du trafic P2P est dû à de multiples facteurs. Tout d'abord, depuis presque 10 ans maintenant, ce type de trafic représente un volume très important de données à transporter dans les réseaux des FAI (cf. Labovitz et al. [18]). Ces derniers ont donc besoin de superviser ce type de trafic afin d'adapter l'ingénierie du réseau en conséquence et afin d'observer son évolution. Par ailleurs, certains FAI peu scrupuleux ont officieusement limité le débit de ce type d'applications dans leur réseau. De plus, historiquement, les applications P2P d'échange de fichiers ont beaucoup servis à l'échange de fichiers non-libres de droits. Ces raisons ont entrainé l'apparition de mécanismes de brouillage (i.e. obfuscation) dans certains logiciels succeptibles d'être la victime de limitations de trafic, tels que Skype ou eMule par exemple. La tâche de supervision de ce type de trafic est cruciale pour les FAI, à cause des quantités importantes de données à transporter, mais elle est rendue ardue par les techniques de brouillage ou de chiffrement du trafic mis en œuvre par les applications.

L'étude de Sen et al. [120] présente une méthode pour identifier 5 protocoles P2P à partir de signatures dans la payload (Gnutella, eDonkey, DirectConnect, BitTorrent, Kazaa). La validation de la technique fut effectuée sur deux traces : une prise dans le réseau d'accès (24+18H, 120 Go), l'autre dans un sous-réseau d'entreprise (6 jours, 1,8 To). Les auteurs ont vérifié les faux positifs en appliquant leur méthode sur la trace de réseau d'entreprise qui ne contient pas ou peu de trafic P2P à cause du règlement de l'entreprise, les pare-feux et la supervision active pour bloquer ce type de trafic. Le résultat de la classification donna peu de positifs, qui furent vérifiés manuellement. La vérification des faux négatifs fut effectuée en appliquant la méthode sur la trace provenant du réseau d'accès. Ils postulent que tout le trafic à partir des ports standards des applications P2P est du P2P. Ils appliquent donc leur solution sur ce trafic pour vérifier que tout est bien identifié. La méthode a un faible taux de faux positifs grâce à des critères suffisamment précis de classification. La validation indique que 98% de la classification est effectuée avec les quatre premiers paquets d'un flux. En revanche la technique ne fonctionne pas sur les protocoles inconnus (ou modifiés), ni avec du chiffrage des

paquets ou du brouillage.

Ohzahata et al. cherchent dans [95] à identifier les protocole P2P brouillés et chiffrés. L'article présente une méthode qui permet d'obtenir les adresses IP et les ports de service des pairs du réseau overlay. On peut ensuite identifier le trafic lié au protocole considéré à partir de ces informations. La méthode présentée fut mise en œuvre avec le protocole P2P Winny [96] mais elle peut facilement être appliquée avec un autre protocole P2P. La technique repose sur un faux pair qui se connecte au réseau overlay et enregistre les informations des pairs du réseau. Le faux pair se connecte et se déconnecte à différents points du réseau à intervalles de temps réguliers. Les avantages de l'idée présentée sont qu'il y a peu de faux positifs et que cela fonctionne avec un protocole distribué, chiffré, brouillé et anonyme (à la freenet [97]). Les inconvénients sont que le taux de faux négatifs est difficile à évaluer, car on ne voit pas forcément tout le réseau. D'autre part cela ne fonctionne pas pour des traitements en ligne car il faut du temps avant d'identifier les pairs du réseau. Ce sont des travaux intéressants qui relèvent un pari difficile, car Winny est certainement un des protocoles les plus ardus à détecter dans le réseau.

L'étude de Perenyi et al. [122] présente une méthode d'identification du trafic P2P dans le réseau sans regarder la charge utile des paquets. La méthode repose sur les 6 heuristiques suivantes :

1. une machine ouvre plusieurs connexions TCP et UDP en parallèle ;
2. l'hôte ouvre des connexions TCP de manière consécutive (par opposition à un serveur web qui ouvre plusieurs connexions en parallèle) ;
3. la machine utilise un port standard d'une application P2P ;
4. une machine génère plusieurs connexions au sein d'un même flux ;
5. une machine utilise le même port plus de 5 fois ;
6. le flux est de taille supérieure à 1 Mo ou bien dure plus de 10 mn.

L'article présente également une étude de trafic réel qui met en œuvre les heuristiques présentées afin de comparer les caractéristiques du trafic P2P et non-P2P. La comparaison n'a pas montré de différence significative entre les caractéristiques observées du trafic P2P et non-P2P (taille paquets, durée des flux). La validation de l'approche fut effectuée sur une trace de trafic générée par les auteurs. La trace est générée en faisant tourner des serveurs web et SMTP et des applications P2P. Avoir une trace générée permet de connaître les faux positifs et les faux négatifs de l'approche testée. Les auteurs ont vérifié que l'on pouvait différencier le trafic P2P et SMTP/web mais on ne sait pas ce qu'il en est des autres types de trafic. Ces travaux ouvrent la voie

à l'études des caractéristques du trafic qui sont pertinentes pour reconnaître le trafic P2P.

Ainsi par la suite, Collins et Reiter [98] exposent une méthode pour distinguer le trafic BitTorrent des trafics FTP, HTTP et SMTP. C'est utile par exemple dans la situation où du trafic P2P chiffré transite par le port 80, pour pouvoir bien l'identifier comme tel et non comme du trafic HTTP. La méthode est généralisable à d'autres protocoles P2P. La technique repose sur l'observation des enregistrements des flux de type Netflow [22] (donc en observant les caractéristiques de trafic de la couche 3). Plus particulièrement, les auteurs observent pour chaque flux les paramètres suivants : la bande passante, le volume de données, le profil de la taille des messages et le taux de connexions échouées. La Figure 2.4 montre un exemple de critère pour distinguer le trafic BitTorrent du trafic HTTP en observant la taille des paquets.

Les auteurs calibrent les seuils des paramètres considérés au moyen d'une courbe ROC, de manière à avoir un taux de faux négatifs faible. Les résultats des seuils obtenus sont : 1% de connexions échouées, une bande passante de 14 ko/s et un volume de 110 paquets par flux. Une des forces de la méthode considérée est qu'elle fonctionne avec du trafic chiffré, puisqu'elle ne repose que sur des informations fournies par Netflow. Un autre avantage est qu'elle produit peu de faux négatifs (0% selon l'évaluation des auteurs). En revanche les auteurs rapportent un taux de vrais positifs assez bas (72%). De plus on distingue seulement le trafic P2P de trois autres types de trafic ayant des caractéristiques différentes. On ne sait pas ce qu'il en est de la distinction avec d'autres protocoles, en particuliers ceux ayant des caractéristiques proches du trafic pair-à-pair.

Finamore et al. étudient et comparent dans [99] trois classifieurs de trafic du service IPTV (télévision par Internet) en P2P. Les trois classifieurs en question sont pDPI [100], IPSVM [101] et KISS [93]. pDPI consiste à chercher des expressions régulières dans la charge utile des paquets, selon la technique décrite dans 2.2.1. IPSVM observe les statistiques sur les entêtes de la couche transport des paquets, ensuite un algorithme de type SVM (Support Vector Machine) effectue la classification. KISS inspecte la charge utile des paquets pour en définir un modèle stochastique, comme expliqué ci-dessus. Les auteurs concluent que KISS est le meilleur classifieur, mais pourrait être amélioré en le combinant aves les autres approches.

En conlusion on peut dire qu'il n'existe aucune solution parfaite, le problème est encore d'actualité et le monde de la recherche travaille encore activement sur la question de la reconnaissance des applications. De plus, il faut souligner que certains développeurs d'applications travaillent dans le sens inverse et définissent des techniques de brouillage afin d'éviter d'être la

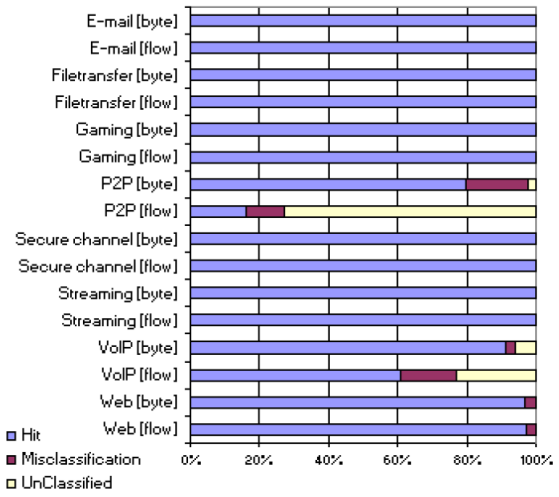


FIGURE 2.3 – Exemple de résultats d'un classifieur

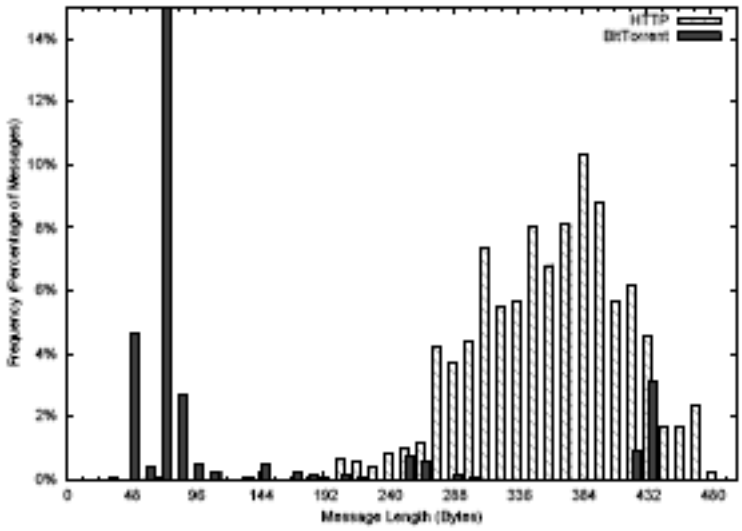


FIGURE 2.4 – Taille des paquets BitTorrent et HTTP

victime de discrimination de la part des FAI ou des administrateurs réseau, par rapport aux autres applications. Skype en est un exemple très frappant, avec ses techniques extrêmement sophistiquées pour empêcher tout reverse-engineering et détection dans le réseau [102]. Ceci pénalise au final les FAI qui n'effectuent pas de discrimination selon les applications, mais qui ont toujours besoin de superviser leur réseau.

2.3 Décodage du protocole eDonkey

Les applications P2P génèrent une proportion importante du trafic dans le réseau (selon Labovitz et al. [18], ainsi que selon un rapport interne à France Télécom sur les statistiques d'usage du réseau). De plus en France, eDonkey est parmi les protocoles les plus utilisés pour les applications de partage de fichiers en P2P. C'est pourquoi nous nous sommes interessés à ce protocole en particulier.

Cette section détaille le travail effectué dans le but d'effectuer une expérimentation de supervision à grande échelle du réseau eDonkey, dans le réseau opérationnel. Il a fallu tout d'abord étudier le protocole eDonkey afin d'être en mesure d'en décoder les messages. Puis nous avons conçu et implémenté la brique logicielle qui va effectivement décoder et journaliser les messages. Cette brique logicielle fut ensuite intégrée à une sonde développée en interne à France Télécom baptisée OTARIE. Cette sonde est déja déployée dans le réseau opérationnel en France, dans un certain nombre de grandes villes françaises.

Les concepteurs du protocole eDonkey n'ont pas mis à disposition du public ses spécifications. Ils veulent ainsi garder le contrôle sur son évolution et sur les applications qui l'utilisent. En préalable à la supervision de ce protocole il faut donc effectuer un travail d'ingénierie inverse (reverse-engineering) pour en découvrir les spécifications. Plusieurs équipes académiques se sont déja penchées sur la question. Nous pouvons citer Kulbak et Bickson [104], qui donne un bon aperçu général du fonctionnement du protocole et une description assez exhaustive des différents types de message. D'autre part, Klimkin [103] décrit plus précisément certains messages. Cependant, aucune source ne spécifie complètement le protocole, ce qui représenterait un travail très conséquent, d'autant plus que le protocole est en constante évolution. Nous avons donc dû analyser le contenu des messages trouvés dans les traces de trafic afin de pouvoir les journaliser. Cette analyse est donnée dans la suite.

2.3.1 Aperçu général du protocole

Dans le but de superviser le réseau overlay eDonkey, il nous est nécessaire de comprendre comment le protocole fonctionne. Cette sous-section donne un aperçu du fonctionnement de eDonkey, ainsi qu'une brève description des messages échangés dans ce protocole.

Le réseau eDonkey est de type centralisé car les serveurs effectuent l'indexation de manière centralisée sans communiquer entre eux. Chaque serveur d'indexation crée donc un réseau overlay indépendant. Un des logiciels client les plus utilisés est appelé eMule. Ce client utilise le protocole eDonkey pour dialoguer, il est entièrement compatible avec les autres clients eDonkey, cependant il a la particularité d'implémenter des extensions qui permettent d'ajouter quelques fonctionalités dans les échanges entre deux clients eMule.

Les clients eDonkey utilisent une seule connexion TCP avec le serveur pour tous les échanges. Par ailleurs les clients maintiennent une queue d'upload pour chaque fichier dans la liste de partage. Les autres clients qui veulent télécharger un fichier en partage s'incrivent dans la queue correspondante et attendent leur tour pour le transfert des données. Les clients peuvent télécharger un fichier de plusieurs pairs, obtenant différentes parties du fichier de chacun. Un pair peut également commencer à uploader à d'autres pairs des morceaux de fichier qu'il est en train de télécharger, même s'il n'a pas le fichier au complet. Les communications entre pairs se font au travers de connexions TCP. Les pairs peuvent également utiliser UDP pour la gestion des files d'attente des uploads mais ce n'est nullement obligatoire (un client incapable d'envoyer ou recevoir des messages UDP, à cause d'un pare-feu par exemple, pourra fonctionner sans problème).

Lorqu'un pair se connecte à un serveur d'indexation, il se connecte au réseau overlay. Le pair reçoit alors un identificateur (appelé «client ID») qui sera valide durant toute la session avec ce serveur. A la connexion, le pair envoie au serveur la liste de ses fichiers en partage. Ceci est effectué au moyen d'un message eDonkey de type *OfferFiles*, décrit plus bas dans la sous-section 2.3.4. Le serveur stocke dans une base de données interne cette liste de fichiers. Le pair envoie par la suite la liste des fichers qu'il souhaite télécharger. Le serveur répond avec les adresses des pairs qui possèdent certains des fichiers désirés. Le pair va alors contacter directement chaque autre pair afin d'obtenir des morceaux des fichiers désirés, jusqu'à ce qu'ils soient obtenus au complet.

En plus du «client ID», qui identifie les pairs et est valide seulement tant que la connexion TCP entre le client et le serveur est maintenue, les utilisateurs eux-mêmes sont identifiés grâce à un identifiant appelé «user ID». Cet identifiant est créé à l'installation du logiciel client et ne change

pas par la suite. L'identifiant est une clé de 128 bits générée en concaténant des nombres aléatoires, de type GUID («Globally Unique Identifier») [13]. Il sert dans le système de crédit de eMule, où les utilisateurs généreux (i.e. qui uploadent beaucoup de données) sont favorisés par rapport aux autres.

Les fichiers sont identifiés par un identifiant appelé indifféremment «file ID» ou «file hash». Ces identifiants servent à la fois à identifier de manière unique chaque fichier et à vérifier leur intégrité après un transfert. En effet cet identifiant est la valeur de hashage sur 128 bits du contenu du fichier. Nous pouvons noter dès à présent que deux fichiers au contenu identique mais avec des noms différents auront le même hash et donc seront considérés par le système comme identiques. Pour les besoins des échanges de données entre pairs, les fichiers sont découpés en *chunks*, c'est-à-dire en morceaux de 180 Ko. Chaque chunk est lui-même divisé en *parties* de 10 Ko. Un message eDonkey de transfert de données contient une *partie* (ce message eDonkey peut être transporté par plusieurs paquets IP bien entendu). Cette terminologie (*chunk*, *partie*) n'est pas reconnue universellement, mais elle sera utilisée comme définie dans ce paragraphe dans la suite.

Lorsqu'un pair se connecte à un autre pair, chacun s'identifie auprès de l'autre au moyen du mécanisme d'authentification clé publique/clé privée. Ensuite le pair qui requiert un fichier se place dans la queue d'upload du fichier correspondant. Quand il atteint le sommet de la queue, il peut débuter le transfert des données du morceau de ficher requis. Ce transfert de données s'effectue au moyen d'un message de type *SendingPart*, décrit plus en détails dans la sous-section 2.3.3.

2.3.2 Encodage des messages eDonkey

Voyons plus en détails l'encodage des messages eDonkey afin de pouvoir extraire les informations des entêtes des messages que nous allons superviser dans notre expérimentation.

Tous les messages eDonkey sont encodés dans l'ordre «little-indian». Ils possèdent tous un entête commun de 6 octets avec la structure suivante :

1. protocole - 1 octet - la valeur de ce champ est 0xE3 pour eDonkey et 0xC5 pour eMule. Ceci sert à déterminer si les extensions eMule peuvent être utilisées ou non.
2. taille - 4 octets - ce champ donne la taille du message en octets sans inclure l'entête. Pour un message *SendingPart*, ce champ sera typiquement à 10 240.
3. type - 1 octet - ce champ contient un identificateur du type du message.

Ainsi par exemple pour un message de type *SendingPart*, le 3ème champ (champ type) aura pour valeur 0x46, puis il y a l'entête du message *SendingPart*. A la suite de cet entête commun de 6 octets, nous trouvons l'entête spécifiques au type de message.

Les entêtes peuvent éventuellement contenir des champs optionnels. Un champ optionnel est codé sous forme d'un *tag*. Un tag a une structure de 4 champs (ou moins, car tous les champs ne sont pas forcément présents) :

1. type - 1 octet - par exemple un tag utilisé pour coder un entier a pour type 3, pour une chaîne de caractères le type est 2 et pour un nombre réel le type est 4.
2. nom - taille variable - ce champ est constitué de 2 octets pour indiquer la taille de la valeur en octet, suivi de la valeur proprement dite. Il est à noter que le nom du tag n'a pas de signification particulière au niveau du protocole, elle sert juste à faciliter la description des tags.
3. valeur - taille variable - la taille de ce champ dépend du type du tag. Pour un tag de type entier ou réel, la taille de ce champ est de 4 octets, qui vont contenir l'entier ou le réel en question. Si le type du tag est une chaîne de caractères, ce champ contient une paire taille-valeur, c'est-à-dire la taille de la chaîne sur 2 octets, suivi de la chaîne.
4. champ special - 1 octet

Pour aider à la compréhension de la structure d'un tag, prenons un exemple. Considérons le tag suivant : `83 02 8d 4f 02 00` (il s'agit d'une représentation en hexadécimal). L'octet 83 représente le type de tag, puis 02 représente le nom du tag (ici il s'agit du nom donné à un tag de taille de fichier). Enfin les 4 octets suivants représentent un entier (qui indique une taille de fichier). Notons que les messages *SendingPart* ne contiennent pas de tag puisque tous les champs sont requis. En revanche les messages de type *OfferFiles* en font usage pour indiquer une liste de fichiers, dont le nombre est bien sûr variable.

2.3.3 Les messages *SendingPart*

Ces messages sont constitués d'un entête puis des données proprement dites, i.e. les 10 240 octets de la *partie* échangée. Si la *partie* échangée est à la fin d'un fichier, il se peut qu'elle soit constituée de moins de 10 240 octets.

Ces messages servent à un envoi d'une partie d'un fichier d'un pair vers un autre. La structure de leur entête est la suivante :

1. protocole - 1 octet

2. taille - 4 octets - taille des données transportées, i.e. typiquement 10 240, sauf si la partie transportée est à la fin d'un fichier, elle peut dans ce cas être de taille inférieure.
3. type - 1 octet - le type est 0x46, qui correspond au code pour un message de type *SendingPart*.
4. hash du fichier - 16 octets - ce champ contient l'identifiant du fichier, i.e. la valeur de hashage de son contenu.
5. position de départ - 4 octets - indique la position du début de cette *partie* dans le fichier.
6. position de fin - 4 octets - indique la position de la fin de cette *partie* dans le fichier.

Les données transportées peuvent être compressées, mais lors de la supervision que nous avons effectué nous n'avons examiné que l'entête des messages, nous ne nous sommes pas intéressés aux données transportées. En effet les entêtes contiennent toutes les informations qui nous sont utiles.

2.3.4 Les messages *OfferFiles*

Ces messages sont utilisés par les pairs pour indiquer leur liste de partage au serveur d'indexation. Un tel message est envoyé par le pair lorsqu'il se connecte au serveur. Par la suite un message *OfferFiles* est envoyé au serveur quand le pair a achevé le téléchargement d'un fichier, dans ce cas la liste de fichiers dans le message ne contient que le fichier qui vient d'être téléchargé. De la même manière un tel message est envoyé lorsque la liste des fichiers partagés change. Voici la structure du message :

1. protocole - 1 octet
2. taille - 4 octets - taille des données transportées sans compter l'entête.
3. type - 1 octet - le type est 0x15, qui correspond au code pour un message de type *OfferFiles*.
4. nombre de fichiers - 4 octets - ce champ indique le nombre de fichiers dans la liste qui suit.
5. liste des fichiers partagés - taille variable - ce champ est constitué d'une suite d'entrées décrivant chaque fichier. La structure des entrées est décrite ci dessous.

Une entrée décrivant un fichier est constituée des champs suivants :

1. hash - 16 octets - identifiant unique du fichier dans le réseau, obtenu par hashage du contenu.

2. client ID - 4 octets - identifiant de la session du pair.
3. port client - 2 octets - numéro de port du client.
4. nombre de tags - 4 octets - indique le nombre de tags qui suivent ce champ.
5. tag de nom de fichier - taillle variable - ce tag est de type chaîne et a pour nom 0x1. La valeur du tag est la taille de la chaîne suivie de la chaîne de caractères proprement dite.
6. tag de taille de fichier - 8 octets - ce tag est de type entier, a pour nom 0x2 et a pour valeur la taille du fichier en octets.
7. divers tags optionnels de description - taille variable - nous ne sommes interessés que par la taille et le nom des fichiers en partage, nous ne détaillons donc pas plus les différents tags optionnels que l'on pourra trouver.

Les champs «client ID» et «port client» sont parfois utilisés pour indiquer si le pair est en possession du fichier au complet ou seulement partiellement. Si ces deux champs ont respectivement la valeur 0xfbfbfbfb et 0xfbfb alors le fichier est partiel, si ces valeurs sont égales à 0xfcfcfcfc et 0xfcfc alors le fichier est complet.

Nous disposons maintenant de tous les éléments pour analyser le trafic eDonkey que nous avons supervisé. Pour réaliser effectivement la supervision du trafic eDonkey dans le cœur du réseau opérationnel, nous avons tiré parti de la sonde OTARIE, qui est déja déployée et dont nous donnons une description rapide dans la section suivante.

2.4 La sonde OTARIE

La sonde Otarie est un outil dévelopé au sein de France Télécom. Il s'agit d'un logiciel écrit en C qui fonctionne sur diverses architectures matérielles de type PC équipées d'une carte de capture de trafic de type DAG (Data Acquisition and Generation) [24]. La carte de capture dispose d'une API qui permet de lire les paquets au fur et à mesure de leur arrivée sur l'interface réseau. La figure 2.5 illustre le fonctionnement de la sonde OTARIE.

Les sondes observent le trafic qui est dupliqué, ainsi il n'y a pas de risque de perturbation du trafic opérationnel.

Chaque paquet est représenté dans le logiciel Otarie au moyen d'une structure en C qui contient les divers champs de l'entête IP, ainsi un tableau de caractères contenant la charge utile du paquet IP. Otarie effectue une classification du protocole applicatif auquel appartient chaque paquet.

Cette classification est effectuée au moyen d'heuristiques et ne se base pas uniquement sur le numéro de port. Une fonction spécifique est appelée selon le protocole applicatif qui est déterminé par le logiciel. Ainsi par exemple, pour les paquets reconnus comme émis par l'application eDonkey, la fonction *eDonkeyFileRequestAnswer* est appelée. Nous pouvons en tirer parti pour effectuer l'analyse qui nous interesse sur les paquets eDonkey. Nous allons placer dans cette fonction le code afin de décoder les paquets *OfferFiles* et *SendingPart* et journaliser les informations obtenues.

Cependant un autre problème se pose : les messages eDonkey que nous souhaitons superviser sont transmis au moyen de connexions TCP, ils peuvent donc se trouver dans plusieurs paquets IP si la taille des messages est supérieure au MTU (Maximum Transmission Unit [14]). Or le logiciel Otarie ne traite les données que paquet IP par paquet IP. Il nous faut donc ajouter une fonction de gestion des contextes TCP afin de pouvoir reconstituer les messages eDonkey répartis sur plusieurs paquets IP.

Cette fonction de gestion des contextes permet de reconstituer le message eDonkey en cas de perte puis retransmission d'un paquet, ou en cas de paquets arrivant dans le désordre. Elle est décrite en détail dans l'annexe C. Afin de ne pas perturber le fonctionnement de la sonde en cas de nombreux contextes TCP ouverts, on limite leur nombre à une valeur fixée lors de la compilation. Dans les sondes déployées dans le réseau opérationnel nous avons fixé ce nombre à 10. Lorsque ce nombre est atteint et qu'il faut créer un nouveau contexte, on écrase simplement le contexte le plus ancien. De cette manière, nous sommes assurés que la mémoire requise pour la gestion des contextes ne dépassera pas les limites fixées au départ. Bien entendu, lorsqu'un message est correctement reconstitué, nous supprimons le contexte correspondant. Dans les faits, au cours de notre supervision le logiciel n'a pas

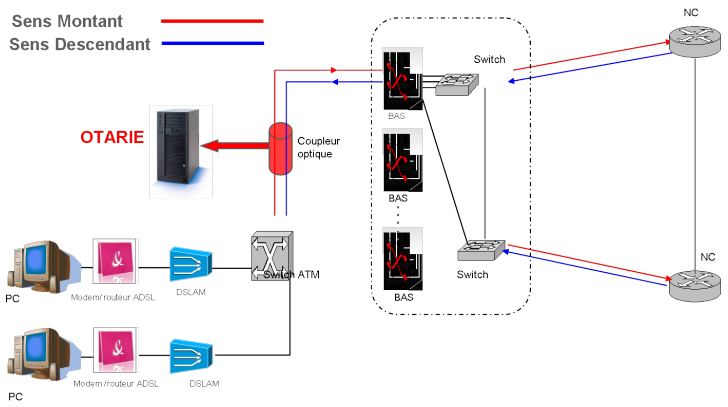


FIGURE 2.5 – La sonde OTARIE

une seule fois été forcé d'écraser un contexte ouvert.

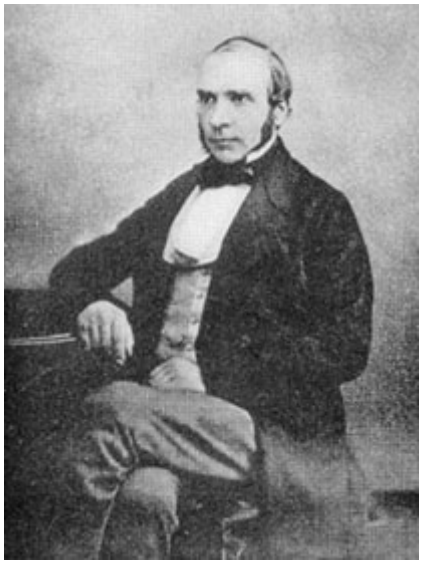
Notons de plus que la fonction de gestion des contextes n'est pas nécessaire pour les messages *SendingPart* car nous n'avons besoin que de l'entête des messages, qui est contenue dans un seul paquet IP puisque sa taille est de 30 octets. En revanche certains messages *OfferFiles* peuvent être transportés sur plusieurs paquets IP puisqu'une liste de fichiers peut contenir jusqu'à 200 fichiers (limite fixée par le protocole), avec pour chaque fichiers diverses informations, donc la taille du message peut être conséquente.

Après avoir décodé les informations qui nous intéressent, c'est-à-dire tous les champs des messages *SendingPart* et les tailles et noms des fichiers des messages *OfferFiles*, nous journalisons toutes les données recceuillies dans un fichier. Ce processus est pris en charge par la sonde Otarie. Pour chaque message eDonkey enregistré nous journalisons également l'horodatage de réception du message, les adresses IP source et destination, ainsi que le sens du message. Dans la suite on appelle sens montant le sens du client vers l'Internet et descendant de l'Internet vers le client. Toutes ces informations sont journalisées dans des fichiers au format texte, puis toutes les 3 heures ces fichiers sont téléchargés automatiquement des sondes vers un serveur de fichiers.

Forts de ces informations, nous serons en mesure dans les chapitres suivants de mettre en place l'expérimentation de supervision nécessaire pour la collecte des données du réseau opérationnel.

Chapitre 3

Epidémiologie des usages applicatifs



John Snow (1818-1853), médecin britannique, est considéré comme l'un des fondateurs de l'épidémiologie moderne.

3.1 Introduction

3.1.1 Intérêt de la supervision applicative

L'étude décrite ci-dessous est fondée sur une observation passive du trafic, en mode hors ligne, à partir de traces complètes, c'est-à-dire contenant l'intégralité de la charge utile des paquets IP. Avoir accès aux paquets en entier permet de mettre en œuvre un outil de détection de trafic malveillant

à base de signature, en l'occurence Snort [8]. Il serait bien sûr irréaliste de vouloir analyser la totalité du trafic dans le cœur de réseau avec un IDS (Intrusion Detection System), pour la simple raison que les coûts pour analyser plusieurs tera-octets de données par seconde seraient bien trop élevés (les performances requises impliqueraient l'emploi d'équipements très couteux et/ou nombreux). Cependant nous allons illustrer que l'analyse du trafic de seulement une petite partie de l'ensemble des utilisateurs permet d'établir des constatations applicables à tous.

Nous avons donc effectué une supervision du trafic dans la couche 7 et nous l'avons exploité à deux titres :

- premièrement, nous avons effectué une reconnaisse des applications utilisées, puis nous avons compilé les statistiques sur les quantités de données émises et reçues pour chaque application, pour chaque client ;
- deuxièmement, nous avons appliqué l'IDS Snort pour déterminer le trafic malveillant émis par les utilisateurs.

Nous voyons donc que la supervision dans la couche applicative nous a permit d'une part de définir avec pertinence l'utilisation qui est faite du réseau et d'autre part de détecter les usages malveillants. Nous avons ensuite pu corréler ces information pour atteindre les objectifs détaillés dans la sous-section suivante.

3.1.2 Objectifs du chapitre

L'épidémiologie est la science de l'étude de l'apparition et de la propagation des maladies dans une population (cf. Bhopal [109]). Dans ce chapitre nous souhaitons mettre en œuvre des techniques issues de cette science dans un but double. Tout d'abord nous voulons valider la pertinence d'appliquer ces techniques dans le contexte de la propagation de virus informatiques. A ce jour, certaines techniques issues de l'épidémiologie – telles que l'étude cas-témoins – n'ont pas encore été mises en œuvre dans le domaine des réseaux informatiques. Nous allons montrer dans la suite de ce chapitre les bénéfices que l'on peut en tirer. Ensuite nous voulons étudier les causes d'infection des machines des utilisateurs de l'internet. L'épidémiologie nous apporte des éléments de réponse à ce sujet, comme nous allons le voir dans la partie 3.6.

Ce chapitre débute par un état de l'art (partie 3.2) des méthodes issues de l'étude des sciences naturelles (épidémiologie, génétique, biologie, etc...) appliquées avec succès au domaine des réseaux informatiques. Puis la partie 3.3 présente quelques concepts et méthodes tirés de l'épidémiologie, que nous allons appliquer. La partie 3.4 suivante détaille la méthodologie de collecte des informations sur lesquelles nous réalisons l'étude épidémiologique. La partie 3.5 donne une analyse statistique des données obtenues. Enfin la partie 3.6

donne les résultats des techniques épidémiologiques que nous avons appliqué et la partie 3.7 interprète ces résultats.

3.2 Travaux précédents sur l'épidémiologie

Le domaine de l'informatique et des réseaux s'est parfois inspiré des sciences naturelles. C'est vrai en particulier pour l'étude des virus informatique, ne serait-ce que par le vocabulaire employé. Virus, ver, infection d'une machine ou d'un fichier, ces termes proviennent directement de la biologie et furent employés pour la première fois par Fred Cohen [105].

Mais ces emprunts vont au-delà du vocabulaire puisque les modèles de popagation des maladies dans les populations humaines furent appliqués avec succès à la propagation de virus ou de vers dans des réseaux de machines. Le premier à étudier la propagation de virus grâce aux modèles épidémiologiques fut sans doute Kephart dans [114]. D'autres modèles plus complexes furent ensuite étudiés dans plusieurs travaux tels que Kephart et al. [110] ou Murray [115].

Dans [106], Goel et Bush étudient les mécanismes biologiques que les organismes vivants utilisent afin de se protéger des agents pathogènes extérieurs. Ils proposent ensuite d'appliquer ces paradigmes biologiques au domaine de la sécurité informatique. Les auteurs s'inspirent en particulier de la génétique (interférences ARN) et de la physiologie (système immunitaire). Ces auteurs ont également appliqué un paradigme issu de l'immunologie humaine à un système informatique en réseau dans [107]. L'article de Stephen Bush [108] est un autre exemple d'étude qui applique un paradigme issu de la génétique pour améliorer la tolérance aux pannes d'un réseau de communication.

Nous proposons dans ce chapitre d'aller plus loin et de mettre en œuvre d'autres outils que l'épidémiologie met à notre disposition afin de mettre en évidence l'impact de la façon dont les utilisateurs utilisent le réseau sur leurs risques d'infection par un virus informatique.

3.3 Concepts et méthodes de l'épidémiologie

3.3.1 Aperçu

La manière dont un virus informatique se propage dans un réseau de machines peut s'étudier de la même façon qu'un virus qui se propage parmi une population humaine. Parmi les outils utilisés en épidémiologie figurent les études de type descriptive, ou bien de type étiologique (comme souligné par Ancelle [111]).

Les études descriptives servent à recueillir des données sur l'état d'une maladie à un moment donné. Les études de type étiologique sont plus ambitieuses car elles cherchent à établir l'existence de relations entre certaines caractéristiques des individus et la survenue de maladie parmi ces individus. Le terme de maladie est ici employé dans un sens très large. Il s'agit de tout évènement pathologique que l'on souhaite étudier, tels que des symptômes précis, ou bien une complication d'une maladie humaine ou encore un comportement anormal d'une machine sur un réseau. Les caractéristiques des individus que nous considérons dans une étude sont appelées «exposition» car ces caractéristiques ont potentiellement un lien avec la survenue de la maladie.

Le but d'une étude étiologique n'est jamais de démontrer une relation de cause à effet entre exposition et survenue de la maladie. En effet l'experimentateur ne fait qu'observer les expositions et le caractère sain ou malade des individus. Il ne peut en aucun cas provoquer une exposition, à la différence d'une étude expérimentale ou d'essais thérapeutiques par exemple. L'étude étiologique sert donc à établir des relations statistiques entre exposition et maladie et éventuellement tout au plus à établir des présomptions de causalité dans ces relations.

3.3.2 Enquête cas-témoins

Une des études étiologiques les plus adaptées pour l'application aux réseaux informatiques est l'enquête cas-témoins. Elle consiste à comparer la fréquence d'exposition au facteur que l'on souhaite tester dans deux groupes : un groupe de sujets malades (appelés les cas) et un groupe de sujet sains (les témoins). Intuitivement, si la survenue de la maladie est liée à l'exposition au facteur, alors on doit observer une plus grande fréquence d'exposition dans le groupe des cas. Naturellement, les deux groupes doivent être sélectionnés dans la même population d'une part et indépendamment de leur exposition au facteur d'autre part, sinon un biais sera introduit dans les résultats de l'étude.

Les résultats d'une enquête cas-témoins sont traditionnellement présentés sous la forme d'un tableau 2x2, comme illustré dans le tableau 3.1.

	cas	témoins
exposés	a	b
non exposés	c	d

TABLE 3.1 – Présentation des résultats d'une enquête cas-témoins

Dans ce tableau, a représente le nombre de cas qui ont été exposé au

facteur que l'on souhaite tester. De la même manière b représente le nombre de témoins qui ont été exposés. La cote d'exposition chez les les cas est égale à a/c. La cote d'expostion chez les témoins est b/d. Le rapport de ces cotes va servir à interpréter le tableau de résultats. Ce rapport est appelé «rapport de cote» (RC) ou bien «odds ratio» (OR). L'appelation anglo-saxone est la plus communément utilisée.

$$OR = \frac{a/c}{b/d}$$

Si l'OR est très supérieur à 1 alors l'association entre exposition et maladie est forte et on dit que le facteur auquel on a observé l'exposition est un facteur de risque. De la même manière si l'OR est très proche de zéro, l'association est forte mais on parle dans ce cas de facteur protecteur. En revanche si l'OR est égal à 1 on ne peut rien conclure sur une éventuelle association exposition-maladie. Comme l'OR est calculé à partir de mesure sur des échantillons de population, il faut également calculer son intervalle de confiance. Pour ceci il existe des formules relativement complexes, détaillées par Martin et Austin [112] et implémentées dans le logiciel OpenEpi [11]. De la même manière pour déterminer le nombre de sujets nécessaires dans l'enquête cas-témoins on peut se référer aux ouvrages de Kelsey et al. [113] et Fleiss et al. [117] et utiliser le logiciel OpenEpi pour effectuer le calcul.

3.3.3 Enquête cas-témoins appariée

Dans certains cas il existe un facteur qui influence à la fois les facteurs de causalité et la survenue de la maladie. Ce facteur tiers peut modifier l'association entre le facteur de risque que l'on est en train d'évaluer et la maladie. Dans ce cas ce facteur tiers est appelé facteur de confusion.

Prenons un exemple concret pour illustrer ce qu'est un facteur de confusion. Nous considérons dans cet exemple la population des Etats-Unis, la maladie (au sens épidémiologique, c'est-à-dire au sens large) considérée est le taux de mortalité et le facteur de risque que nous voulons tester est le fait d'habiter dans une ville balnéaire américaine. Si l'on effectue une enquête cas-témoins avec ces paramètres, nous allons constater qu'il existe un lien entre un taux de mortalité élevé et habiter dans une ville balnéaire. Pourtant ce lien n'est pas causal, car les stations balnéaires ne sont pas spécialement plus dangeureuses que les autres villes du pays. Ce lien existe à cause du facteur de confusion de l'âge. L'âge moyen des habitants est plus élevé donc le taux de mortalité est plus élevé. L'âge influence donc la survenue de la maladie et le facteur d'exposition (habiter au bord de la mer) est lié statistiquement à l'âge (car beaucoup de retraités des USA s'installent dans une

station balnéaire à la fin de leur carrière professionnelle). Ce dernier facteur est donc bien un facteur de confusion. Voici un autre exemple, si l'on considère l'exposition au facteur : «avoir un briquet dans sa poche», nous allons constater qu'il existe un lien entre avoir un briquet dans sa poche et la survenue d'un cancer des poumons. Or il n'y a bien entendu aucun lien de causalité entre exposition et maladie. Le facteur de confusion est ici le fait d'être fumeur ou non. Les fumeurs ont naturellement plus de chance d'avoir un briquet sur eux. Le fait d'être fumeur influence à la fois la survenue de la maladie et l'exposition au facteur que nous voulions tester.

Afin d'éliminer un facteur de confusion, on peut réaliser une enquête cas-témoins appariée. L'appariement consiste à associer à chaque cas un témoin similaire relativement au facteur que l'on souhaite éliminer. On compte alors le nombre de paires pour lesquelles le cas et le témoin sont exposés, les paires pour lesquelles le cas seulement est exposé, les paires pour lesquelles seulement le témoin est exposé et celles ou aucun n'est exposé. Le résultat de l'enquête va donc se présenter sous la forme du tableau 3.2. Dans ce tableau, e par exemple représente le nombre de paires où le cas et le témoin sont tous les deux exposés.

	cas exposé	cas non-exposé
témoin exposé	e	f
témoin non-exposé	g	h

TABLE 3.2 – Présentation des résultats d'une enquête cas-témoins appariée

Dans ce cas on peut obtenir l'Odds Ratio de cette façon :

$$OR = \frac{g}{f}$$

Dans l'exemple des porteurs de briquet donné ci-dessus, chaque cas (i.e. malade du cancer des poumons) porteur d'un briquet est associé à un témoin porteur de briquet et les cas qui ne porte pas de briquet sont associés à un témoin non porteur de briquet également. Dans cette étude on va constater qu'il n'existe plus de lien statistique entre le fait de porter un briquet et être malade.

Nous allons maintenant utiliser ces notions afin de démontrer leur utilité dans le cas de la supervision de clients ADSL. Pour ceci nous avons réalisé une étude cas-témoins sur une population de clients, dont nous détaillons la méthodologie dans la section suivante.

3.4 Méthodologie

Tout d'abord nous devons définir le périmètre de population de notre étude. Nous nous concentrons ici sur les clients ADSL de France Télécom. Parmi cette population nous avons sélectionné un échantillon de clients puis nous avons enregistré tout le trafic émis et reçu par ces clients. A cette fin, nous avons mis en place des sondes dans le réseau opérationnel de France Télécom comme expliqué ci-dessous.

3.4.1 Installation dans le réseau opérationnel

De manière à respecter la vie privée des personnes, nous utilisons une trace de trafic totalement anonyme et nous identifions les clients uniquement grâce à leur numéro VPI/VCI (Virtual Path Identifier/Virtual Channel Identifier). Ces numéros sont utilisés dans le protocole de transport ATM (Asynchronous Transfer Mode). Il nous est totalement impossible d'obtenir les identités des clients à partir de ces numéros de canaux ATM.

Les clients ADSL sont connectés à l'Internet au travers de ce que l'on appelle la chaîne ADSL, qui est composée d'un DSLAM (Digital Subscriber Line Access Multiplexer) et d'un BRAS (Broadband Access Server). Afin de récolter des informations de supervision sur notre échantillon de clients, nous avons mis en place une sonde passive dans la chaîne ADSL. Cette sonde enregistre le trafic émis et reçu par les clients. Le trafic d'un port d'un BRAS est dupliqué avant d'être observé par la sonde, donc il n'y a pas d'interférence sur le trafic opérationnel. Le trafic d'un client donné passe toujours par le même port d'un BRAS, donc la sonde voit bien tout le trafic des clients de l'échantillon. La figure 3.1 montre le placement de la sonde dans l'architecture ADSL. Dans cette figure le trafic du client A est supervisé par la sonde tandis que le trafic du client B transite par un autre port du BRAS et n'est donc pas supervisé.

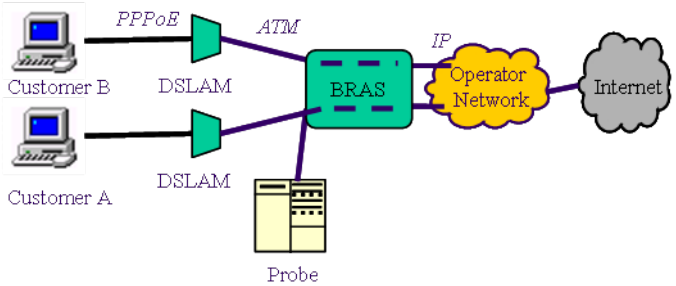


FIGURE 3.1 – Placement de la sonde dans la chaîne ADSL

Le protocole PPPoE (Point-to-Point Protocol over Ethernet) est typiquement utilisé entre le client et le DSLAM, ATM entre le DSLAM et le BRAS, puis IP (Internet Protocol) après.

Nous avons donc mis en place trois sondes et nous avons recueilli des données sous deux formes différentes. Tout d'abord nous avons enregistré des traces de trafic complet pendant une période de temps limitée, à cause des volumes de trafic importants. Par ailleurs, nous avons collecté des statistiques sur le trafic de nos clients supervisés pendant une période de 2 mois. Les traces de trafic complet furent utilisées pour déterminer les clients émettant du trafic malveillant et les statistiques sur 2 mois furent utilisées pour établir un profil d'utilisation du réseau pour chaque client de notre échantillon. Les sondes possèdent des mécanismes pour déterminer le protocole applicatif (i.e. le protocole de niveau 7 de la couche OSI) de chaque flux qu'elles observent. Cette reconnaissance de l'application s'effectue au moyen de signatures et d'heuristiques sur le contenu du champ de données des paquets IP. Grâce à cette reconnaissance on peut établir le profil d'utilisation du réseau pour chaque client. Ce profil contient notamment la quantité de trafic TCP et UDP, le nombre de paquets, ainsi que la quantité de trafic pour le web, le pair-à-pair, FTP, les news, le mail, le streaming, le chat et la voix sur IP.

Les clients infectés/malveillants sont identifiés au moyen d'un IDS (Intrusion Detection System) exécuté sur la trace complète que nous avons enregistré. L'IDS utilisé pour ceci est Snort [8]. Il s'agit d'un IDS qui inspecte chaque paquet IP et lance une alerte lorsque la signature d'un paquet malveillant est reconnue. Une alerte Snort donne l'adresse IP source du paquet qui a généré l'alerte. Puisque les clients ADSL n'ont pas d'adresse IP fixe, nous avons dû garder trace des différentes adresses IP de chaque client et nous avons identifié les clients grâce à leur numéro VPI/VCI (en effet nos captures, entre le DSLAM et le BRAS, utilisent le protocole ATM). Le numéro VPI/VCI reste fixe pour chaque client sauf si ce dernier déménage, c'est donc un moyen d'identification préférable à l'adresse IP.

3.4.2 Résumé des données collectées

Une description des traces complètes est donnée dans la table 3.3 :

Nous pouvons donc constater que 4 634 clients furent observés au total.

Parallèlement à ces traces complètes, nous avons également récolté des statistiques sur le trafic aux même BRAS, sur une période de 2 mois qui inclut la date et l'heure des captures des traces complètes. Ces statistiques sont résumées dans la table 3.5 ci-dessous.

Nous pouvons exécuter Snort en mode hors-ligne sur les traces complètes, mais auparavant il faut configurer l'outil pour nos besoins particuliers.

	Trace 1	Trace 2	Trace 3
nom du BRAS	B1	B2	B3
date	5 déc 2006	5 déc 2006	18 juillet 2007
heure	9h51 à 10h35	9h52 à 10h52	18h03 à 19h04
clients observés	2133	912	1589

TABLE 3.3 – Résumé des traces complètes

3.4.3 Configuration de Snort

Snort est configuré de manière à éviter le plus de faux positifs possible. Pour cela nous avons éliminé les règles qui ont un grand risque de générer un faux positif. Le but recherché est de distinguer les clients ADSL qui lancent des attaques ou bien dont le PC a été infecté et génère du trafic d'attaque. Dans cette optique nous avons donc éliminé les règles qui :

1. dépendent de politiques administratives spécifiques.

   De nombreux domaines administratifs (les entreprises par exemple) interdisent l'usage d'applications telles que le chat ou le pair-à-pair. De même certains administrateurs réseaux veulent une alerte pour les paquets ICMP par exemple. Dans notre cas l'utilisation de ICMP ou d'applications sans rapport avec le travail est bien entendu légitime, en conséquence nous n'incluons pas ce genre de règles dans notre configuration de Snort.

2. ont une définition trop large.

   Certaines règles ont une définition très large, i.e. elles se déclenchent sur de nombreux paquets. C'est ensuite à l'administrateur réseau de vérifier si ces paquets constituent une attaque, avec la connaissance des services et des vulnérabilités potentielles de son réseau. Ainsi par exemple la règle avec l'identifiant SID 247, qui correspond à la description «DDOS mstream client to handler», ne vérifie que le numéro de port destination du paquet. Dans nos traces les alertes remontées par cette règle sont toutes des faux positifs. Ces fausses alertes sont provoquées par du trafic pair-à-pair qui utilise par hasard le numéro de port de la règle. Nous avons vérifié manuellement que toutes les règles éliminées n'ont effectivement généré que des faux positifs dans nos traces.

3. supposent une configuration réseau particulière.

   En général Snort (comme tout IDS) est utilisé dans un environement où il y a un réseau interne de confiance et un extérieur dont il faut se protéger. Ainsi par exemple la règle avec l'identifiant SID 537 (correspondant à la description «NETBIOS SMB IPC$ share access», se

déclenche lorsque un accès réseau est effectué de l'extérieur vers un système de fichiers interne. Dans notre cas, il est parfaitement légitime qu'un client ADSL accède à un système de fichiers sur une machine distante. Comme on ne peut pas vérifier si cet accès est autorisé ou non, nous éliminons cette règle afin de limiter au maximum les faux positifs.

4. ont généré uniquement des faux positifs dans notre trace.

   Parfois certaines règles semblent raisonnablement spécifiques et utiles mais elles n'ont généré que des faux positifs dans nos traces. C'est le cas par exemple de la règle avec l'identifiant SID 2441 («WEB-MISC NetObserve authentication bypass attempt») qui vérifie si une requête http contient un cookie avec un certain mot-clé. Dans nos traces cette règle s'active sur des paquets contenant le mot-clé à l'intérieur d'un autre champ de l'entête HTTP, générant par là-même un faux positif.

5. sont des scans de port.

   Les applications pair-à-pair sont aisément confondues avec des attaques de scan de ports car elles ont besoin de contacter de nombreuses machines sur des nombreux ports différents (ces ports sont spécifiés par le protocole pair-à-pair ou bien décidés par les pairs eux-mêmes). Dans Snort l'algorithme de détection d'attaque de scan est fondé sur un seuil sur le nombre de tentatives de connexion d'une machine, il va donc générer de nombreux faux positifs simplement à cause du trafic pair-à-pair. En conséquence nous éliminons le pré-processeur portscan de Snort pour l'analyse de nos traces.

Une vérification manuelle des alertes observées dans nos traces fut effectuée afin de trouver des faux positifs évidents. Cependant il est parfois impossible de déterminer si une alerte correspond bien à une attaque sans vérifier directement le paquet et son contexte. Or il est impossible de le faire pour chaque alerte à cause de leur nombre trop important. La liste des alertes les plus fréquentes est donnée dans l'annexe B.

Après avoir configuré Snort de la manière décrite dans cette section, nous l'avons exécuté sur nos trois traces complètes. Les résultats sont donnés dans la section 3.6 suivante.

Nous avons également utilisé les 3 traces complètes pour déterminer l'empreinte du système d'exploitation (OS) des utilisateurs présents dans les traces. Nous avons fait appel pour ceci à un outil d'empreinte passif (passive fingerprinting) appelé p0f (cf. [10]). Cet outil détermine l'OS en observant les options TCP des paquets de type SYN émis par les machines. Puis il compare ces options avec une base de données de signatures d'OS connus.

La section 3.5 suivante donne l'analyse des données obtenues grâce à la méthodologie expliquée dans cette section.

3.5 Analyse statistique des données collectées

Dans cette section nous analysons statistiquement les données collectées dans le but de formuler des hypothèses sur les facteurs de risque potentiels que nous allons tester dans les études cas-témoins de la section 3.6 suivante.

La table 3.4 donne un résumé de l'analyse des traces complètes avec les outils Snort et p0f.

	Trace 1	Trace 2	Trace 3
nom du BRAS	B1	B2	B3
nb de clients observés	2133	912	1589
trafic total moyen	8.4 Mo/s	4 Mo/s	8 Mo/s
trafic moyen par client	4 Ko/s	4.4 Ko/s	5 Ko/s
nb d'alertes	325	465	19
OS Windows	536 (25%)	222 (24%)	362 (23%)
OS Linux	42 (2%)	26 (3%)	61 (4%)
OS autre	83 (4%)	19 (2%)	131 (8%)
OS inconnu	1472 (69%)	645 (71%)	1035 (65%)

TABLE 3.4 – Résumé de l'analyse de la trace complète

Nous pouvons observer dans cette table que le trafic moyen par client se situe aux alentours de 4-5 Ko/s. La reconnaissance d'empreinte de l'OS laisse un peu à désirer car on n'a pas pu déterminer l'OS pour environ les deux tiers des clients. Cette faible efficacité de l'outil est dûe au fait qu'il est basé sur des techniques de reconnaissance passive, qui sont bien moins efficaces que les techniques actives (où l'outil va dialoguer avec la machine dont on veut reconnaître l'OS). Mais puisque nous effectuons nos analyses sur une capture de trafic nous ne pouvons bien sûr pas utiliser d'outil de reconnaissance active. Parmi les clients dont on a pu déterminer l'OS, environ les trois quart utilisent Windows.

Voyons maintenant l'utilisation du réseau que font les clients, en volume de trafic pour un certain nombre d'applications. La table 3.5 résume les statistiques récoltées sur une période de deux mois.

Comment lire le tableau : par exemple sur le BRAS B1, chaque client a émis et reçu en moyenne 5 091 Mo de trafic web sur la période de 2 mois, ce qui représente 21,1% du trafic total moyen pour chaque client. Les clients que nous considérons dans cette analyse (dont le nombre est indiqué à la 3ème

		BRAS B1	BRAS B2	BRAS B3
période		Nov-Dec 2006	Nov-Dec 2006	June-July 2007
nb de clients		3,316	1,140	2,219
Trafic moyen par client (en Mo)	total	24,076	6,967	23,721
	TCP	23,334 (96.9%)	6,578 (94.4%)	22,907 (96.5%)
	UDP	716 (2.9%)	208 (2.9%)	772 (3.2%)
	Web	5,091 (21.1%)	2,958 (42.4%)	6,230 (26.2%)
	P2P	15,214 (63.1%)	1,927 (27.6%)	9,075 (38.2%)
	Inconnu	1,452 (6.0%)	618 (8.8%)	1,559 (6.5%)

TABLE 3.5 – Résumé de l'utilisation du réseau par les clients

ligne du tableau) sont tous les clients qui ont eu une activité réseau pendant la période d'observation de 2 mois. Ce nombre de clients est différent du nombre dans la table 3.4 car tous les clients n'ont pas forcément eu d'activité réseau pendant la capture complète.

Dans le but d'effectuer une étude cas-témoins dans la section 3.6 nous devons en premier lieu définir avec précision ce qu'est un cas.

Définition 2 *Un cas est un client qui a été observé dans une trace complète et qui a généré au moins une alerte Snort.*

Remarquons que pour être observé dans une trace complète, un client doit avoir émis ou reçu au moins un paquet durant la période d'observation. D'autre part puisque Snort a été configuré pour générer un minimum de faux positifs, nous considérons que le seuil de 1 alerte est un bon indicateur de trafic malveillant.

Fort de la définition d'un cas, nous pouvons maintenant séparer les cas des autres clients dans notre échantillon de population. La table 3.6 donne un résumé des statistiques d'utilisation du réseau des cas et de la population dans son ensemble. La colonne intitulée «population» correspond à l'ensemble des clients observés dans les traces complètes. En effet il est préférable de se resteindre à ces clients (plutôt que de prendre l'ensemble – plus grand – des clients observés dans les statistiques sur 2 mois) car les cas sont nécessairement issus des clients observés dans les traces complètes.

Nous observons qu'environ 3% des clients dans notre population d'étude sont des cas. Notons dès maintenant que notre méthode de détection des cas sous-évalue leur nombre puisque pour être un cas, un client doit avoir émis une alerte Snort durant une période de capture du trafic. Si le trafic malveillant est émis à une autre période de temps, ce client ne sera pas détecté comme un cas. Nous discutons dans la section 3.7 l'effet de cette sous-estimation.

		cas	population
nb de clients		141	4,634
Trafic moyen par client, par type d'application (en Mo)	Total	66,669	24,997
	TCP	65,398	24,146
	UDP	1,215	775
	Web	17,641	6,029
	P2P	30,231	12,840
	FTP	1,497	344
	News	0	348
	Mail	533	214
	Game	52	113
	Streaming	7,659	2,774
	Chat	213	84
	VoIP	2,930	304
	Inconnu	5,067	1,643
OS Windows		53%	24%
OS Linux		5%	3%
OS Inconnu		38%	68%

TABLE 3.6 – Comparaison des statistiques des cas et de la population totale

Nous observons également que le trafic total moyen des cas est plusieurs fois plus grand que la moyenne de la population globale. Ceci est également vérifié pour le trafic moyen web, pair-à-pair, streaming, chat et le trafic inconnu. De plus, il est intéressant de remarquer que de la même manière le pourcentage d'utilisateurs de Windows est plus grand chez les cas que dans la population totale.

En se fondant sur ces observations nous pouvons formuler des hypothèses sur les facteurs de risque potentiels que nous allons tester dans les études cas-témoins de la section suivante. Ces hypothèses sont les suivantes :

Hypothèse 1 *Utiliser beaucoup des applications pair-à-pair (P2P) est un facteur de risque d'être un cas (cf. la définition 2, page 47).*

Cette hypothèse se comprend de manière intuitive car les utilisateurs qui se servent beaucoup des applications P2P peuvent avoir un plus grand risque de télécharger un fichier contenant un virus et donc peuvent avoir un plus grand risque de se faire infecter.

Hypothèse 2 *Utiliser Windows est un facteur de risque.*

En effet la grande majorité des virus et vers se répandent au travers de l'OS Windows, puisque c'est l'OS le plus courant chez les utilisateurs. Il est donc raisonnable de suspecter que faire usage de ce logiciel est un facteur de risque.

Hypothèse 3 *Surfer beaucoup sur le web est un facteur de risque.*

Certaines pages web contiennent des scripts malveillants qui sont automatiquement exécutés par le navigateur. Utiliser beaucoup le web peut donc augmenter les risques de visiter une telle page et donc de voir son ordinateur infecté par un virus.

Hypothèse 4 *Utiliser beaucoup les applications de streaming est un facteur de risque.*

Pour les mêmes raisons que le trafic web, visiter de nombreux sites de streaming peut augmenter les risques de se faire infecter.

Hypothèse 5 *Utiliser beaucoup les applications de chat est un facteur de risque.*

Bien entendu nous devons préciser ce que nous appelons ici utiliser une application *beaucoup*. Dans la suite nous considérons qu'un client est exposé au facteur de risque associé avec l'utilisation d'une certaine application si ce client est dans les 20% des clients les plus importants en termes de quantité de trafic généré pour cette application. Le chiffre 20% a été choisi de manière à ce qu'il y ait un nombre suffisant de clients exposés. Ceci permet par la suite de valider statistiquement la taille des échantillons de cas et de témoins dans les études épidémiologiques. En effet dans toutes les études cas-témoins effectuées dans ce qui suit nous avons vérifié que les tailles d'échantillons sont supérieures aux minimums donnés par Fleiss et al. dans les formules 3.18 et 3.19 de [117].

Nous allons maintenant dans la section suivante tester les hypothèses formulées précédemment.

3.6 Resultats des études cas-témoins

Le but de l'étude cas-témoins est d'évaluer une association éventuelle entre l'exposition à un facteur (potentiellement un facteur de risque) et le fait que l'individu soit un cas.

Dans la sous-section 3.6.1 plusieurs exemples d'études cas-témoins sont donnés afin d'illustrer la manière dont nous appliquons les concepts et méthodes détaillés préalablement dans la section 3.3. Puis dans la sous-section 3.6.2 nous réalisons des études cas-témoins afin de vérifier les hypothèses formulées dans la section précédente (cf. 3.5). Ce faisant nous mettons à jour un facteur de confusion, nous procédons ensuite à une étude cas-témoins appariée dans la sous-section 3.6.3 de manière à éliminer ce facteur de confusion. Nous pouvons ainsi conclure sur la validité des hypothèses.

3.6.1 Exemples d'études cas-témoins

Dans cet exemple nous allons évaluer l'utilisation d'applications pair-à-pair comme facteur de risque potentiel de voir son ordinateur compromis par un logiciel malveillant. Nous effectuons donc une étude cas-témoins avec la définition 2 d'un cas donnée en 3.5 et la définition de l'exposition suivante : le client est exposé si son trafic P2P (Peer-to-Peer) total est supérieur à 3 547 Mo sur la période de deux mois de supervision. Ce chiffre est choisi pour correspondre au percentile 80 du trafic P2P total. De cette manière 20% des clients sont au-dessus de ce seuil et donc sont considérés comme exposés au facteur. Les résultats de cette étude cas-témoins sont présentés dans la table 3.7.

	cas	témoins
exposés	47	972
non exposés	94	3 521

TABLE 3.7 – Etude cas-témoins, le facteur d'exposition est le P2P

Grâce à ces résultats nous pouvons calculer l'Odds Ratio : il est égal à 1,81 et l'intervalle de confiance à 95% est 1,28 – 2,56. L'intervalle de confiance est calculé grâce à la suite de logiciels pour l'épidémiologie OpenEpi [11] qui utilise les formules de Martin et Austin [112]. Puisque la borne inférieure de l'intervalle de confiance est supérieure à 1, nous pouvons constater qu'il existe bien une association entre exposition et le fait d'être un cas. Cependant nous remarquons également que cette association n'est pas très forte car l'OR n'est pas très supérieur à 1.

De la même manière nous pouvons tester l'association entre le système d'exploitation utilisé par les clients et le statut de cas. La table 3.8 montre les Odds Ratios pour deux études cas-témoins, une avec l'exposition définie comme «utilisateur de Windows» et l'autre comme «utilisateur de Linux». Les Odds Ratios sont donnés dans la table, avec entre parenthèses l'intervalle de confiance à 95%.

	Odds Ratio
Windows	3,7 (2,7 – 5,2)
Linux	1,9 (0,8 – 3,9)

TABLE 3.8 – Résulats des études cas-témoins avec l'exposition à Windows et à Linux

Nous pouvons conclure des résultats de cette étude qu'il y a une association entre être utilisateur de Windows et le statut de cas. Par contre, pour les utilisateurs de Linux, nous ne pouvons rien conclure car l'intervalle de confiance inclut la valeur 1.

Il est également intéressant de combiner plusieurs facteurs d'exposition en un seul. Par exemple nous pouvons définir les clients exposés comme étant ceux qui sont utilisateurs de Windows et qui utilisent le P2P (d'après la définition ci-dessus). Dans ce cas nous obtenons le résultat de l'étude cas-témoins donné dans la table 3.9.

	cas	témoins
exposés	28	325
non exposés	113	4 168

TABLE 3.9 – Etude cas-témoins, le facteur d'exposition est Window + P2P

L'Odds Ratio devient dans ce cas 3,2 avec un intervalle de confiance à 95% compris entre 2,0 et 4,8. L'association entre exposition et statut de cas devient plus forte que dans la table 3.7, ce qui n'est pas surprenant.

On peut aussi se demander ce que l'on obtient avec un seuil différent pour l'exposition au P2P. Nous donnons ici un exemple d'étude cas-témoins avec le facteur d'exposition suivant : «utilisation de P2P $>$ 1 o/s». Un client est donc exposé s'il a généré le moindre trafic P2P.

	cas	témoins
exposés	119	160
non exposés	98	4 049

TABLE 3.10 – Etude cas-témoins, le facteur d'exposition est P2P $>$ 1 o/s

Le résultat de l'étude cas-témoins de la figure 3.10 est OR = 30, avec un intervalle de confiance à 95% compris entre 22 et 42. Ceci confirme les résultats de la table 3.7, à savoir que l'utilisation du P2P est un facteur de risque potentiel, comme nous allons le détailler dans la sous-section suivante.

3.6.2 Identification des facteurs de risque potentiels

Dans cette section nous cherchons à découvrir les facteurs de risque potentiels. A cette fin, nous avons effectué des études cas-témoins pour différentes définitions de l'exposition. En particulier nous avons testé toutes les caractéristiques que nous avons pu supervisé lors de la capture de 2 mois. La table 3.11 résume les résultats obtenus.

	seuil (perc. 80)	Odds Ratio
Total traffic	21,564	**3.6 (2.4 – 5.4)**
TCP traffic	20,538	3.3 (2.1 – 5.0)
Web	6,396	**3.7 (2.4 – 5.5)**
P2P	3,547	3.2 (2.0 – 4.8)
FTP	43	1.9 (1.1 – 3.1)
Streaming	1,692	3.1 (2.0 – 4.8)
Chat	62	**4.1 (2.7 – 6.2)**
VoIP	122	2.0 (1.1 – 3.5)

TABLE 3.11 – Résumé des études cas-témoins avec diverses expositions

La première colonne indique le facteur de risque potentiel que nous testons. La seconde colonne donne le percentile 80 pour le facteur correspondant, qui va constituer le seuil pour déterminer si un client est exposé ou non. Nous voyons donc par exemple que 80% des clients sont en dessous du seuil de 6 396 Mo pour le trafic Web, sur la période de 2 mois. Ces clients seront considérés comme non exposés. Dans notre définition de l'exposition nous avons également inclus le facteur «utilisateur de Windows». Ainsi tous les clients exposés ont un trafic supérieur au seuil donné dans la table et sont également utilisateur de Windows. Enfin la troisième colonne donne l'Odds Ratio de l'étude cas-témoins effectuée, ansi que l'intervalle de confiance à 95% entre parenthèses. Les trois Odds Ratios les plus élevés sont indiqués en gras.

La première chose à remarquer est la forte association entre le trafic total, le trafic web et le trafic de chat, d'une part et le statut de cas d'autre part. Ces associations sont bien sûr uniquement statistiques et pas nécessairement causales. En particulier, pour comprendre l'Odds Ratio obtenu avec l'étude qui teste le trafic total comme facteur d'exposition, il nous faut revenir à la définition d'un cas. Un cas est un client qui a généré au moins une alerte Snort. Donc l'étude nous dit que plus un client émet de trafic, plus il a de risque de déclencher une alerte. Il se peut que certaines alertes soient en fait des faux positifs. En effet, en dépit du soin prit pour éviter les faux positifs dans la configuration de Snort, il est possible qu'il subsiste de fausses alertes

dans notre capture. De fausses alertes peuvent se produire lorsqu'une suite d'octets correspondant à une signature apparaît dans un paquet contenant des données au format binaire. Il s'agit alors d'un hasard, qu'il est virtuellement impossible d'éviter car on peut trouver à peu près n'importe quelle suite d'octets dans un paquet IP transportant des données au format binaire. Dans cette situation, plus un client donné émet de trafic, plus il y a de chance d'y trouver une signature d'une attaque. Le facteur «trafic total» est donc un facteur de confusion, au sens de la définition donnée en 3.3.3. En effet ce facteur est lié à tous les autres facteurs (c'est la somme de tous les types de trafic) et il influence le statut de cas (à cause des faux positifs). En conséquence, nous voulons éliminer ce facteur de confusion de nos études cas-témoins. C'est possible grâce à des études cas-témoins appariées.

3.6.3 Etudes cas-témoins appariées

Dans les études suivantes nous apparions chaque cas avec un témoin similaire concernant leur trafic total (i.e. leur trafic total est identique, à plus ou moins 5% du trafic du cas). La table 3.12 résume les résultats obtenus avec les études cas-témoins appariées. Chaque ligne correspond au résultat d'une étude. La première colonne donne le facteur que nous avons testé. La seconde donne le seuil pour considérer un client comme exposé ou non. Enfin la troisième colonne donne l'Odds Ratio ainsi que l'intervalle de confiance à 95% entre parenthèses. Les Odds Ratios significatifs (i.e. dont la borne inférieure de l'intervalle de confiance est strictement supérieure à 1) sont indiqués en gras.

	seuil (perc. 80)	Odds Ratio
Total trafic	21 564	NA
TCP trafic	20 538	1,8 (0,8 – 4,0)
Web	6 396	**3,2 (1,5 – 7,6)**
P2P	3 547	1,7 (0,8 – 3,9)
FTP	43	1,4 (0,6 – 3,5)
Streaming	1 692	**7,7 (2,5 – 32)**
Chat	62	2,0 (1,0 – 4,4)
VoIP	122	3,4 (0,5 – 3,6)
Windows	NA	**4,7 (2,5 – 8,8)**
Linux	NA	1,0 (0,3 – 3,1)

TABLE 3.12 – Résultats des études cas-témions appariées

Nous observons donc que lorsqu'on élimine le facteur de confusion, les clients qui utilisent «beaucoup» le web et les applications de streaming ont

plus de risque d'être des cas. De même pour les clients qui utilisent Windows. Ces trois facteurs d'exposition sont donc bien des facteurs de risque. Il faut noter cependant qu'il n'y a pas nécessairement une relation de cause à effet entre l'exposition au facteur de risque et l'infection par un virus informatique.

Nous pouvons maintenant conclure sur certaines hypothèses formulées à la fin de la section 3.5 : les hypothèses 2, 3 et 4 sont bien vérifiées. En revanche on ne peut rien conclure sur les hypothèses 1 et 5 car les intervalles de confiance des Odds Ratios incluent 1.

Maintenant que nous avons identifié trois facteurs de risque, nous pouvons les combiner en un seul facteur afin d'établir un profil à risque. La table 3.13 donne le résultat d'une étude cas-témoins appariée dans laquelle le facteur testé est l'exposition simultanée aux trois facteurs «utilisateur de web», «utilisateur de streaming» et «utilisateur de Windows».

	témoin exposé	témoin non-exposé
cas exposé	2	15
cas non-exposé	3	119

TABLE 3.13 – Etude cas-témoins appariée, facteur d'exposition est P2P + streaming + Windows

Pour cette étude, nous voyons d'après la table 3.13 que nous avons fais intervenir 139 paires cas-témoins, soit 278 individus. En effet nous sommes limités car nous devons associer à chaque cas un témoin similaire par rapport au trafic total généré. Il y a donc certains cas qui n'ont pas pu être inclus dans l'étude, faute de témoin pour l'appariement. Par conséquent, il nous faut maintenant vérifier que le nombre d'individus inclus dans l'étude est suffisant pour la validité statistique des résultats obtenus. Pour cela nous avons utilisé la formule de Fleiss et al. [117]. Compte tenu de l'intervalle de confiance désiré (95%), la puissance du test (80% de précision pour la détection des cas), le ratio de cas par rapport aux témoins (ici 1), la proportion de cas exposés (3,6%) et l'Odds Ratio (5), la formule nous indique qu'il nous faut au moins 216 individus pour l'étude. Les résultats de l'étude sont donc bien statistiquement significatifs puisqu'elle comprend 278 individus.

L'Odds Ratio pour cette étude est égal à 5 (intervalle de confiance 1,6 - 22). 205 clients furent exposés à cette combinaison de facteurs sur les 4 634 clients de l'étude, donc approximativement 4%. Nous avons donc établi un profil à risque représentant 4% des clients, qui ont 5 fois plus de risque que les autres de générer du trafic malveillant.

3.6.4 Discussion

Ce chapitre a montré qu'utiliser Windows et qu'utiliser beaucoup des applications web et streaming sont des facteurs de risque pour se faire infecter par des logiciels malveillants. On peut supposer que les utilisateurs Windows sont plus exposés puisque de nombreux virus et vers exploitent des failles de sécurité dans cet OS. Cela est probablement dû au fait que cet OS est le plus répandu chez les utilisateurs de l'Internet.

Ce type d'étude ne nous dit rien sur les causes d'infection, mis à part que ces causes sont en lien, d'une manière ou d'une autre avec l'usage des applications web et streaming. Déterminer les causes exactes d'infection aurait requis une analyse au cas-par-cas de chaque client infecté et cela dépasserait largement le cadre des outils offerts par l'épidémiologie. En revanche nous avons montré avec succès qu'il existe des tendances au niveau de la population des clients ADSL prise dans son ensemble. Effectuer une analyse des causes au niveau de chaque client individuellement supposerait une étude totalement différente, si toutefois c'était possible.

Nous avons choisi Snort comme mécanisme de détection des clients qui émettent du trafic malveillant, car cet outil est à base de signature et cela permet un nombre de faux positifs réduits – grâce à une configuration adéquate. Cependant cet outil n'est pas parfait, sinon à l'évidence les virus et les vers ne poseraient pas autant de problèmes qu'ils ne le font actuellement. Nous avons pu néanmoins être en mesure d'obtenir des résultats significatifs, en dépit de ces imperfections, en prenant en compte la possibilité de faux positifs dans nos études cas-témoins. C'est précisément pour cette raison que nous avons effectué des études cas-témoins appariées par rapport au facteur «trafic total». Nous avons pour cela dû supposer que la quantité de faux positifs émis par un client est liée à la quantité de trafic émis. Cette supposition semble raisonnable car un client qui émet très peu de trafic a beaucoup moins de chance de générer un faux positif par hasard, qu'un client qui a un trafic conséquent. Grâce à cette supposition et grâce à des techniques mises à notre disposition par l'épidémiologie nous avons pu éliminer l'impact des faux positifs des études cas-témoins.

Comme nous l'avons souligné dans la section 3.5, notre méthode de détection sous-estime le nombre de cas puisque il est possible qu'un client soit infecté mais qu'il ne génère pas de trafic malveillant pendant notre capture de trafic. La conséquence est que certains cas ont peut-être été compté dans la catégorie témoin lors de nos études cas-témoins. Ceci a pour effet d'affaiblir l'association entre exposition et statut de cas et donc certaines associations ont pu passer inaperçu dans nos études. Cependant les associations mises en évidence ne sont pas remise en cause et pourraient même être plus fortes que

ce que nous avons obtenu. Si nous avions voulu montrer d'autres associations que celles découvertes dans ce chapitre, nous aurions du effectuer une capture de trace complète plus longue que celle à notre disposition. Cependant ceci est très coûteux tant en termes de capacités de stockage que de capacités de traitement.

Les résultats obtenus et présentés dans ce chapitre sont intéressants pour les raisons suivantes :

- ils soulignent la validité et l'utilité des concepts et des méthodes de l'épidémiologie appliqués aux virus informatiques et à une population de clients ADSL ;
- ils ont étés obtenus à partir de données provenant du réseau opérationnel, à partir d'un grand nombre d'utilisateurs réels ;
- les résultats présentés confirment où les causes d'infection par des virus doivent être recherchées. Ce point est certainement déja bien connu par les spécialistes du domaine, mais cela n'a jamais été prouvé avec une méthodologie rigoureuse.

3.7 Conclusion

En supervisant le trafic dans le cœur de réseau de France Télécom, nous avons pu récolter des données sur l'utilisation que font du réseau un certain nombre de clients ADSL. En parallèle, nous avons utilisé un IDS pour déterminer les clients qui émettent du trafic malveillant, à cause d'une infection par un virus (cas supposé le plus fréquent), ou parce que le client s'adonne à des activités malveillantes (cas supposé plus rare). Ensuite nous avons appliqué des techniques issues de l'épidémiologie de manière à formuler des hypothèses sur les facteurs de risque potentiels, puis nous avons testé ces hypothèses. Nous avons tout d'abord identifié l'utilisation d'applications de P2P, web, streaming et chat comme des facteurs de risque potentiels, de même que l'utilisation de l'OS Windows. Les résultats des études cas-témoins ont suggérés que l'utilisation de Windows, ainsi que d'applications de web et streaming sont bien des facteurs de risque. Pour les autres facteurs d'exposition, nous ne pouvons rien conclure, soit car les données ne sont pas assez exhaustives, soit tout simplement car il n'existe pas d'association avec le statut de cas.

Enfin nous avons établi un profil d'utilsateurs à risque, qui ont plus de risque de se faire infecter par un virus. Ce profil est fondé sur l'utilisation des applications réseau qui est faite.

Cette étude peut être appliquée par exemple pour effectuer une sélection rapide (i.e. screening) pour détecter les clients les plus exposés à une contami-

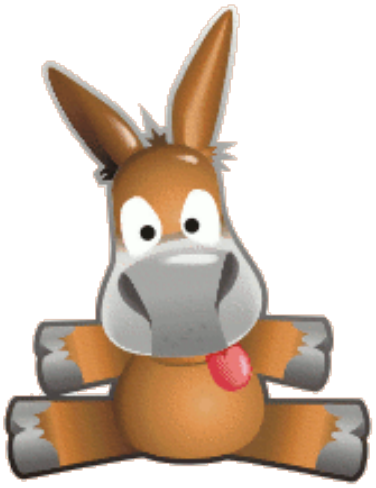
nation d'un ver ou d'un virus. Elle peut être également utilisée pour réduire le coût de la supervision pour la sécurité, puisque nous pouvons restreindre le nombre de clients à superviser.

Au-delà de l'utilité des méthodes issues de l'épidémiologie et des résultats présentés dans ce chapitre, nous devons noter que toute cette étude repose sur la supervision des usages applicatifs des clients. Ainsi nous pouvons conclure en affirmant que la supervision dans la couche 7 permet d'ouvrir de nouveaux champs d'investigation dans le domaine de la sécurité, comme nous l'avons illustré ici.

Chapitre 4

Caches P2P

EMule est l'un des logiciels d'échange de fichiers en P2P les plus utilisés en France.



4.1 Introduction

L'étude réalisée dans ce chapitre est fondé sur l'observation en ligne du trafic d'une application de partage de fichiers en P2P. Les données exploitées résultent du décodage de certains champs du protocole de niveau 7, qui se trouvent dans la charge utile des paquets. Nous allons montrer dans la suite que grâce aux informations collectées, nous sommes en mesure de quantifier avec pertinence les bénéfices à attendre d'une architecture réseau qui comporte des caches spécialisés pour ce type de trafic.

La supervision dans la couche applicative nous permet ici d'améliorer l'ingénierie du réseau en orientant les choix d'architecture.

Afin d'illustrer ce dernier point, nous allons tout d'abord expliquer pourquoi les caches P2P présentent un intérêt (4.2.1) puis quels sont les travaux préalables sur ce sujet (4.2.2 pour la supervision du trafic P2P et 4.2.3 pour les caches P2P). Dans la partie 4.3, nous exposons de manière la plus précise possible la méthodologie employée pour l'expérimentation de supervision. La partie 4.4 donne quelques statistiques descriptives sur les données récoltées. Puis la partie 4.5 présente les résultats des simulations de cache basées sur les données provenant de l'expérimentation de supervision dans le réseau opérationnel. Enfin la partie 4.6 expose l'interprétation des résultats et discute des performances des caches P2P en fonction de leur localisation dans le réseau.

4.2 Contexte

4.2.1 Intérêt de l'étude

Le volume du trafic transporté par les opérateurs de réseaux augmente constamment. Ces derniers voient donc leurs coûts d'installation et d'exploitation augmenter pour faire face à ce surcroît de trafic. Cette augmentation est dûe principalement à l'essor de l'utilisation des applications multimédias (en particulier les flux mutlimedia par HTTP) et des applications P2P (pair-à-pair). Actuellement une proportion importante du trafic est constituée de flux P2P, pour des services tels que le streaming, l'échange de fichiers, ou encore la téléphonie. Ainsi, optimiser ce type de trafic peut bénéficier grandement à l'opérateur, en diminuant la charge sur son réseau. Ceci peut également bénéficier aux utilisateurs qui peuvent jouir d'une meilleure qualité de service dans l'utilisation de leurs applications P2P.

Plusieurs approches existent pour optimiser le trafic P2P. Certaines sont basées sur l'amélioration du mécanisme de sélection des sources pour accéder à une ressource dans l'overlay. L'approche initiée par le consortium P4P (Proactive Participation of ISPs in P2P) en fait partie, cf. [141, 142]. Une autre approche, non-exclusive, est l'utilisation de caches pour les applications P2P [127]. Les caches peuvent être mis en place par les opérateurs afin de diminuer la charge réseau aux points de transits et de peering, ce qui permet de diminuer leurs coûts.

Cependant l'efficacité des caches dépend de nombreux facteurs, tels que leur placement dans le réseau, leur taille, leurs mécanismes de cache, etc... L'un des facteurs les plus importants n'est probablement pas lié directement au cache lui-même, mais plutôt aux caractéristiques des ressources de l'overlay P2P. Ainsi par exemple, si nous constatons que dans une application de partage de fichiers tous les utilisateurs téléchargent des fichiers différents les uns

des autres, les performances d'un cache vont être très faibles, à l'évidence.

Un des objectifs de ce chapitre est donc d'étudier et de modéliser les caractéristiques d'une application P2P sur des données suffisamment importantes en volume et en nombre d'utilisateurs, de manière à ce que nous puissions en tirer des conclusions significatives sur l'opportunité des caches P2P pour un opérateur tel que France Télécom. Nous avons choisi de baser notre étude sur eDonkey parce qu'il s'agit de l'une des applications P2P les plus populaires en France.

De plus, ce chapitre va déterminer quelles seraient les performances d'un cache s'il avait été placé dans le réseau opérationnel de France Télécom et s'il voyait passer du trafic réel. A cette fin, nous avons conçu une simulation dans laquelle les échanges de données entre les pairs eDonkey, observés dans nos captures de trafic réel, sont injectés. Ces données sont traitées par le cache simulé. En résumé les objectifs de cette étude sont :

- évaluer les performances d'un cache à partir de trafic capturé dans le réseau opérationnel de France Télécom ;
- évaluer la réduction de consommation de bande passante dans les points de transit ou de peering de France Télécom, grâce à un cache P2P.

La trace provenant du réseau opérationnel que nous avons exploité vient d'une expérimentation de supervision du trafic dans la couche 7, conduite sur environ 10 mois, sur plus de 18 850 utilisateurs de l'Internet.

La décision de déployer des caches P2P est au final une décision économique. Pour prendre une telle décision, il est nécessaire de connaître les bénéfices des caches en termes de réduction de bande passante. C'est pourquoi cette question est importante.

4.2.2 Travaux en lien avec la supervision du trafic P2P

Le trafic P2P a été largement étudié, pour de multiples raisons. Tout d'abord, un certain nombre d'applications très populaires génèrent une quantité colossale de trafic dans le réseau. Certains FAI ont réagit en limitant le débit, voire en bloquant complètement ce trafic. Cette situation a donc conduit à des efforts de la part des développeurs d'applications P2P de brouiller le trafic émis par les pairs, de manière à ne pas être victime de ces limitations. Parallèlement, des efforts ont été conduits pour étudier les propriétés du trafic P2P, afin de pouvoir identifier le trafic P2P, même brouillé. L'identification du trafic P2P peut s'effectuer grâce à la reconnaissance de signatures dans la charge utile des paquets comme le font Sen et Wang [120], ou en observant des propriétés de couche 3 qui sont caractéristiques du trafic P2P comme Perenyi et al. [122]. D'autres techniques consistent en participer au réseau overlay pour l'espionner (Ohzahata et Kawashima [121]) ou mettre en œuvre

des techniques statistiques, comme Bonfiglio et al. [123] par exemple. Plus de détails sur la reconnaissance du trafic P2P dans le réseau sont donnés dans la partie 2.2.3.

Une alternative à la limitation de débit du P2P consiste à optimiser ce trafic, de manière à réduire la charge dans le réseau. Cette problématique a été décrite par Karagianis et al. [137]. Par ailleurs, la prise en compte de la proximité dans les applications P2P fut soulignée comme une façon de diminuer la charge dans le réseau également par Choffnes et al. [139]. Par la suite, le consortium P4P (Proactive Provider Participation in P2P) a développé cette approche avec des expérimentations [141] et un groupe de travail de standardisation [140]. D'autres études ont visé à modéliser les systèmes P2P afin d'améliorer leur efficacité et ont étudié leur impact sur le réseau sous-jacent. Dans [125], Pouwelse et al. étudient les propriétés d'overlays BitTorrent réels. En particulier, ils ont étudié des propriétés telles que l'activité réseau du système, la disponibilité des pairs, le débit de téléchargement, ou les phénomènes de flash-crowd. Cependant aucune de ces propriétés n'ont d'impact sur l'efficacité des caches P2P.

De la même manière, dans [126], Al-Hamra et al. supervisent des overlays BitTorrent et étudient l'impact de leur structure sur leur robustesse. Ils s'intéressent particulièrement à la taille de l'ensemble des pairs dans l'overlay, la vitesse pour atteindre la taille maximale et le diamètre de l'overlay. Dans [143], Sen et al. étudient le trafic P2P au niveau flot, sur plusieurs routeurs de bord d'un grand FAI. En particulier ils se sont interessés à la connectivité des applications P2P et la dynamicité des systèmes P2P. Handurukande et al., dans [124], ont étudié la popularité ds fichiers dans eDonkey et ont observés des grapes géographiques de pairs qui offrent un même fichier. Ces regroupements géographiques indiquent que les caches pourraient être utiles, mais dans leur étude les auteurs n'ont pu recueillir des données que sur les fichiers offerts et pas sur les fichiers téléchargés.

En conclusion, parmi les caractéristiques du trafic P2P qui peuvent avoir un impact sur les performances d'un cache, seulement la popularité des fichiers fut étudiée. Nous allons nous attacher dans ce chapitre à donner une liste des facteurs qui vont impacter la performance des caches P2P.

4.2.3 Travaux précédents sur les caches P2P

Les caches web furent étudiés dans de nombreux travaux. On a découvert en particulier que la popularité des objets du web suivait une loi de Zipf, comme indiqué par Wang dans [129]. Cependant les caches P2P représentent en réalité une problématique bien différente à cause des disparités importantes entre contenus webs et P2P.

Wierzbicki et al. évaluent dans [128] les algorithmes de remplacement des caches web et proposent un nouvel algorithme qui tire parti des caractéristiques du trafic P2P. Plus tard, dans [127] et [133], Saleh et Hefeeda modélisent la popularité des objets P2P, en se basant sur une expérimentation de supervision passive du réseau Gnutella. Ils observent que la popularité des fichiers ne suit pas la loi de Zipf, mais plutôt une loi dite de Mendelbrot-Zipf. Ils proposent également un nouvel algorithme de remplacement de cache, évalué grâce à une simulation. Les résultats montrent qu'un taux de hits de 35% peut être réalisé avec un cache de taille égale à 10% du trafic total. Cet algorithme consiste à admettre les objets dans le cache de manière partielle et incrémentielle. Les fichiers les plus populaires vont donc par conséquent avoir plus de segments dans le cache. Cependant, les effets de la taille de la population servie par le cache ne sont pas considérés dans cet article, même s'il s'agit en fait d'un des facteurs les plus déterminants sur les performances d'un cache. Une autre limitation de ces études est que la trace de données utilisée ne contient que les requêtes des utilisateurs. Mais les utilisateurs Gnutella peuvent très bien effectuer une recherche sans pour autant lancer le téléchargement des fichiers obtenus en réponse de la requête. En conséquence, supposer que chaque requête est suivie par le téléchargement des fichiers conduit à sur-estimer les quantités de trafic téléchargé et donc cela va également sur-estimer les perfomances du cache P2P. A l'inverse, nous nous sommes basés sur la capture des téléchargements effectifs des utilisateurs pour nos études.

Les études sur les caches P2P comparent les politiques de remplacement les unes avec les autres, pour un nombre donné de clients servis par le cache. Mais le nombre de clients servis est un facteur crucial sur la performance du cache, comme nous allons le montrer dans la suite. Nous allons donc estimer dans la suite les performances d'un cache P2P avec différentes valeurs du nombre de clients, en particulier avec des valeurs typiques dans le cadre d'un déploiement opérationnel dans le réseau d'un opérateur.

4.3 Méthodologie de collecte des données de supervision

Afin d'être en mesure d'observer le trafic P2P nous avons conçu et implémenté un module logiciel destiné à être intégré dans les sondes Otarie, déployées dans le réseau opérationnel. Ce module voit et effectue des traitements spécialisés sur les paquets IP qui sont classifiés comme eDonkey par Otarie. Plus précisément, le module comporte les fonctions suivantes :

1. reconstitution des contextes des flots TCP à partir des paquets : en cas de perte ou d'arrivée en désorde des paquets IP, le module effectue un ré-ordonnancement des paquets de manière à reconstituer le flot TCP. En effet parfois les messages eDonkey sont découpés en plusieurs paquets IP, à cause de leur taille ;
2. décodage de tous les champs de l'entête eDonkey des messages *SendingPart* ;
3. décodage de tous les champs des messages *OfferFiles* ;
4. journalisation dans un fichier de toutes les informations décodées dans les messages eDonkey, ainsi que les informations des entêtes IP et TCP.

Après une période de test et de débogage exhaustifs en mode hors-ligne, sur des traces de trafic, nous avons pu déployer le code de supervision et de journalisation du trafic eDonkey dans trois sondes. Deux de ces sondes sont dans le réseau de collecte ADSL (Asynchronous Digital Suscriber Line) et la troisième est dans le réseau de collecte FTTH (Fiber To The Home). La figure 4.1 montre schématiquement l'emplacement des sondes dans l'architecture réseau.

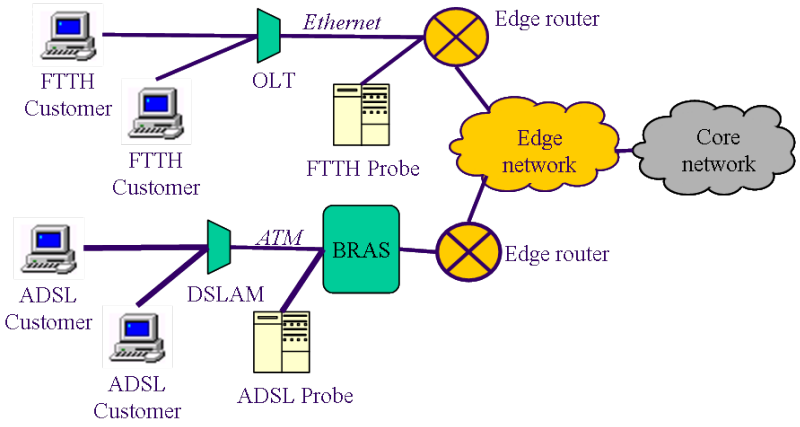


FIGURE 4.1 – Placement des sondes dans le réseau de collecte

La partie supérieure de la figure schématise le réseau de collecte FTTH. La STB (Set-Top Box) ou le modem du client est connecté à l'OLT (Optical Termination Line), qui est lui-même connecté au routeur d'accès. Le trafic (dans les deux sens) d'un port de ce routeur est dupliqué et envoyé à la sonde. Dans notre cas il s'agit de trafic Ethernet.

La partie inférieure de la figure montre le réseau de collecte ADSL. Le modem ou la STB du client est connecté au DSLAM (Digital Subscriber Line Access Multiplexer) qui est connecté au BRAS (Broadband Access Server).

De la même manière que dans le cas de la collecte FTTH, le trafic dans les deux sens sur le port d'un BRAS est dupliqué et envoyé à la sonde ADSL. Dans ce cas le protocole de niveau 2 du trafic supervisé est ATM (Asynchronous Transfer Mode).

Chaque sonde voit donc le trafic de certains clients uniquement. En revanche elles voient la totalité du trafic des clients supervisés car le trafic de chaque client passe toujours sur le même port du BRAS ou du routeur d'accès. Puisque les sondes ne voient que du trafic dupliqué, elles ne peuvent avoir aucun impact sur le trafic opérationnel, en cas de panne ou de dysfonctionnement.

Les clients ne peuvent pas être identifiés grâce à leur adresse IP car elle change au cours du temps (l'analyse des traces a montré que chaque client utilise en moyenne plus de 5 adresses IP). Nous n'avons donc pas utilisé l'adresse IP mais plutôt l'adresse MAC (Media Access Control) du modem ou de la STB du client pour identifier les clients FTTH. Pour les clients ADSL nous avons utilisé les identifiants VP/VCI (Virtual Path/Virtual Channel Identifier). En effet le trafic des clients ADSL est transporté sur ATM. Ces identifiants sont uniques et sont associés à la ligne physique (pour ADSL) ou au matériel du client (pour FTTH), donc ils ne changent que si le client déménage ou change son matériel, ce qui n'arrive peu souvent à l'échelle de temps de l'expérimentation de supervision. Il n'y a aucun moyen pour les expérimentateurs d'obtenir le nom des clients à partir des identifiants de niveau 2 du trafic supervisé, la trace collectée est donc totalement anonyme.

La sonde FTTH fut mise en place dans le réseau à partir du 11 août 2008, jusqu'au 31 mai 2009 (pour un total de 293 jours). Une des deux sondes ADSL a supervisé le trafic dans notre expérimentation du 3 novembre 2008 jusqu'au 19 janvier 2009, date à laquelle la sonde a dû être retirée du réseau pour des raisons de modifications de la topologie du réseau. L'autre sonde ADSL a fonctionné du 3 novembre jusqu'au 31 mai 2009.

La table 4.1 donne un petit résumé de l'échelle de l'expérimentation.

Sur les 18 851 clients supervisés, 7 012 ont utilisé eDonkey pendant la période d'observation, ce qui représente environ 37% (eDonkey est donc très populaire en France).

Afin de pouvoir exploiter efficacement une quantité aussi importante de données (plus de 4 milliards de messages eDonkey), les données collectées furent stockées de manière agrégée dans une base de données relationnelle de type SQL (Structured Query Language). Afin de permettre un accès flexible aux données, tout en gardant une certaine rapidité d'exécution des requêtes SQL, nous avons créé six tables dans la base.

- une table des clients utilisateurs de eDonkey ;
- une table des tailles des fichiers, obtenues à partir des messages *Offer-*

Files ;
- une table des partages de chaque client ;
- deux tables pour les fichiers respectivement uploadés et téléchargés par les clients ;
- une table des adresses IP distantes observées.

Ces tables sont décrites en détails dans l'annexe D. L'agrégation s'effectue dans les tables des fichiers uploadés et téléchargés. En effet on ne stocke que le nombre de messages par périodes de trois heures, pour un client et un hash donnés. Les informations que l'on perd en procédant ainsi sont :

- l'horodatage précis de chaque message ;
- le numéro de la *partie* qui est téléchargée ou uploadée.

Cependant ce découpage permet un gain très important en stockage et en temps de traitements pour l'exploitation des données. En effet il existe une grande redondance des données dûe à leur nature même : on peut observer des rafales de messages *SendingPart* pour télécharger ou uploader le même fichier vers le même client, ce sont simplement différentes parties du fichier qui transitent. Ainsi, s'il n'y avait pas d'agrégation dans la table des téléchargements, on aurait donc une entrée dans la base de données pour chaque message *SendingPart* dans le sens descendant. Dans ce cas on aurait environ 673 millions d'entrées dans la table. En effectuant l'agrégation par périodes de trois heures, on obtient alors environ 1 million d'entrées dans la table. Pour la table des uploads, le gain est encore plus spectaculaire puisqu'on passe de environ 2,6 milliards d'entrées à 2 millions, soit un facteur d'environ 1 250.

Examinons maintenant les données collectées avec la méthodologie que nous venons d'exposer.

4.4 Statistiques

Dans la suite nous donnons un aperçu des données collectées, grâce à quelques statistiques descriptives ainsi que des paramètres utiles pour l'évaluation des performances des caches.

4.4.1 Tailles des fichiers

Nous avons observé environ 551 000 fichiers uniques dans les messages *OfferFiles*. Or ces messages contiennent la taille de chaque fichier listé. Nous avons donc pu journaliser la taille de ces fichiers, qui représentent environ la moitié de tous les fichiers observés. En effet nous avons pu observer également

des fichiers dans les messages *SendingPart* mais ces messages ne donnent pas d'indication sur la taille du fichier.

La figure 4.2 présente la distribution des tailles des fichiers observés. Pour des raisons de lisibilité la première barre, correspondant à l'intervalle 0–20Mo, ne figure pas. En effet cet intervalle concentre un pic de 455 200 fichiers.

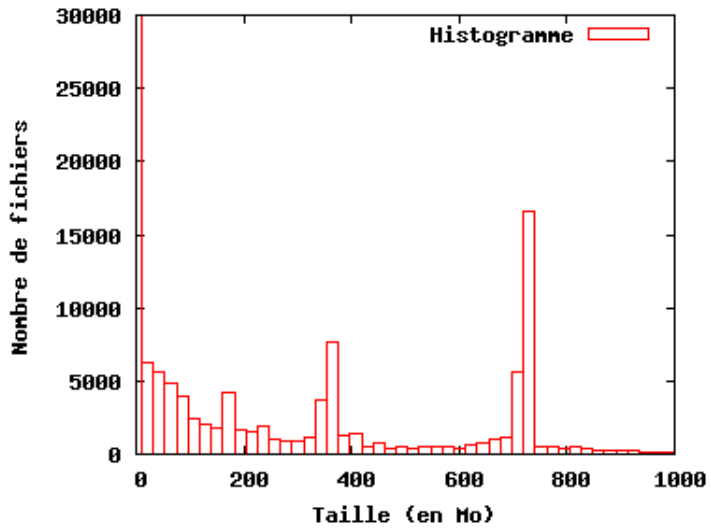


FIGURE 4.2 – Histogramme de la répartition des tailles de fichiers

Le deuxième pic le plus grand se situe autour de la valeur 730Mo, avec environ 16 500 fichiers. Nous observons également un pic autour de la valeur 365Mo (7 800 fichiers). Ces valeurs s'expliquent par le fait que la taille d'un film au format vidéo divx est généralement autour de 720–750Mo et la taille d'un épisode d'une série télévisée dans le même format vidéo est généralement autour de 350–380Mo. La multitude de fichiers de taille inférieure à 20Mo s'explique par différents facteurs :

- des utilisteurs qui mettent en partage des répertoires de logiciels, avec de nombreux fichiers de petite taille
- des partages de répertoires de photos
- des fichiers qui contiennent des pistes d'albums de musique
- etc...

Enfin, notons que la taille moyenne des fichiers est de 74Mo et que 99% des fichiers pèsent moins de 834Mo.

La distribution des tailles des fichiers est très importante pour déterminer les paramètres des caches (tels que la taille de stockage par exemple) et pour comprendre et interpréter les performances des algorithmes de remplacement des caches.

De la même manière, une autre caractéristique utile à connaître est la durée de vie des fichiers dans l'overlay.

4.4.2 Durée de vie des fichiers

Nous mesurons la durée de vie d'un fichier dans l'overlay en soustrayant le numéro du jour d'apparition au numéro du dernier jour où l'on a observé ce fichier, plus un. Puisque notre expérimentation de supervision a duré 293 jours, la durée de vie des fichiers est comprise entre 1 et 293. La figure 4.3 expose la distribution des durées de vie de l'ensemble des 1,2 millions de fichiers uniques vus. Pour des raisons de lisibilité, les barres pour les valeurs de 1 et 2 jours ne sont pas représentées sur la figure, en effet un total de 941 000 fichiers (soit 80% de l'ensemble des fichiers) ont une durée de vie de 1 ou 2 jours.

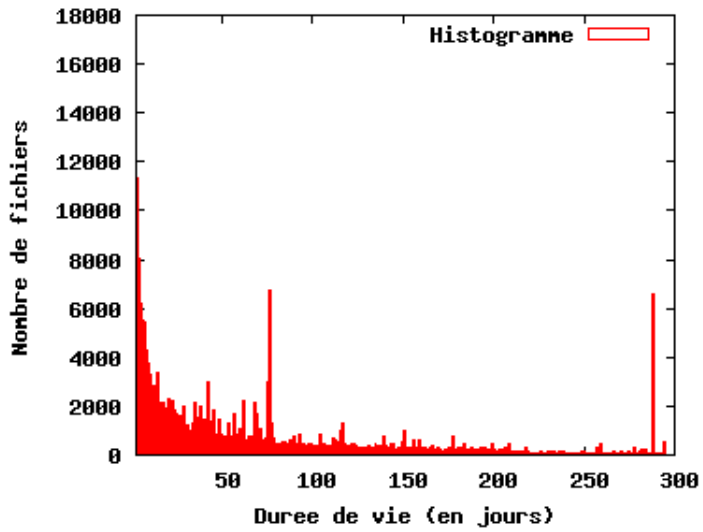


FIGURE 4.3 – Distribution des durées de vie des fichiers

Le pic que l'on peut voir à la valeur 77 est dû au retrait d'une sonde ADSL le 19 janvier 2009, comme expliqué dans la partie 4.3. De même, le pic à la valeur 288 est dû à la mise en place tardive des sondes ADSL par rapport à la sonde FTTH. Ainsi un certains nombre de fichiers observés seulement sur une période réduite voient leur durée de vie tronquée par notre méthode de calcul.

La durée moyenne de vie d'un fichier est de 14 jours et 95% des fichiers ont une durée de vie inférieure à 90 jours.

Maintenant si l'on ne considère que les téléchargements, on obtient une courbe de la même forme, comme illustré sur la figure 4.4. On calcule ici la

durée de téléchargement d'un fichier donné en prenant l'intervalle de temps entre : d'une part le dernier message *SendingPart* dans le sens descendant et d'autre part le premier message *SendingPart* dans le sens descendant observés dans la trace et relatifs à ce fichier.

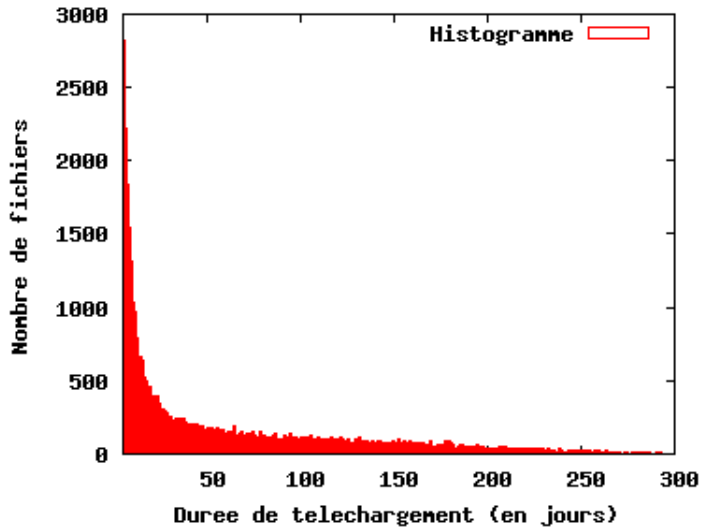


FIGURE 4.4 – Distribution des durées de téléchargement

La durée de téléchargement d'un fichier est en moyenne de 2,33 jours et 95% des fichiers sont téléchargés en moins de 2 jours.

Ces valeurs sont utiles pour la conception des algorithmes de remplacement, en effet nous pouvons observer que la durée de vie de la grande majorité des fichiers ne dépasse pas quelques jours, après quoi ils disparaissent de l'overlay, il devient donc inutile de garder ces fichiers en cache.

4.4.3 Nouveaux clients observés

Nous souhaitons savoir si après environ dix mois de supervision nous avons pu observer tous les clients utilisateurs de eDonkey. A cette fin, nous avons tracé sur la figure 4.5 l'évolution au cours du temps du nombre de clients uniques observés.

La courbe intitulée «Nouveaux clients» représente le nombre de clients observés chaque jour dans la trace, qui n'ont jamais été observés auparavant. Le pic au jour 85 (qui correspond au 3 novembre 2008) s'explique par le fait que deux sondes ADSL n'ont pu être démarrées avant cette date, à cause de contraintes liées au réseau opérationnel. A cause de cette soudaine augmentation du nombre de clients supervisés, nous observons de nombreux

Nombre de jours d'observation	293
Nombre de clients supervisés	18 851
Clients ayant utilisé eDonkey	7 012
Adresses IP utilisées par les clients	36 110
Clients avec une liste de partage	2 582
Clients qui ont téléchargé ou uploadé	6 864
Fichiers uniques observés	1 173 499
Fichiers uniques téléchargés	289 170
Fichier uniques uploadés	600 705
Fichiers uiques dans les listes de partage	551,298
Adresses IP distantes uniques	4 031 149

TABLE 4.1 – Aperçu des données collectées

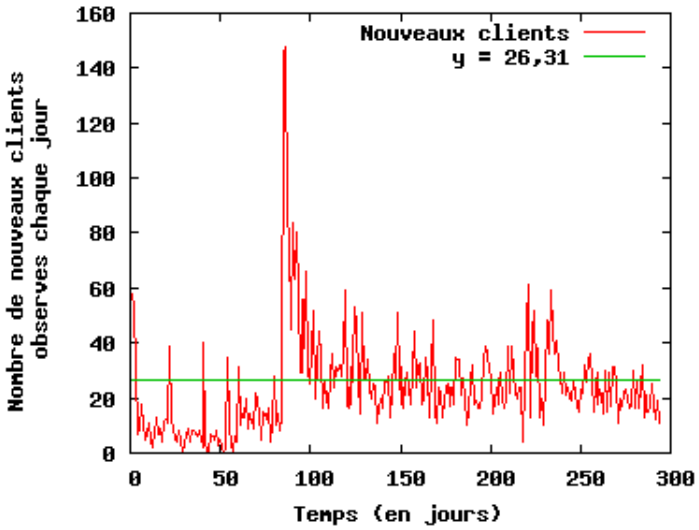


FIGURE 4.5 – Nombre de nouveaux clients chaque jour

nouveaux clients utilisateurs d'eDonkey ce jour là. Si on observe la période entre les jours 94 et 293 nous voyons une stabilisation du nombre de nouveaux clients eDonkey par jour autour de la valeur 26,31. Cette valeur est obtenue par approximation avec la méthode des moindres carrés (algorithme de Marquardt-Levenberg [12]) implémentée dans gnuplot [19].

Ces nouveaux clients représentent des utilisateurs de l'Internet qui font partie des 18 851 clients supervisés et qui ont lancé leur logiciel eDonkey pendant la période d'observation.

4.4.4 Nouveaux fichiers observés

La figure 4.6 représente le nombre de nouveaux fichiers observés chaque jour, en fonction du temps.

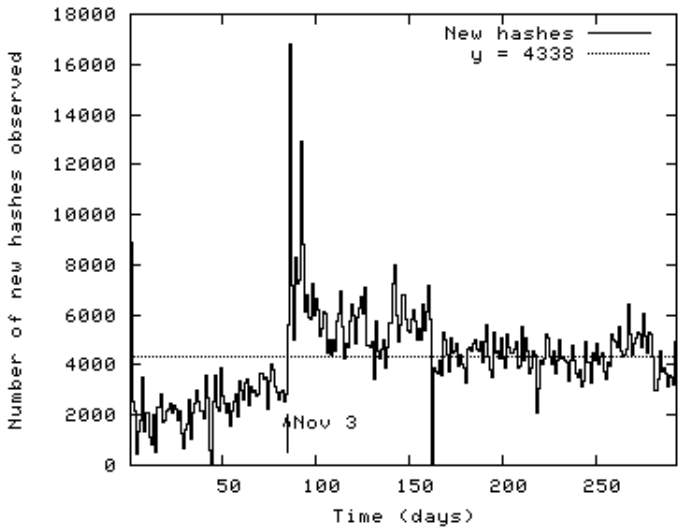


FIGURE 4.6 – Nombre de nouveaux hashs chaque jour

La courbe intitulée «New hashes» représente le nombre de hashs observés chaque jour dans la trace, qui n'ont jamais été observés auparavant. Nous observons sur cette courbe le même pic au jour 85 que sur la figure 4.5, pour les mêmes raisons. Comme précédemment, lorsque le nombre de nouveaux hashs chaque jour se stabilise, nous pouvons mesurer le taux d'apparition de nouveaux hashs sur la période entre le jour 94 et le jour 293, qui correspond à la période sur laquelle le pic du jour 85 a disparu. Nous obtenons ainsi un taux de 4 338 nouveaux hashs chaque jour, avec nos 7 012 clients eDonkey. Cela représente un taux de 0,62 nouveau fichier par jour, par client.

Ce taux élevé d'apparition de nouveaux fichiers est une caractéristique dont il faut tenir compte pour l'algorithme de remplacement dans le cache, afin qu'il soit efficace. Ceci est illustré dans la section 4.5.3 où sont comparés entre eux différents algorithmes de remplacement.

Si nous supposons qu'il existe une relation linéaire entre le nombre de fichiers distincts téléchargés et le nombre de clients (ce qui est vérifié dans nos traces), nous aurions environ 1 650 000 fichiers distincts avec 40 000 clients (sur notre période de 293 jours) et environ 4 124 000 fichiers avec 100 000 clients. Ces chiffres nous seront utiles dans la section 4.6.2 pour évaluer les performances des caches avec un plus grand nombre de clients que n'en contiennent nos trace, par extrapolation.

4.4.5 Popularité des fichiers

La popularité des resources web et P2P (exprimée en tant que nombre d'accès à la resource) a été traditionnellement modélisée avec la loi de Zipf, exprimée par l'equation 4.4.1, avec r le rang d'un fichier et a et b les paramètres de la loi.

$$p(r) = \frac{a}{r^b} \tag{4.4.1}$$

Cependant, Saleh et Hefeeda [127] ont découvert que la popularité des resources P2P est modélisée de manière plus exacte avec la loi de Mandelbrot-Zipf, exprimée dans l'équation 4.4.2, avec les mêmes notations que l'équation 4.4.1 et avec q un paramètre.

$$p(r) = \frac{a}{(r+q)^b} \tag{4.4.2}$$

La figure 4.7 montre la popularité des fichiers observés dans nos traces, exprimée en nombre de clients distincts qui ont téléchargé le fichier. La figure est en échelle log-log.

Nous pouvons observer sur la figure par exemple que le fichier le plus populaire (rang 1) fut téléchargé par 83 clients. Nous avons effectué une approximation des données par la méthode des moindres carrés (algorithme de Marquardt-Levenberg [12]) implémentée dans gnuplot. Il est clair sur la figure que la loi de Mandelbrot-Zipf correspond mieux aux données de nos captures. La loi de Zipf a convergé vers les paramètres $a = 158$ et $b = 0,43$. La loi de Mandelbrot-Zipf a convergé vers les paramètres $a = 270$, $b = 0,48$ et $q = 16,5$.

Si l'on observe la popularité des fichiers en termes d'uploads, on obtient la courbe de la figure 4.8.

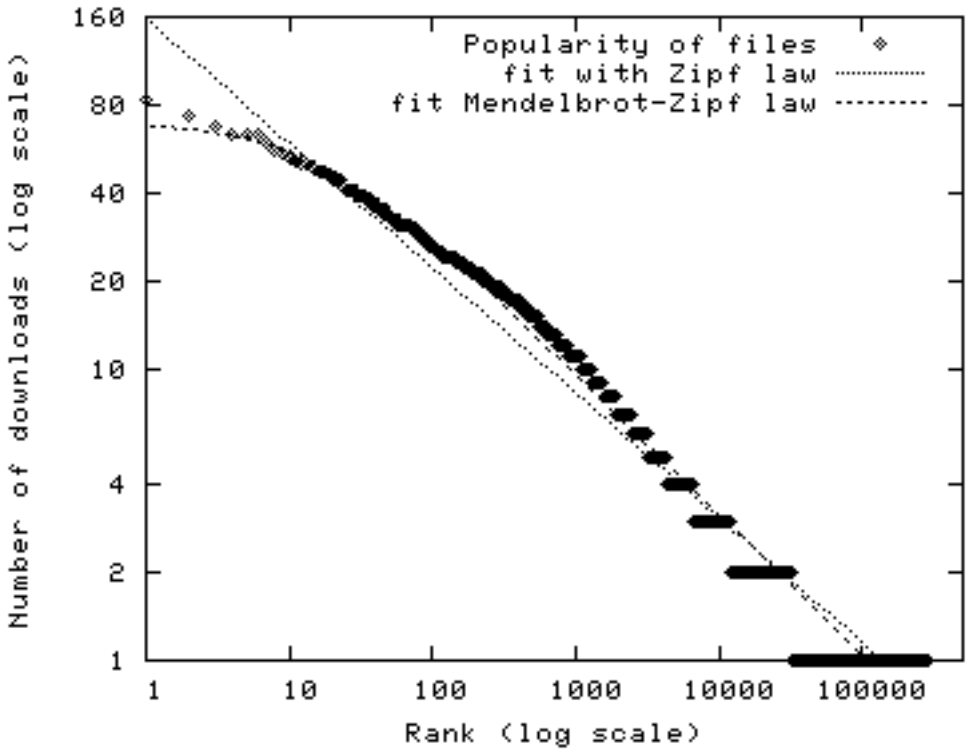


FIGURE 4.7 – Popularité des fichiers (exprimée en nombre de clients distincts ayant téléchargé le fichier)

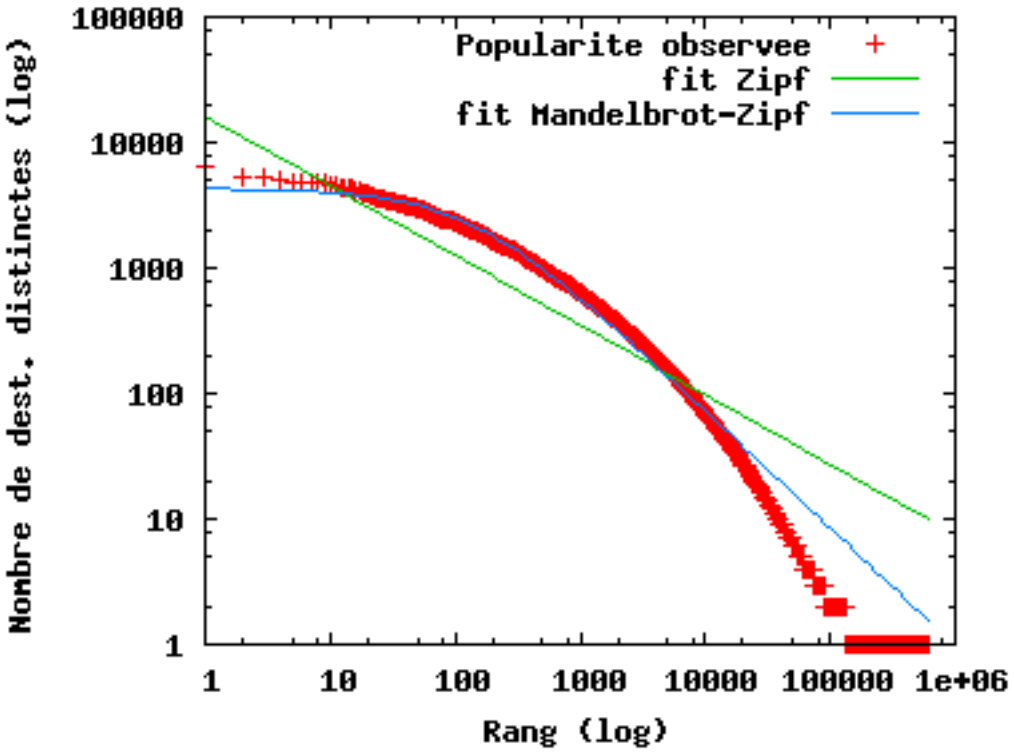


FIGURE 4.8 – Popularité des fichiers en upload (exprimée en nombre de clients distincts ayant uploadés le fichier)

L'approximation par la loi de Zipf a convergé vers les paramètres $a = 16306$ et $b = 0,55$. Pour la loi de Mandelbrot-Zipf on a obtenu $a = 408646$, $b = 0,94$ et $c = 129$. Ici encore on peut observer que la loi de Mandelbrot-Zipf est beaucoup plus adaptée à la description des données observées.

Cela confirme l'observation de Saleh et Hefeeda [127] sur la modélisation de la popularité des objets P2P. Ce modèle peut être utilisé pour estimer la performance théorique d'un cache, comme expliqué dans la section 4.5. Cependant nous allons aussi voir que la réalité est bien différente des résultats obtenus théoriquement.

4.5 Performance des caches

Dans ce qui suit, nous évaluons la performance des caches au moyen de deux valeurs : le ratio de hits et les données servies. Un hit se produit lorsqu'un client télécharge une partie d'un fichier qui est déja dans le cache. Le ratio de hits est le rapport du nombre total de hits par le nombre total de téléchargements de parties de fichier. Les données servies sont la quantité cumulée de données servies par le cache aux clients lorsqu'un hit se produit.

Dans la sous-section 4.5.1, la performance du cache est calculée de manière analytique en utilisant le modèle de popularité défini précedemment. Puis dans la suite de cette section nous allons tirer parti de résultats de simulation basée sur nos captures de vrai trafic eDonkey provenant de vrais utilisateurs.

4.5.1 Bande passante économisée théorique

La bande passante théorique économisée est la quantité de données servie par le cache, moins la quantité de données téléchargée par le cache. Une première façon d'estimer la bande passante est de compter le nombre de fois que chaque fichier observé dans la trace est téléchargé. C'est l'approche adoptée dans la littérature, cf. Hefeeba et Saleh [133] par exemple et elle est basée sur la popularité des contenus de l'overlay. Avec l'hypothèse que les clients téléchargent chaque fichier complètement et seulement une fois, nous pouvons calculer la bande passante économisée si nous avions un cache dans le réseau pour servir les clients supervisés. Dans ce cas la bande passante économisée B serait donnée par l'équation 4.5.2, avec N le nombre total de fichiers observés, $P(i)$ et $S(i)$ respectivement la popularité et la taille du fichier i .

$$B = \sum_{i=1}^{N} P(i)S(i) - \sum_{i=1}^{N} S(i) \tag{4.5.1}$$

$$B = \sum_{i=1}^{N} [(P(i) - 1)S(i)] \tag{4.5.2}$$

La popularité est définie comme le nombre de clients distincts qui ont téléchargé le fichier. L'équation suppose également que dès qu'un fichier a été téléchargé, il est disponible dans le cache pour tous les téléchargements subséquents. Dans l'équation 4.5.1 la première somme représente les données servies par le cache et la deuxième représente les données téléchargées par le cache, lorsque ce dernier est actif.

En supposant toujours que les clients téléchargent les fichiers complètement, le trafic P2P total descendant (i.e. de l'Internet vers les utilisateurs) observé est exprimé par l'équation 4.5.3 ci-dessous, avec les mêmes notations que précédemment.

$$B = \sum_{i=1}^{N} (P(i)S(i) \tag{4.5.3}$$

Pour la distribution des popularités et les téléchargements observés dans nos captures, nous obtenons B = 47 655 Go et T = 97 998 Go. La bande passante économisée représente donc 48,6% du trafic P2P descendant total. Ceci représente la performance théorique du cache en supposant que les clients téléchargent les fichiers complètement. Cependant, en pratique les clients n'ont pas téléchargé les fichiers complètement puisque le trafic P2P descendant était en fait 6 890 Go (et non 97 998 Go). Cette différence importante entre ce que l'on a observé et ce à quoi on pouvait s'attendre s'explique par la conjonction des facteurs suivants :

- les utilisateurs peuvent tout simplement changer d'avis et interrompre un téléchargement ;
- certains fichiers ne sont pas disponibles au complet dans le réseau ;
- bien souvent, il existe plusieurs versions d'un même contenu, les utilisateurs lancent donc plusieurs téléchargements en parallèle, puis les arrêtent dès que l'un d'entre eux est achevé.

Ce point illustre donc bien le fait qu'il faut tenir compte du comportement des utilisateurs dans l'évaluation des caches P2P, car il a un impact important sur les résultats. La popularité des contenus ne suffit pas. Dans la suite nous allons évaluer les performances des caches en nous basant sur nos captures de trafic, donc en tenant compte de ce qu'ont téléchargé réellement les pairs.

4.5.2 Simulation

De manière à bien estimer l'impact d'un cache eDonkey, nous avons implémenté une simulation de cache qui prend en entrée tous les messages *SendingPart* observés dans nos captures du réseau opérationnel. Dans cette simulation le trafic eDonkey descendant de tous nos clients supervisés passe par le cache. Pour chaque message *SendingPart*, le cache décide s'il stocke le contenu du message. Si par la suite un autre client télécharge cette partie de fichier, la simulation considère qu'elle va être transmise du cache. Donc les messages *SendingPart* sont vus par la simulation comme des requêtes. Dans la simulation nous pouvons configurer la taille du cache, ainsi que son comportement : actif ou passif. Un cache actif effectue le téléchargement du restant d'un fichier lorsqu'une partie est téléchargée par un client. Nous supposons dans ce cas que le débit de téléchargement du cache est un paramètre constant. A l'inverse un cache passif ne stocke que les parties de fichier qui ont été effectivement téléchargées par les clients. La séquence des requêtes durant les dix mois de capture est représentative de ce qu'aurait vu un véritable cache s'il avait été déployé dans le réseau.

Cependant cette approche a une limitation à cause de l'aggrégation des données de supervision au moment de leur stockage dans la base de données de capture. En effet nous ne savons pas exactement quelle partie d'un fichier est transportée dans les message *SendingPart*, puisque cette information n'est pas enregistrée. Nous ne connaissons que le nombre de messages. Nous pouvons pallier à cette limitation simplement en formulant la supposition que les pairs eDonkey téléchargent les parties de fichier dans un ordre aléatoire.

Dans la simulation un message *SendingPart* dans nos traces correspond à une requête. La simulation est configurée en donnant le comportement (actif ou passif), la capacité de stockage, le débit de téléchargement (pour un cache actif) et l'algorithme de remplacement.

4.5.3 Algorithmes de remplacement

Quand le cache est plein et qu'une nouvelle requête lui parvient qui nécessite de stocker des données additionnelles, il faut alors supprimer un ou plusieurs objets du cache pour faire de la place. L'algorithme de remplacement détermine quels objets seront supprimés. Nous avons implémenté dans la simulation 4 algorithmes de remplacement : LRU (Least Recently Used), LFU (Least Frequently Used), FILO (First In Last Out) and SIZE. LRU et LFU sont parmi les algorithmes les plus utilisés dans la littérature (cf. Wierzbicki et al. [128] par exemple) et dans les produits commerciaux.

Chaque fois qu'un accès à un fichier est effectué, la date de l'accès est mise

à jour et le compteur d'utilisation est incrémenté. Dans LRU, les fichiers avec une date d'accès la plus ancienne sont supprimés en premier. Pour LFU, ce sont les fichiers avec le compteur d'utilisation le plus faible qui sont éliminés en premier et en cas d'égalité, LRU est appliqué à ces fichiers. L'algorithme FILO ne considère que la date d'arrivée dans le cache pour la sélection et SIZE retire du cache les plus gros fichiers d'abord (en cas d'égalité, LRU s'applique sur les fichiers à égalité). Nous pouvons maintenant évaluer nos algorithmes grâce à notre simulation.

La quantité de données servies par le cache est la métrique la plus importante pour un opérateur car elle est directement liée à la raison d'être d'un cache : économiser la bande passante. C'est donc la raison pour laquelle nous avons choisi de montrer cette métrique sur les figures dans la suite (et pas le ratio de hits).

La figure 4.9 montre la performance d'un cache avec une capacité de stockage de 1 To, avec divers débits de téléchargement.

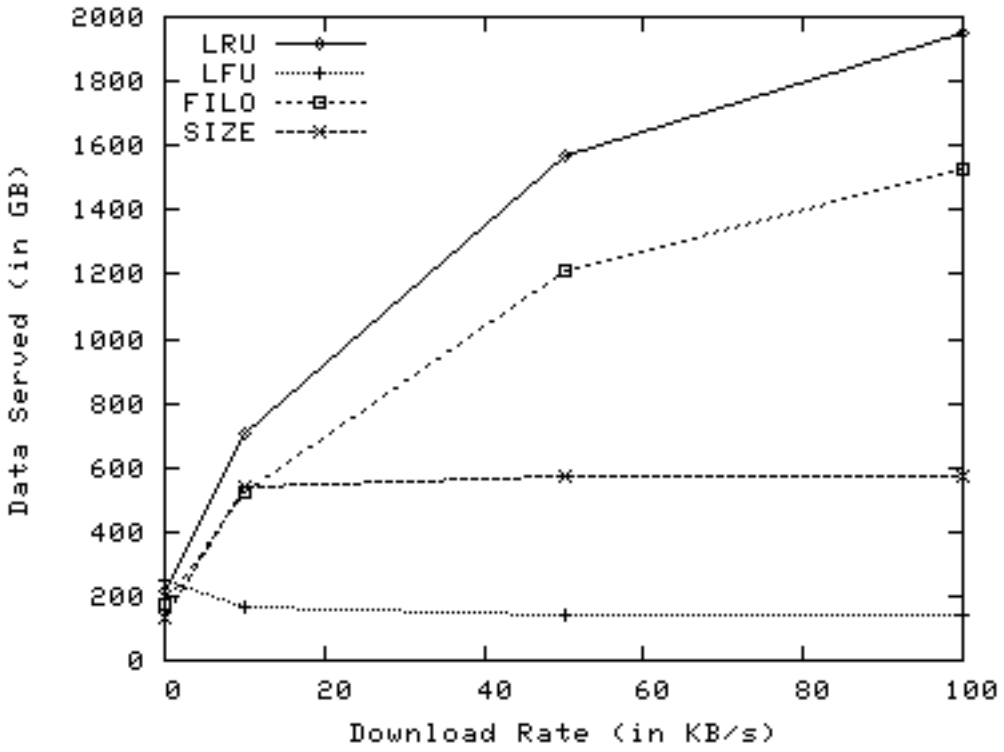


FIGURE 4.9 – Performance d'un cache de taille 1To en fonction du débit de téléchargement

Sur cette figure, un débit de téléchargement de zéro correspond au cas

où le cache est passif, i.e. il n'y a pas de téléchargement. Les données servies par le cache sont toujours entre 220 et 320 Mo et le ratio de hits est aux alentours de 4%, pour tous les algorithmes de remplacement.

Nous pouvons constater sur la figure que les caches actifs ont une bien meilleure performance que les passifs, avec les algorithmes LRU et FILO. De manière surprenante, avec LFU les performances diminuent quand le débit de téléchargement augmente. Ceci est dû au fait qu'une large majorité des fichiers ne sont téléchargés qu'une seule fois. Donc tous les fichiers téléchargés deux fois ou plus ne sont jamais remplacés dans le cache LFU (car il y a toujours un fichier utilisé qu'une seule fois à retirer) et donc le cache se remplis rapidement lorsque le cache est actif. Ceci explique les performances médiocres de LFU pour un cache actif.

4.5.4 Influence de la taille du cache

La figure 4.10 montre la performance de chaque algorithme de remplacement, pour un cache passif, en fonction de la taille du cache.

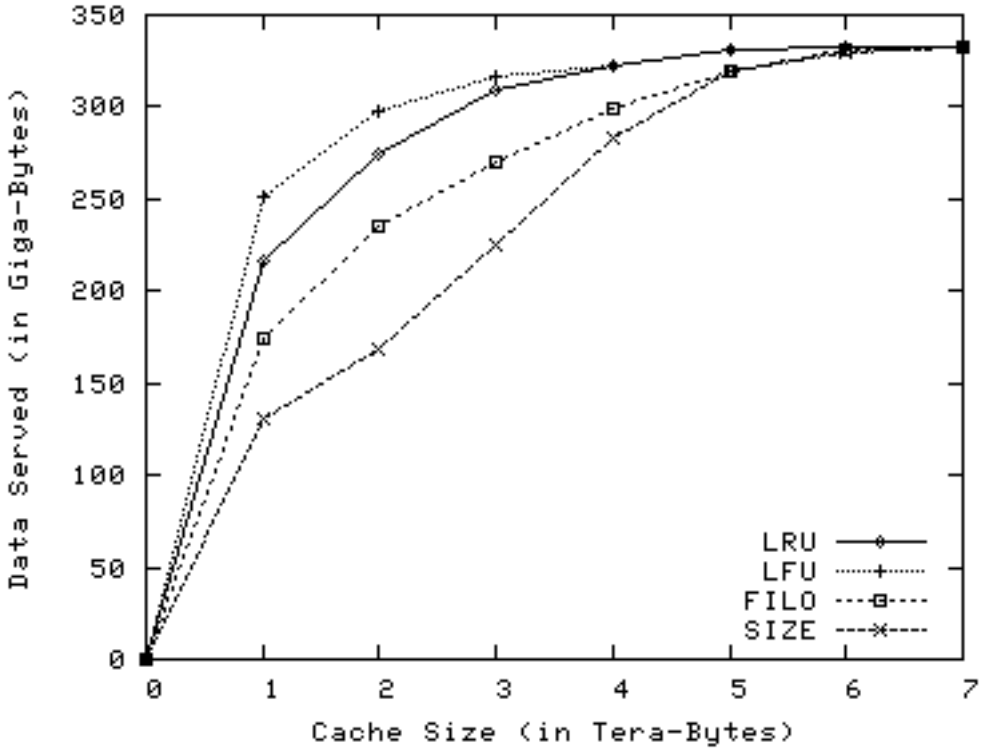


FIGURE 4.10 – Performance d'un cache passif en fonction de la capacité de stockage

Puisque la quantité totale de trafic P2P téléchargé dans nos traces est de 6 890 Go, si la capacité de stockage du cache est supérieure à cette valeur alors le cache peut tout stocker sans avoir besoin de supprimer des fichiers. Avec un cache de taille supérieure ou égale à 7 Go nous avons obtenus une quantité de données servie de 332 Go et un ratio de hits de 4,8%.

Les performances des algorithmes LRU et LFU sont proches et FILO n'est pas très loin derrière. SIZE est plus faible que les autres. Puisque la plupart des fichiers ne sont téléchargés qu'une seule fois, l'algorithme de remplacement utilisé (LRU ou LFU) n'a pas un grand impact pour un cache passif. En revanche la taille du cache est importante car lorsque la taille augmente de 1 To à 2 To, les données servies augmentent de 26% avec LRU et de 18% avec LFU.

4.6 Discussion

Les performances d'un cache P2P sont influencées par les facteurs suivants :

- les caractéristiques de l'overlay et en particulier la distribution des popularités des fichiers ;
- les propriétés du cache : la taille, la politique de remplacement, le débit de téléchargement du cache (pour un cache actif) ;
- le comportement des clients. Comme les utilisateurs peuvent ne pas télécharger complètement les fichiers, connaître la distribution de popularité des contenus ne suffit pas ;
- le nombre de clients. A l'évidence le cache est plus utile s'il peut servir un grand nombre de clients, car dans ce cas le trafic a plus de chance de contenir des redondances.

Les caractéristiques de l'overlay et les propriétés des caches furent étudiées respectivement dans les sections 4.4 et 4.5. L'impact du comportement des utilisateurs sur les performances des caches est refleté dans les résultats de la section 4.5 parce que ces résultats sont obtenus à partir de tranferts de données entre vrais utilisateurs. Cette section va se focaliser sur le nombre de clients servis par le cache.

La sous-section 4.6.1 va discuter de la localisation du cache, car c'est le point qui va déterminer essentiellement le nombre de clients que le cache devra servir. Enfin la sous-section 4.6.2 donne les performances des caches P2P extrapolées à un nombre de clients supérieur à celui de nos traces.

4.6.1 Localisation du cache

En fonction de la localisation du cache dans le réseau, il va servir un nombre variable de clients. Un nombre typique de clients supportés par un cache P2P du commerce se situe aux alentours de 40 000. Ceci va se produire si le cache est déployé en amont d'un BRAS par exemple, car ce dernier gère le trafic d'environ 120 000 clients Internet. Donc si l'on suppose qu'un tiers d'entre eux sont utilisateurs de eDonkey alors le cache va bien servir environ 40 000 clients.

Une autre possibilité est de placer le cache au niveau d'un routeur d'interconnexion. Dans ce cas un nombre typique d'utilisateurs Internet est 300 000, ce qui nous donne environ 100 000 utilisateurs eDonkey.

Plus il y a d'utilisateurs servis par le cache, meilleures sont ses performances, parce qu'il y a plus de trafic redondant dans ce cas. Cependant si leur nombre est trop grand, le cache peut avoir des difficultés à soutenir la charge, mais ceci est en dehors du périmètre traité ici. Dans le suite nous allons étudier l'opportunité de déployer un cache P2P pour 40 000 et 100 000 utilisateurs, en nous basant sur les résultats présentés dans les sections 4.4 et 4.5.

4.6.2 Bande passante économisée

Le but de cette sous-section est d'estimer la bande passante économisée pour les caches actifs et passifs, en faisant varier le nombre de clients servis par le cache.

Caches actifs

Quand le cache est actif, il télécharge complètement les fichiers pour lesquels il y a au moins une requête. En conséquence, les données téléchargées par le cache est la somme des tailles des fichiers demandés. Dans nos traces cette somme est égale à 262 To.

D'un autre côté, la quantité de données servies a pour borne supérieure la quantité de données requise par les clients, qui est égale dans notre cas à 6,9 To. Les données téléchargées par le cache sont donc supérieures aux données servies, par conséquent le cache consomme de la bande passante plutôt que d'en économiser.

Cependant, le ratio de hits est meilleur avec un cache actif, donc ce dernier peut quand même être utile, en permettant des meilleurs débits de téléchargement pour les utilisateurs.

Caches passifs

Avec les caches passifs, il n'y a pas de téléchargement en plus, par rapport à une situation sans cache. La bande passante économisée est donc égale à la quantité de données servie par le cache aux clients.

Pour nos 7 012 clients, nous avons 289 170 hashs téléchargés, ce qui représente 6 890 Go. Ceci représente une moyenne de 24 Mo téléchargés par fichier.

Les données collectées nous permettent de savoir que chaque client, pris séparément, a téléchargé en moyenne 52 fichiers sur la période d'observation. Donc pour 40 000 clients, nous aurions un total de 2 080 000 téléchargements. Grâce à notre analyse de la trace dans la sous-section 4.4.4, l'estimation du nombre de fichiers distincts téléchargés est de 1 650 000. Par conséquent le nombre de téléchargements multiples (i.e. un fichier téléchargé plusieurs fois) est la différence entre ces deux valeurs, ce qui représente 430 000. Ce nombre est une borne supérieure sur le nombre de hits du cache. En effet, un téléchargement multiple n'est pas nécessairement un hit puisque le fichier a pu être retiré du cache entre deux requêtes.

Puisque chaque hit représente une économie de 24 Mo en moyenne, nous obtenons la bande passante économisée estimée pour 40 000 clients : cela représente 10 320 Go servis par le cache. Pour 100 000 clients servis par le cache, cette valeur devient 25 824 Go économisés (soit 21% du trafic eDonkey total).

Comparaison cache actif / passif

Les caches actifs n'économisent pas de bande passante, mais permettent d'avoir un meilleur ratio de hits, autorisant ainsi de meilleurs taux de téléchargement pour les utilisateurs. Cela peut être un choix de l'opérateur qui veut améliorer la qualité de service de ses clients pour certaines applications P2P.

Le choix de déployer un cache, actif ou passif, peut être guidé par des considérations financières, en particulier par rapport au coût de la bande passante aux points d'interconnexion du réseau. Ainsi par exemple certains opérateurs ne sont connectés à l'Internet qu'à travers des communications par satellite, qui sont très couteuses. C'est le cas dans certains pays émergents et certaines zones rurales. Le coût de la bande passante à travers une communication satellite peut atteindre jusqu'à 50 fois celui d'une liaison filaire [132]. Donc dans ce genre de situations, un cache actif n'est pas indiqué, tandis que les gains induits par un cache passif peuvent donner lieu à d'importantes économies sur les coûts d'exploitation opérationnelle du réseau.

4.7 Conclusion

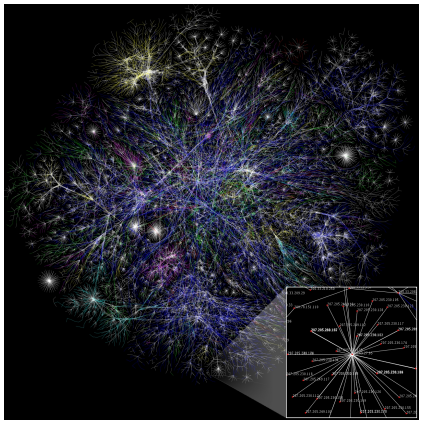
Ce chapitre a décrit une expérimentation de supervision à large échelle conduite dans le réseau opérationnel de France Télécom pendant une période de temps imporante. Pendant cette expérimentation, la charge utile des paquets eDonkey a été analysée afin de décoder les champs du message eDonkey. Ces informations ont été journalisées dans une base de données, de manière totalement anonyme afin de préserver la vie privée et la confidentialité du trafic des clients supervisés. Nous avons analysé les données collectées avec une attention particulière sur les propriétés qui ont un impact sur la performance d'un cache P2P. Nous avons décrit une simulation d'un cache P2P, basée sur les données récoltées, afin de mesurer la performance de ce dernier, en termes de ratio de hits et de données servies. Les différents facteurs influençant la performance du cache furent discutés et évalués.

Pour 100 000 clients, un cache passif aurait pu permettre d'économiser 21% du trafic P2P à l'interconnexion. Bien évidemment si un nombre plus petit de clients est servi, la réduction de bande passante sera inférieure. Ainsi par exemple avec notre échantillon de 7 012 clients, la bande passante économisée totale serait aux alentours de 4%, ce qui est bien moins qu'avec 100 000 clients. C'est normal car moins de clients servis par le cache signifie moins de redondance dans le trafic.

Puisque la proportion de trafic P2P est actuellement autour d'un tiers du trafic total, une réduction de 21% du trafic P2P représente une réduction de la consommation de bande passante globale d'environ 7%. Ce chiffre est moins élevé que ceux donnés dans la litérature. Ceci est dû au fait que dans l'état de l'art, le comportement des clients n'a pas (ou pas assez) été pris en compte, en particulier le fait que les clients peuvent ne télécharger les contenus que partiellement. A l'inverse, nous avons pris ceci en compte puisque nous nous basons sur des transferts réels entre vrais utilisateurs.

Chapitre 5

Optimisation du trafic pair-à-pair avec P4P



L'Internet est constitué d'une multitude de domaines administratifs interconnectés.

5.1 Introduction

Tout comme le chapitre précédent, ce chapitre est fondé sur la supervision du trafic d'échange de fichiers en P2P, donc dans la couche applicative. Grâce aux informations de supervision dans la couche 7, nous pouvons déterminer l'opportunité de la mise en place de fonctions d'optimisation de trafic, spécialisées pour le trafic d'échange de fichiers P2P.

La supervision applicative nous permet ici d'améliorer l'ingénierie du réseau en orientant les choix d'architecture fonctionnelle.

Ce chapitre est organisé comme suit. Tout d'abord le contexte du P4P est présenté (5.2.1), puis un aperçu des travaux précédents est donné, dans

le domaine de la sélection de pairs (5.2.2) et dans l'étude des bénéfices de P4P (5.2.3). Ensuite nous donnons un aperçu des données collectées et nous détaillons certaines caractéristiques des données qui ont une utilité pour la suite de notre étude (5.3). Puis, grâce aux données collectées, nous déterminons l'impact de P4P sur le trafic inter-domaine (5.4). En effet la réduction du volume de trafic aux points d'interconnexion est le principal avantage que les FAI peuvent tirer d'une architecture de type P4P. D'autre part, nous étudions également l'impact pour les utilisateurs des applications P2P. A cette fin, nous déterminons l'influence du nombre et de la proximité des sources sur le débit moyen de téléchargement (5.5). Enfin, la partie 5.6 donne une interprétation des résultats.

5.2 Contexte

Le trafic pair-à-pair (P2P) représente une proportion importante du volume de trafic transitant dans le cœur du réseau, comme montré par Labovitz et al. [18]. Cependant les réseaux P2P sont bien souvent sans connaissance du réseau sous-jacent et n'utilisent donc pas les resources réseau de manière efficiente. A fortiori les réseaux P2P n'ont pas de connaissance des accords de transit des FAI, générant par là peut-être plus de trafic que nécessaire aux points d'interconnexion et en particulier aux points d'interconnexion les plus coûteux pour le FAI (Fournisseur d'Accès à l'Internet).

Le but de ce chapitre est d'évaluer les bénéfices d'une architecture P4P (Proactive Participation of Providers in P2P) à partir de données de supervision du réseau opérationnel.

5.2.1 Qu'est-ce que P4P

P4P est un consortium initié par des études menées à l'Université de Yale [141]. Ces études sont parties du constat que dans la plupart des protocoles pair-à-pair et en particulier les plus utilisés (tels que eDonkey, BitTorrent, KaaZaa ou encore Gnutella), il n'est tenu aucun compte du réseau sous-jacent. Or ces protocoles induisent des charges de trafic non négligeables dans le réseau. Il a donc été proposé d'effectuer la sélection des pairs en prenant en compte divers caractéristiques des réseaux (telles que par exemple les règles d'ingénierie, les accords de peering, l'état du réseau, etc...), en plus des besoins des différents protocoles P2P. Ceci de manière à réguler le trafic P2P de façon efficiente dans les réseaux traversés.

Pour les opérateurs, les bénéfices attendus sont une meilleure répartition de la charge dans le réseau, l'ingénierie et le dimensionnement du réseau qui

doivent être facilités et surtout une réduction du trafic aux points d'interconnexion avec les autres opérateurs.

Pour les utilisateurs, une approche de type P4P doit autoriser une meilleure performance des applications P2P, grâce à la sélection de pairs plus proches, avec un meilleur débit ou une meilleure latence. Ainsi pour une application d'échange de fichiers par exemple, l'amélioration des performances va consister en une réduction du temps de téléchargement des fichiers.

5.2.2 Sélection des pairs

Tout d'abord, il a été démontré par Akella et al. dans [136] que contrairement à une idée assez répandue, les points de congestion dans le réseau ne sont pas nécessairement au niveau du réseau d'accès. Cela va être d'autant plus vrai à l'avenir avec l'émergence et la démocratisation de technologies d'accès large bande telles que FTTH (Fiber To The Home). Donc la sélection des pairs est un problème qu'il ne faut pas négliger car si cela n'est pas effectué correctement, cela peut conduire à des engorgements de certains liens dans le réseau, ou aux points d'interconnexion. Ceci est confirmé par Sen et Wang dans [143], où ils analysent le trafic P2P et son impact sur le réseau sous-jacent. Ils constatent que la charge P2P peut (et doit) être gérée par les opérateurs. De plus, Karagiannis et al. [137] discutent les implications des mécanismes de distribution de contenu P2P pour les opérateurs. Ils montrent que l'émergence de ces systèmes P2P déplace les coûts de distribution de contenu des fournisseurs de contenus vers les opérateurs. Ils montrent également que les solutions P2P qui prennent en compte la localisation des pairs peuvent alléger ces coûts tout en améliorant la performance pour les utilisateurs.

Par la suite, dans [135], Bernstein et al. établissent l'utilité d'avoir un bon algorithme de sélection des pairs dans un système P2P. Ils proposent une stratégie de sélection de pairs basée sur un arbre de décision à apprentissage. Ce problème de sélection de pairs est aussi traité dans [144], où Tang et al. présentent un algorithme qui minimise le nombre de sauts dans l'overlay pour atteindre un pair. Cependant dans ces articles, le réseau sous-jacent n'est pas pris en compte.

Certaines études, comme [145], [146], [147] ou [148], proposent que l'application sonde le réseau afin de prendre en compte certains paramètres réseau en compte dans le choix des pairs. Mais ces solutions sont loin d'être parfaites car sonder le réseau fait porter une charge supplémentaire sur le réseau et les applications n'auront de toute façon qu'une vue incomplète de la topologie, les politiques et l'état du réseau.

Pour résoudre ce problème, l'approche P4P a été proposée. La sous-section

suivante détaille les études effectuées, concernant les bénéfices que l'on peut attendre de cette approche.

5.2.3 Bénéfices apportés par P4P

La solution proposée par P4P comporte une architecture dans laquelle les pairs du système P2P peuvent contacter un élément controllé par l'opérateur, dans le but d'obtenir des informations sur quels pairs distants doivent être contactés en priorité. L'architecture P4P requiert que ceci soit totalement optionnel et à discrétion de l'application si elle veut utiliser P4P ou non, donc P4P ne peut pas en théorie faire baisser les performance des applications P2P. Plus de détails sont disponibles dans les documents de normalisation de l'architecture P4P à l'IETF [141, 142].

Dans le but de déterminer l'apport de P4P, plusieurs études furent menées, sous la forme de simulations, expérimentations à petite échelle sur PlanetLab, ou des expérimentations de terrain à petite échelle. Les résultats que nous présentons ici viennent en complément puisque nous présentons les résultats d'une expérimentation à grande échelle, avec des utilisateurs et du trafic réels. Dans [141], Xie et al. présentent les résultats d'une simulation d'un overlay BitTorrent et les résultats d'une expérimentation avec 160 nœuds PlanetLab. Ils ont également réalisé une expérience dans laquelle des utilisateurs réels téléchargaient un clip vidéo d'une taille de 20 Mo, sur une période de 10 jours. Les auteurs ont observé que la quantité de trafic inter-domaine pouvait être divisée par 4 et le débit de téléchargement moyen augmenté de 23 %, grâce à P4P. De la même manière, dans [138] Aggarwal et al. établissent les bénéfices apportés par P4P au moyen d'une simulation et d'une expérience en laboratoire avec 45 nœuds. Les résultats montrent que le nombre de connexions intra-AS augmente en passant de 15 % à 82 % grâce à P4P.

Ces résultats sont très prometteurs car ils montrent les gains que pourraient obtenir potentiellement à la fois les opérateurs de réseau et les utilisateurs de P2P. Dans le but d'obtenir des résultats complémentaires de ceux présentés dans cette sous-section, nous utilisons l'expérimentation de supervision à grande échelle (avec plus de 7 000 utilisateurs de eDonkey et sur environ 10 mois) mise en place et décrite dans le chapitre 4.3 pour déterminer les bénéfices attendus par P4P.

5.3 Données collectées

La méthodologie de collecte des données est la même que dans 4.3 puisque ce sont les mêmes données de départ qui sont exploitées. Pour rappel, nous supervisons le trafic eDonkey dans trois sondes dans le réseau opérationnel français. Les sondes enregistrent tous les paquets de type *SendingPart* (transmission de données entre deux pairs) et *OfferFiles* (sert à un pair pour annoncer sa liste de fichiers partagés). Les données collectées nous disent donc très exactement qui partage quoi et qui échange avec qui.

En plus de ces données, nous avons déterminé l'AS (Autonomous System) de chaque adresse IP observée dans nos captures. L'AS est déterminé en interogeant le serveur cymru.com [134] grâce au protocole «whois».

5.3.1 Aperçu général

Nous avons donc supervisé un échantillon de 7 012 clients pendant 293 jours, du 15 août 2008 jusqu'au 31 mai 2009, soit une période d'environ 10 mois. La table 5.1 donne un aperçu des données collectées pour les besoins de l'étude réalisée dans ce chapitre.

Nb de jours d'observation	293
Nb de clients supervisés	18 851
Nb de clients utilisant eDonkey	7 012
Nb d'adresses IP utilisées par les clients	36 110
Clients avec liste de partage	2 582
Clients qui ont téléchargé ou uploadé	6 864
Nb de hashs distincts observés	1 173 499
Nb de hashs distincts téléchargés	289 170
Nb de hashs distincts uploadés	600 705
Nb de hashs distincts en partage	551 298
Nb d'adresses IP distinctes	4 031 149

TABLE 5.1 – Résumé des données collectées

Nous voyons dans la table que nous avons observé plus de 4 millions d'adresses IP distinctes. Ces adresses sont réparties parmi 5 818 AS différents. Si nous ne considérons que les adresses IP distantes lors des téléchargements de nos clients supervisés, on obtient au total 3 011 309 adresses IP distinctes, réparties parmi 4 686 AS.

5.3.2 Répartition des adresses sources par AS

Voyons maintenant la répartition de ces adresses de téléchargement dans les différents AS. La table 5.2 donnes la distribution des adresses IP parmi les 10 AS les plus fréquents.

rang	numéro d'AS	Nb d'adresses IP	Proportion
1	3215	710 214	23,5%
2	12 322	510 227	16,9%
3	15 557	427 173	14,1%
4	12 876	118 927	3,9%
5	3 269	99 203	3,2%
6	21 502	85 818	2,8%
7	3 352	72 472	2,4%
8	6 678	64 441	2,1%
9	8 228	48 592	1,6%
10	5 410	27 201	0,9%

TABLE 5.2 – Répartition des adresses IP dans les AS

Nous pouvons observer qu'environ un quart des adresses d'origine des téléchargements viennent du même AS que nos clients supervisés (i.e. l'AS 3215 qui correspond au backbone français de France Télécom). D'autre part, environ la moitié des adresses proviennent d'un des trois FAI français majeurs. Enfin, environ deux tiers des adresses proviennent des 7 AS les plus fréquemment observés.

Ces observations indiquent que les téléchargements sont assez localisés géographiquement, en dépit du fait qu'il n'y ait pas de mécanisme de localisation dans eDonkey. La raison pour cette situation est que les utilisateurs provenant des mêmes zones géographiques ont tendance à vouloir les mêmes contenus. En effet ces utilisateurs partagent bien souvent les mêmes intérêts et le même langage. Si les téléchargements s'effectuent déja de manière locale, à cause de la nature même du service qui s'exécute dans le réseau overlay (échange de fichiers dans notre cas), alors il se peut que P4P n'apporte pas de gros bénéfices aux utilisateurs. Nous allons nous attacher à montrer ce qu'il en est vraiment dans la suite (en particulier dans la sous-section 5.5).

5.3.3 Les pairs et leurs téléchargements

Intéressons nous maintenant aux téléchargements de chaque pair. En moyenne un client supervisé a téléchargé 52 fichiers (pas nécessairement en

entier) sur notre période d'observation. La figure 5.1 illustre comment sont répartis les nombres de fichiers téléchargés pour chaque client.

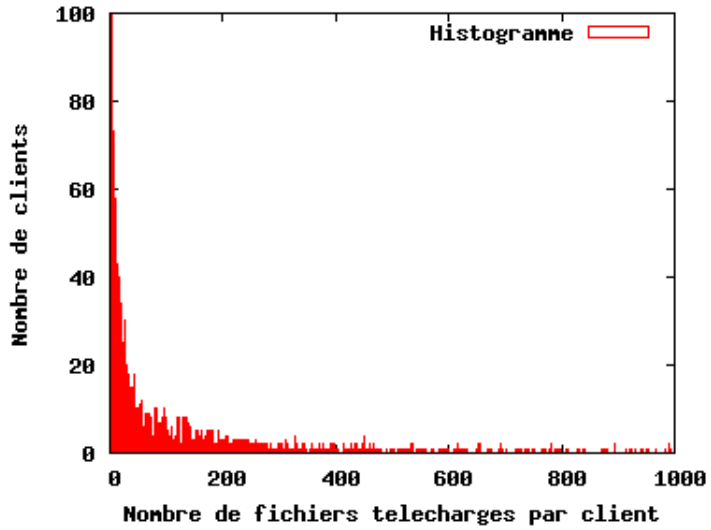


FIGURE 5.1 – Distribution du nombre de fichiers téléchargés par client

Le pic à la valeur d'abscisse 1 n'est pas montré sur la figure pour des raisons de lisibilité. En effet environ 4 700 clients (soit les deux tiers des clients) n'ont téléchargé que 2 fichiers ou moins. A l'inverse un client seul a téléchargé 7 433 fichiers. Il y a donc une grande variabilité du nombre de fichiers téléchargés selon les clients.

Si l'on regarde maintenant le volume téléchargé par client, on observe le même phénomène, puisque la moyenne est de 967 Mo téléchargés par client, mais 4 626 clients (environ les deux tiers) ont téléchargé moins de 1 Mo.

La figure 5.2 montre la distribution du nombre de sources pour chaque fois qu'un téléchargement a été observé.

Pour la large majorité des téléchargements, le nombre de sources est inférieur à 10. La figure ne montre pas la barre d'histogramme pour l'intervalle 0-10 pour des raisons de lisibilité, car il y a plus de 298 000 téléchargements (82% de tous les téléchargements dans nos traces) avec moins de 10 sources. Le nombre de sources pour un téléchargement va de 1 à 1 915 sources et il est en moyenne de 9,1. Seulement 1% des téléchargements ont plus de 110 sources.

La figure 5.3 illustre la distribution des débits de téléchargement.

Les débits moyens de téléchargement d'un fichier vont de 0 à 114 ko/s et leur moyenne sur l'ensemble des fichiers est de 270 o/s. Sur la figure la barre correspondant à l'intervalle 0-100 o/s n'est pas représentée afin de

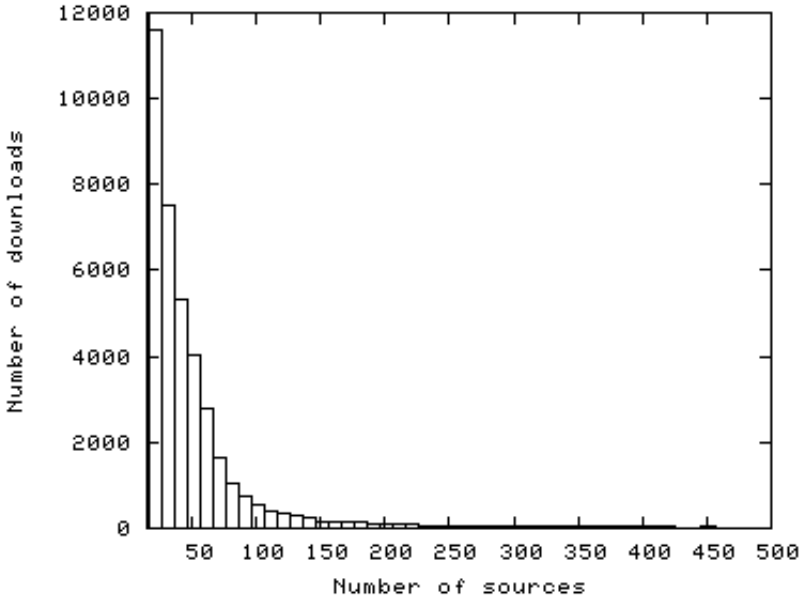


FIGURE 5.2 – Distribution du nombre de sources pour chaque téléchargement

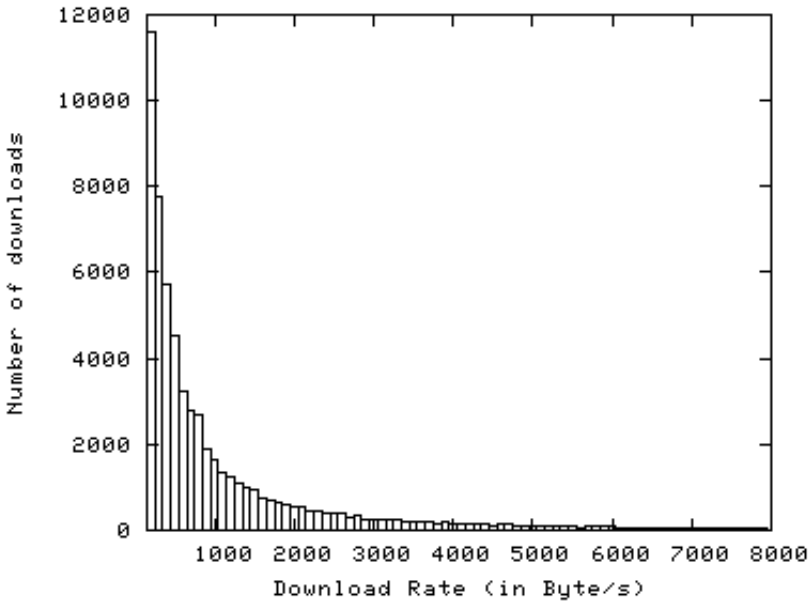


FIGURE 5.3 – Distribution du débit de téléchargement moyen pour chaque téléchargement

conserver la lisibilité de la figure. Seulement 5% des téléchargements ont un débit moyen supérieur à 1 200 o/s. Les faibles valeurs des chiffres obtenus proviennent de la manière dont nous calculons le débit moyen. En effet le temps de téléchargement pris en compte ici est la différence entre la date d'arrivée du premier paquet *SendingPart* et le dernier. Ceci ne correspond pas forcément au temps de téléchargement effectif car les téléchargements peuvent être interrompus, puis repris par la suite.

Les facteurs qui ont un impact sur le débit de téléchargement sont les suivants :

- Comportement de l'utilisateur. L'utilisateur peut éteindre l'application P2P ou bien enlever un fichier de sa liste de partage durant un transfert ;
- Infrastructure utilisateur. La bande passante disponible du pair qui télécharge ou bien de ses sources peut limiter le débit de transfert ;
- Nombre de sources disponibles. Les fichiers avec un nombre de sources limité prennent plus longtemps à télécharger en moyenne. Cette rareté de sources peut être dûe à la rareté du fichier, ou bien au serveur d'indexation qui peut ne connaître qu'une partie des sources disponibles pour ce fichier.

Grâce à ces données, nous allons maintenant pouvoir estimer de manière quantitative les bénéfices de P4P escomptés, d'une part pour le FAI (sous-section 5.4) puis pour les utilisateurs de l'application P2P (sous-section 5.5).

5.4 Impact de P4P sur le trafic inter-domaines

Les bénéfices du P4P attendus par les FAI sont multiples :

- utilisation efficace des resources réseau ;
- contrôle dans une certaine mesure du trafic P2P ;
- ingénierie et dimensionnement du réseau facilités ;
- économie de bande passante aux points d'interconnexion.

Seul ce dernier facteur peut être quantifié de manière précise, nous nous focaliseront donc sur ce point dans la suite.

Dans le cas d'un fonctionnement idéal pour le FAI d'une architecture P4P, les pairs téléchargent le contenu dont ils ont besoin d'une source proche, c'est-à-dire dans le même AS. Par conséquent si un fichier est disponible dans le même AS, les pairs vont sélectionner en priorité les pairs dans le même AS pour le téléchargement. Pour pouvoir estimer la quantité de trafic qui aurait pu être économisée aux points d'interconnexion grâce au P4P, nous allons additionner les téléchargements effectués à l'exterieur de l'AS, pour tous les fichiers disponibles à l'intérieur.

Nous ne savons pas exactement quels fichiers sont disponibles à l'intérieur de l'AS car nous ne supervisons qu'une fraction des pairs de l'AS. Nous pouvons cependant prendre comme approximation que les fichiers disponibles sont ceux partagés par les clients supervisés. Il y a en fait plus de fichiers disponibles en interne de l'AS donc notre supposition va sous-estimer le trafic économisé aux points d'interconnexion.

Nous connaissons la taille des fichiers en partage dans l'AS 3215, dans lequel sont les clients supervisés. Grâce à nos captures nous connaissons également, pour chaque fichier, combien de fois il est téléchargé et de quelles adresses IP, donc de quel AS. Si nous appelons :

- N le nombre de fichiers distincts disponibles en partage dans l'AS 3215,
- S_h la taille en octets du fichier h,
- et D_h le nombre de fois qu'il est téléchargé dans nos captures,

nous obtenons la quantité de trafic Q économisée aux points d'interconnexion.

$$Q = \sum_{h=1}^{N} S_h D_h \tag{5.4.1}$$

Avec les donnéees de nos captures nous calculons $Q = 4970$ Go, ce qui représente 72% de la quantité de trafic téléchargée totale. La table 5.3 résume les résultats de ce calcul.

Trafic total téléchargé	6 895 Go
Hashs distincts télécharchés	289 170
Hashs disponibles dans l'AS 3215	76 784
Trafic économisé par P4P	4 970 Go
Proportion de bande passante économisée	**72%**

TABLE 5.3 – Bande passante économisée aux points d'interconnexion

Nous constatons que dans [142] les auteurs déterminent la trafic inter-domaine économisé au moyen d'une expérimentation et obtiennent une division du trafic par 4, ce qui correspond à 75% de réduction. Ce chiffre est très proche des 72% obtenus grâce à nos captures de trafic réel.

5.5 Impact de P4P sur le débit moyen de téléchargement

Pour les utilisateurs et les développeurs d'applications P2P, les bénéfices attendus du P4P sont une meilleure performance de l'application, c'est-à-dire

dans le cas par exemple d'une application d'échange de fichiers, un temps de téléchargement réduit. Nous quantifions donc les bénéfices du P4P pour les utilisateurs de eDonkey en termes d'amélioration du débit moyen de téléchargement. Le débit moyen de téléchargement est calculé en divisant le nombre de *parties* (fragment de fichier de 10 240 octets) téléchargées par la durée de téléchargement.

Dans les solutions P4P, le principal facteur pris en compte pour la sélection des sources est la proximité, comme expliqué par Seedorf et Burger dans [119] par exemple. C'est pourquoi cette section va se focaliser sur l'impact de la proximité des sources sur les performances de l'application eDonkey.

Dans le but de déterminer l'impact de P4P, nous allons tout d'abord étudier l'impact du nombre de sources (sous-section 5.5.1), puis nous étudierons l'influence de la proximité des sources (sous-section 5.5.2).

5.5.1 Impact du nombre de sources sur le débit de téléchargement

En étude préliminaire, observons la relation entre le nombre de sources pour un téléchargement et le débit moyen de ce téléchargement.

Il est important de noter ici que nous n'avons aucun contrôle sur le nombre de sources pour chaque téléchargement puisque nos résultats proviennent d'une supervision passive du réseau. En revanche, nous effectuons un tri des téléchargements observés dans nos captures en fonction du nombre de sources.

La figure 5.4 représente le débit moyen de téléchargement pour tous les téléchargements de nos captures, lorsque le nombre de sources varie. Les barres d'erreur indiquent les intervalles de confiance à 95%.

Comment lire la figure : nous pouvons lire par exemple en abcisse, à la valeur 300, que le débit moyen est d'environ 3 345 o/s. Cela signifie que pour tous les téléchargements avec un nombre de sources compris entre 300 et 325 (les nombres de sources sont groupés par intervalles de 25), la moyenne des débits est de 3 345 o/s. De plus l'intervalle de confiance à 95% est compris entre 1 684 o/s et 5 007 o/s.

La courbe intitulée «fit ax+b» représente la fonction linéaire la plus proche des données présentées, au sens de l'algorithme des moindres carrés Marqardt-Levenberg, implémenté dans Gnuplot (cf. [12] pour plus de détails). L'équation de cette fonction linéaire est $y = 7,45x + 883$. Ceci montre que les pairs avec un grand nombre de sources total pour leurs téléchargements ont un meilleur taux de transfert, en moyenne. C'est compréhensible car plus l'application a de choix pour sa sélection de sources, meilleure est

sa performance, puisqu'elle pourra plus facilement télécharger en provenance de pairs avec un bon taux de transmission.

5.5.2 Impact de la proximité des sources sur le débit de téléchargement

Afin d'estimer l'impact de la localisation des sources d'un pair sur son débit de téléchargement, nous avons observé pour chaque pair supervisé, pour chacun de ses téléchargements, le débit moyen de téléchargement ainsi que le nombre de sources à l'intérieur et à l'extérieur de l'AS 3215. Notons ici que les clients supervisés sont à l'intérieur de l'AS 3215, donc les sources à l'intérieur de cet AS sont locales.

La figure 5.5 représente le débit moyen de téléchargement de l'ensemble des fichiers par l'ensemble des clients supervisés, en fonction du ratio de sources dans l'AS 3215.

Ainsi par exemple, sur cette figure nous voyons que pour un ratio de sources internes égal à 0,2 (i.e. une source sur cinq était dans l'AS 3215), les téléchargements avaient un débit de 835 octets/s en moyenne. Les barres d'erreur représentent l'intervalle de confiance à 95%.

La droite intitulée «fit ax+b» a la même signification que dans la figure 5.4. Cette droite a pour équation $y = -741x + 977$. La pente est négative, ce qui signifie que si le ratio de sources en interne augmente, le débit moyen de l'ensemble des téléchargements diminue. Ce résultat peut paraître contre-intuitif car le fait de télécharger en provenance de pairs plus proches, du même AS, devrait faire augmenter le taux moyen de téléchargement. Ceci peut s'expliquer par le fait que lorsqu'on choisit de nombreuses sources à l'intérieur d'un même AS, on limite ainsi le nombre de sources disponibles. Par conséquent, en diminuant le nombre de sources disponibles on diminue ainsi le taux de téléchargement moyen observé, comme montré dans la sous-section 5.5.1.

Dans le but de vérifier si ce phénomène se produit aussi avec les fichiers populaires (c'est-à-dire avec de nombreuses sources), nous avons tracé la même figure que la figure 5.5, mais en ne considérant que les 10 fichiers les plus populaires. Populaires signifie ici les fichiers les plus téléchargés. Le résultat est présenté dans la figure 5.6

Ici l'équation de la régression linéaire est $y = -1585x + 1455$. L'échelle de l'ordonnée est la même que sur la figure 5.5, on peut donc observer que les intervalles de confiance à 95% sont plus importants. La raison en est que le nombre de téléchargements est inférieur, ce qui diminue la précision de la moyenne.

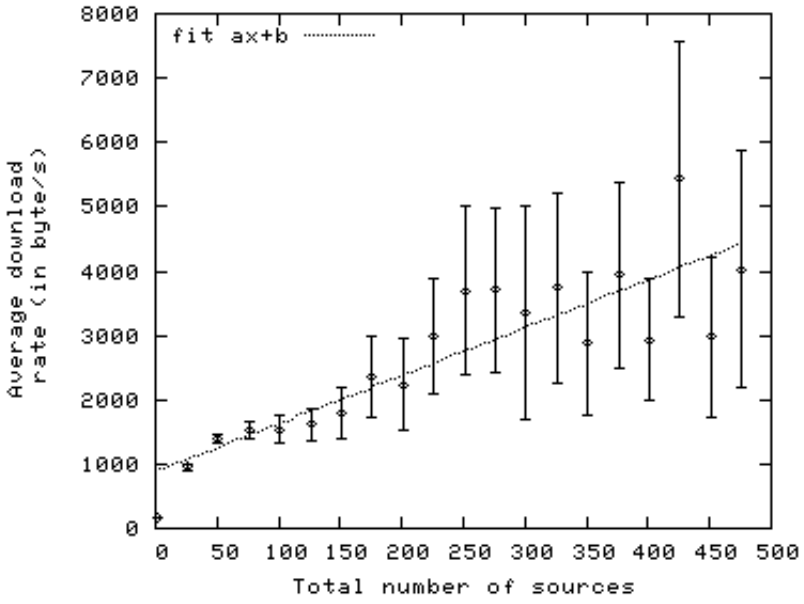


FIGURE 5.4 – Moyenne des débits de téléchargement en fonction du nombre de sources

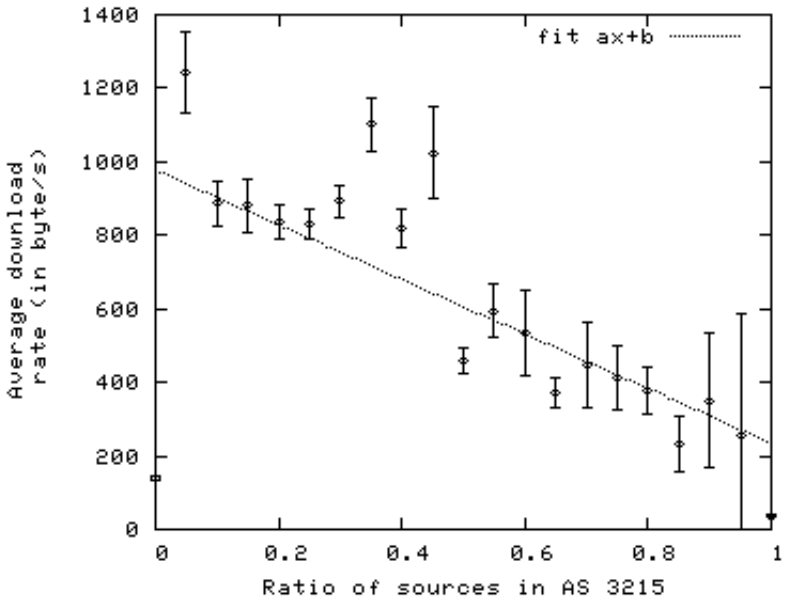


FIGURE 5.5 – Moyenne des débits de téléchargement en fonction du ratio de sources internes

La même conclusion que dans la figure 5.5 s'applique : le fait d'avoir plus de sources en interne dans le même AS signifie que le taux moyen de téléchargement est plus bas en moyenne.

A des fins de comparaison, la figure 5.7 montre la moyenne des taux de téléchargement quand le nombre de sources dans l'AS 12322 varie : l'AS 12322 est le deuxième des AS qui ont le plus d'échanges avec les clients supervisés.

Nous avons également tracé le même type de figure pour les 7 AS qui ont le plus d'échanges avec les clients supervisés (les figures ne sont pas présentées ici). Les mêmes observations que dans la figure 5.5 peuvent être faites, i.e. quand le ratio des sources dans un AS particulier augmente, le taux moyen de téléchargement observé diminue.

La section suivante discute ces résultats.

5.6 Discussion

Dans les données présentées dans la section précédente, on peut observer que certains téléchargements ont une proportion très élevée de sources dans un AS particulier. Par exemple, nous avons observé que 4 % des téléchargements avec 10 sources ou plus se sont effectués avec toutes les sources dans l'AS 3215. Si la sélection de sources était aléatoire, la probabilité d'avoir les 10 sources dans le même AS serait extrèmement faible (au plus $1/4^{10}$, soit moins d'une chance sur 1 million) et en pratique on ne pourrait observer ce cas de figure dans nos capture que sur des cas isolés. La raison de ces cas de figure se trouve dans la manière dont eDonkey sélectionne les sources.

Dans eDonkey les pairs sont agnostiques du réseau sous-jacent et en particulier ils ne considèrent pas l'AS des sources potentielles avant de les contacter pour un transfert. Un pair demande tout d'abord au serveur d'indexation les adresses des pairs distants possédant tout ou partie du ou des fichiers désirés. Puis le pair se place dans les files d'attente d'upload des pairs indiqués comme source par le serveur. Lorsque le pair passe premier dans la file, la source envoie les données du chunk désiré. Dans la pratique il existe de nombreux serveurs d'indexation eDonkey. Certains d'entre eux sont séparés des autres et ne possèdent qu'un petit nombre de clients. Les pairs connectés à ce type de serveur auront un accès à un nombre très limité de sources et ces sources peuvent se trouver localisées dans le même AS. Ceci explique donc les cas de certains téléchargements qui ont une grande proportion de leurs sources dans le même AS. En même temps cela explique également pourquoi le taux de téléchargement moyen est très bas pour les proportions hautes de sources dans un même AS : le nombre de sources disponibles est trop bas.

Le débit entre deux pairs de l'Internet peut être influencé par de nombreux

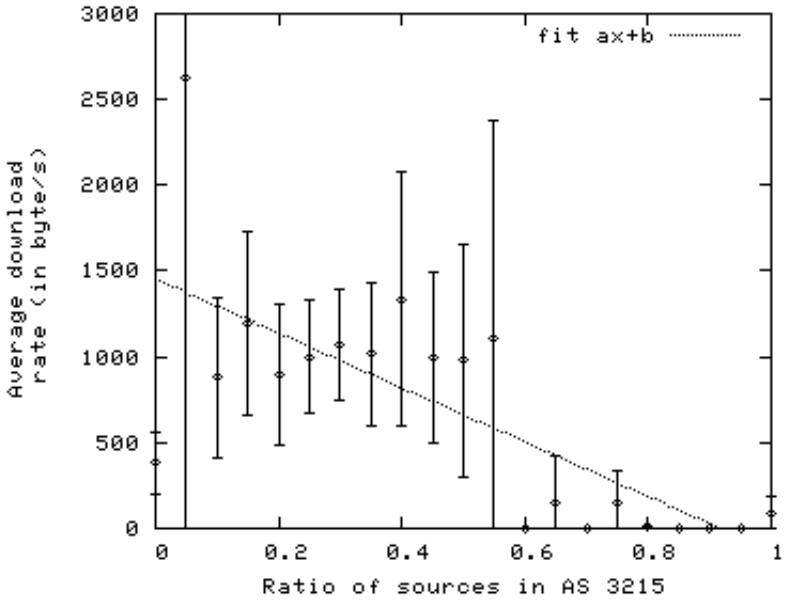


FIGURE 5.6 – Moyenne des débits de téléchargement en fonction du ratio de sources internes (les 10 fichiers les plus populaires)

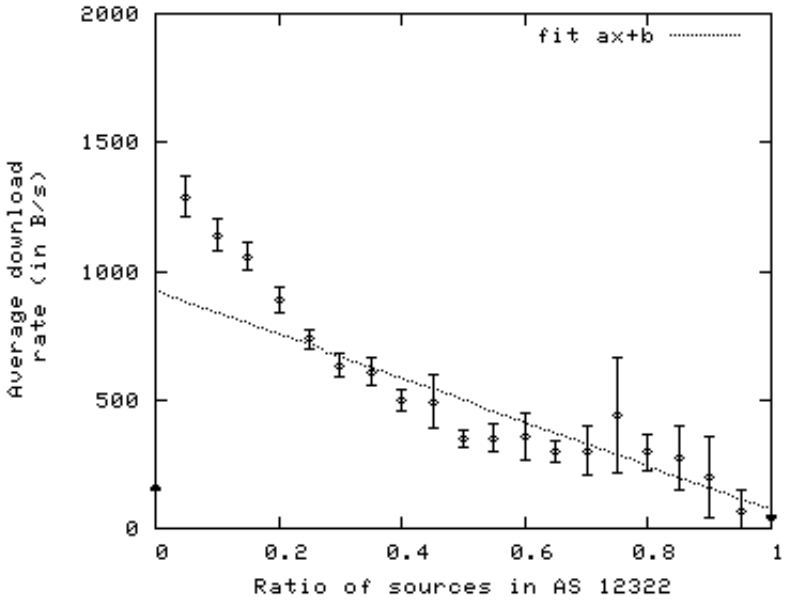


FIGURE 5.7 – Moyenne des débits de téléchargement en fonction du ratio de sources dans l'AS 12322

facteurs, en relation avec les pairs eux-mêmes mais aussi en relation avec les différents réseaux traversés par le trafic entre ces deux pairs. Plus il y a de domaines administratifs à traverser, plus grands sont les risques d'observer une diminution dans le débit de ce trafic. Donc nous nous attendons à ce que le débit entre deux pairs du même AS soit plus élevé en moyenne. Cependant nous avons vu que rester dans le même AS limite le nombre de sources, ce qui diminue le débit de téléchargement moyen.

Enfin, pour conclure sur les résultats concernant les bénéfices de P4P pour les utilisateurs, il nous faut remarquer que les applications P2P pourraient localiser les sources potentielles sans l'aide du FAI, en regardant à quel AS appartiennent leurs adresses IP. Mais il apparaît à travers les résultats présentés dans la sous-section 5.5.2 que connaître l'AS ne suffit pas pour déterminer la proximité des sources potentielles de manière profitable. De nombreux facteurs influencent le débit entre deux pairs donnés. Ceci semble indiquer que le FAI puisse être utile, en fournissant plus de renseignements sur les sources potentielles que seulement l'AS d'appartenance.

5.7 Conclusion

Nous avons mis à profit dans ce chapitre l'expérimentation de supervision à grande échelle décrite dans le chapitre 4.3, dans laquelle nous avons supervisé et journalisé pendant environ 10 mois tous les paquets eDonkey *SendingPart* et *OfferFiles* d'un échantillon de clients. A partir de ces informations, nous avons effectué une analyse a posteriori sur les bénéfices escomptés d'une architecture P4P, à la fois du point de vue de l'opérateur de réseau et du point de vue de l'utilisateur (i.e. des performances de l'application P2P).

D'un côté, l'utilité de P4P pour l'opérateur de réseau est clairement (et même spectaculairement) démontrée par les résultats de notre étude. Le trafic inter-domain peut être grandement réduit aux points d'interconnexion (d'au moins 72 %), lorsque les pairs sélectionnent des sources dans leur AS en priorité.

D'un autre côté, pour les utilisateurs de P2P, l'augmentation attendue de la performance de l'application P2P n'a pas pu être mise en évidence, à cause de la nature a posteriori de notre étude. En effet, puisque le trafic que nous avons supervisé n'utilisait pas P4P, nous ne pouvons pas réellement savoir avec certitude comment les performances de l'application seraient impactées si l'application avait choisi d'autres pairs pour ses téléchargements de chunks. Tout ce que nous avons démontré est que connaître l'AS de provenance des sources potentielles n'est pas suffisant pour améliorer les performances. En effet, deux pairs peuvent être dans le même AS et soit être dans le même

réseau de collecte, soit dans le même réseau d'accès, ou bien ils peuvent aussi être à l'opposé l'un de l'autre dans le backbone. Dans ces différents cas de figure, les débits moyens de téléchargement constatés seraient bien différents les uns des autres. Donc ceci montre que plus d'informations sont requises, des informations dont seulement le FAI dispose, à propos de l'état du réseau, de sa topologie et également à propos des accords de peering et de transit avec les autres FAI.

Chapitre 6

Conclusion

Dans ce chapitre nous dressons le bilan des travaux décrits dans ce document (6.1), puis nous suggérons des travaux futurs qui représentent des pistes de recherche ouvertes et intéressantes (6.2). Enfin nous donnons les perspectives ouvertes par ces travaux, à plus long terme (6.3).

6.1 Bilan

Nous avons dans ces travaux mis en œuvre des techniques de supervision applicative du trafic dans le cœur de réseau.

Tout d'abord nous avons utilisé un outil de reconnaissance du protocole de niveau 7 dans le trafic. Cet outil placé dans des sondes du réseau opérationnel a permis d'affiner notre connaissance de l'usage qui est fait du réseau par les clients. Nous avons pu ainsi pu mieux comprendre, par le moyen d'études épidémiologiques, les facteurs et mécanismes qui ont une influence sur la propagation des logiciels malveillants sur une population de clients Internet à haut-débit.

Ensuite nous avons mis en œuvre un outil de décodage d'un protocole applicatif particulier, eDonkey. Cet outil, également placé dans des sondes du réseau opérationnel, a permis de collecter des données, de manière anonyme, sur une période de temps et un nombre de clients jamais atteints auparavant. Ces données furent ensuite exploitées dans le but de déterminer quantativement l'impact de modifications de l'architecture du réseau. Les modifications d'architecture étudiées sont l'ajout de caches P2P et la collaboration de l'opérateur avec le réseau P2P pour la sélection des pairs, au travers de l'interface P4P.

L'avantage d'effectuer la supervision dans le réseau par rapport à l'effectuer au niveau des postes clients est principalement que l'on a accès au trafic

d'un nombre de clients plus importants. L'avantage apporté par la supervision dans la couche applicative est que l'on peut déterminer avec une grande précision comment les utilisateurs supervisés font usage du réseau : nous avons par exemple obtenu des statistiques sur les différents services Internet utilisés par les clients, ou bien sur les contenus téléchargés dans le réseau P2P. L'inconvénient est que les temps de traitement sont plus importants que pour récolter des statistiques de niveau 3 ou 4, il faut donc réaliser un échantillonnage plus important. Nous avons cependant montré que grâce à la supervision applicative nous pouvons obtenir des informations très utile pour l'ingénierie du réseau, que l'on n'aurait pas pu obtenir autrement.

6.2 Travaux futurs

L'application des techniques épidémiologiques aux systèmes en réseau, avec en particulier les études cas-témoins et les études cas-témoins appariées, a ouvert de nombreuses pistes d'étude sur la propagation des logiciels malveillants dans l'Internet.

En effet une première piste d'études consiste en raffiner la définition d'un cas. Cela peut s'effectuer de multiples façons. On peut par exemple observer les échanges DNS des clients afin de déterminer si ces derniers hébergent un logiciel malveillant. En effet, nous avons vu que les machines hôtes participant à un botnet peuvent être détectées au moyen de leur flux anormal de requêtes DNS. On pourrait également observer les canaux C&C ou encore les fichiers P2P dont on sait qu'ils sont infectés par un logiciel malveillant.

Une autre piste de recherche consiste en la définition de facteurs de risques pertinents pour expliquer les causes des contaminations des machines par des logiciel malveillant.

Comme nous l'avons vu, le trafic P2P représente une part importante du trafic dans les réseaux à l'heure actuelle. Mais nous pouvons également observer une augmentation du trafic de flux vidéo (i.e. «streaming»). Le volume de ce type de trafic prend des proportions très importantes, à tel point que Xavier Niel, le fondateur de Iliad, a critiqué YouTube a cause du volume de trafic que ce site impose sur le réseau de Free [151]. Il pourrait donc être profitable, voire nécessaire, pour les opérateurs de mettre en place des solutions de cache des contenus streaming. De la même manière, les services de vidéo à la demande (VoD, Video on Demand) pourraient bénéficier de l'installation de caches pour l'amélioration du service et l'allègement du trafic en certains points du réseau.

Cependant les contenus dans les sites web de streaming et les catalogues de VoD ont des distributions de popularité bien différentes des contenus des

réseau P2P d'échange de fichiers. Il serait donc intéressant et utile d'étudier ces distributions dans ces deux cas. Par la suite, on peut chercher à déterminer les performances du cache, en termes de «hit ratio» (nombre de requêtes servies par le cache divisé par le nombre de requêtes total) ou bien de «byte ratio» (nombre d'octets servis par le cache divisé par le volume total de données requises). Ces performances peuvent dans certains cas être déterminées analytiquement en fonction de la loi de distribution des popularité et de l'algorithme de remplacement. Dans ce cas, la supervision du trafic au niveau applicatif va servir à fournir des ordres de grandeur pour les paramètres de la loi de popularité.

En revanche, lorsque la distribution de popularité ne suit pas une loi simple, ou lorsque l'algorithme de cache est complexe, alors il n'est plus possible de déterminer analytiquement les performances. Il devient nécessaire de réaliser des simulations afin de pouvoir prédire les performances des cache – et donc ainsi déterminer l'opportunité de leur déploiement opérationnel.

D'autre part, il faut pouvoir tenir compte de l'architecture globale du déploiement du cache, qui va influencer ses performances. Dans ce contexte, il est utile d'avoir à disposition des traces qui vont permettre de connaître le trafic. Ensuite, à partir de ces traces, nous pouvons déterminer les performances des caches grâce à des simulations. Nous pouvons également évaluer l'impact des paramètres du cache ainsi que les choix d'architecture sur sa performance. Cette étude devrait pouvoir conduire à des recommendations sur le déploiement opérationnel.

6.3 Perspectives

L'opérateur du réseau, grâce à son accès privilégié aux équipements du réseau et son expertise, est un acteur primordial dans les modèles économiques d'exploitation des services Internet. Cependant son rôle ne doit pas se résumer à fournir des liens à la bande passante de plus en plus élevée.

En effet, à l'avenir, la consommation en bande passante des utilisateurs de l'Internet devrait continuer sa progression fulgurante. Les accès à très haut-débit se démocratisent et d'importants investissement sont consentis par les opérateurs européens pour le développement de la fibre optique résidentielle (FTTH, Fiber To The Home). Les services commencent à s'adapter à ce surcroît de bande passante à la disposition des particuliers, avec par exemple la diffusion de la télévision ou de la VoD (Video on Demand) en haute définition ou en 3D. D'autres services, peut-être encore plus consommateurs de bande passante, vont certainement voir le jour dans un avenir proche. D'après Cisco [152], le trafic IP global annuel devrait atteindre 600 exa-octets (600.10^{18}) en

2013, soit une croissance annuelle de 40%.

Nous voyons donc que les réseaux vont arriver à saturation dans un avenir proche et nous pouvons observer que les services s'adaptent à la bande passante disponible, condamnant ainsi les réseaux à une évolution constante. Dans ce contexte, tous les moyens pour gérer le trafic de manière à économiser la bande passante en certains points critique du réseau seront profitables. Nous pensons que prendre en compte les caractéristiques et les requis des applications qui génèrent le trafic que l'on peut observer dans le réseau est essentiel afin de gérer de manière efficace ce trafic.

Remerciements

Je remercie chaleureusement mes tuteurs de thèse Ludovic Mé, Hervé Debar et Yvon Gourhant.

Je remercie également mes collègues pour leur aide, leur compréhension et/ou les conversations enrichissantes que nous avons pu échanger : Pierre Paris, Bertrand Mathieu, François Bougant, Maël Thouvenot, Carine Toham, Patrick Andrieux, Inès Doghri, Véronique Ruen, Denis Barbaron, Nassima Khiat, Riadh Kortebi, Sara oueslati, Philippe Olivier, Eric Gourdin, Olivier Klopfenstein, Jean-Mathieu Segura, Fabien Mathieu, Luca Muscariello, Christine Gabet.

Enfin, je remercie les rapporteurs de ce mémoire Philippe Owezarski et André-Luc Beylot, ainsi que le président du jury César Viho.

Annexe A

Illustrations

1. *Page 6, télégraphe*
   Source : Edward Henry Knight (1884), dans «Knight'S New Mechanical Dictionary ; A Description Of Tools, Instruments, Machines, Processes, And Engineering», page 722. Houghton, Mifflin and Co.
   Licence : domaine publique
2. *Page 12, photo du musée des sciences de Boston*
   Source : photo par Shannon Bullard
   Licence : «Creative Commons»
   (cf. http ://fr.wikipedia.org/wiki/Creative_Commons)
3. *Page 36, John Snow*
   Source : en.wikipedia
   Licence : domaine publique
4. *Page 58, la mascotte eMule*
   Source : eMule
   Licence : GPL (GNU General Public License)
5. *Page 83, représentation de l'Internet*
   Source : Wikimedia Commons
   Licence : «Creative Commons»

Annexe B

Liste des alertes Snort du chapitre 3

Un total de 809 alertes Snort furent observées dans les 3 traces complètes utilisées dans le 3. Le nombre d'alertes distinctes est de 51. Nous donnons ici la liste des 20 alertes les plus fréquentes, qui représentent à elles seules 92% du total des alertes.

```
[1:721:8] VIRUS OUTBOUND bad file attachment
[1:2417:3] FTP format string attempt
[1:7567:2] SPYWARE-PUT Trackware funwebproducts
           mywebsearchtoolbar-funtools runtime detection
[1:1767:6] WEB-MISC search.dll access
[1:2923:3] NETBIOS SMB repeated logon failure
[1:1288:9] WEB-FRONTPAGE /_vti_bin/ access
[1:1769:3] WEB-MISC .DS_Store access
[1:6238:1] SPYWARE-PUT Adware lop runtime detection - collect
           info request 1
[1:5964:1] SPYWARE-PUT Hijacker searchfast detection - track user
           activity \& get 'relates links' of the toolbar
[1:5806:1] SPYWARE-PUT Hijacker searchmiracle-elitebar runtime
           detection
[1:6236:1] SPYWARE-PUT Adware lop runtime detection - pass info
           to server
[1:4485:3] NETBIOS SMB-DS spoolss AddPrinterEx unicode little
           endian overflow attempt
[1:2376:3] EXPLOIT ISAKMP first payload certificate request
           length overflow attempt
[1:326:10] FINGER remote command execution attempt
```

```
[1:5846:4] SPYWARE-PUT Trickler VX2/DLmax/BestOffers/Aurora
           runtime detection
[1:5850:1] SPYWARE-PUT Adware warez_p2p runtime detection - check
           update
[1:11837:2] SMTP MS Windows Mail UNC navigation remote command
            execution
[1:1091:10] WEB-MISC ICQ Webfront HTTP DOS
[1:2393:1] WEB-PHP /_admin access
[123:2:1] (spp_frag3) Teardrop attack
```

Annexe C

Gestion des contextes TCP

Un message eDonkey peut être réparti sur plusieurs paquets IP. Les applications de bout en bout se reposent sur la pile TCP du système d'exploitation pour reconstituer le flux TCP, en revanche la sonde Otarie ne voit que les paquets un par un et ne garde pas en mémoire de contexte pour les flux TCP. Cela pose donc un problème car dans un flux TCP, certains paquets IP peuvent être perdus puis retransmis, ou bien les paquets peuvent arriver dans le désordre. Il faut donc garder en mémoire un contexte pour pouvoir ré-ordonner les paquets et gérer les retransmissions.

Dans ce but, nous avons ajouté quatre fonctions en C au code source de la sonde Otarie : une pour créer un contexte TCP, une pour chercher un contexte existant lorsqu'on voit un nouveau paquet IP arriver, une pour stocker les données d'un paquet dans un contexte et enfin une fonction pour chercher le contexte le plus ancien afin de le supprimer lorsque c'est nécessaire.

Un contexte est défini par une adresse et un port source et une adresse et un port destination.

C.1 Création d'un contexte TCP

```
struct OFContext newOFContext(
    // source address of the TCP flow
    unsigned long srcA,
    // source port of the TCP flow
    unsigned int srcP,
    // destination address of the TCP flow
    unsigned long dstA,
    // destination port of the TCP flow
    unsigned int dstP,
```

```
    // size of "Offer file" message (without edk header)
    int size,
    // first part of the message "Offer files"
    unsigned char* data,
    // size of the first part of the message
    int sizePart,
    // expected next sequence number of the source
    unsigned int nextSN,
    // timestamp of the packet
    double ts
    )
{
  struct OFContext newContext;
  int i;
  newContext.srcAdd = srcA;
  newContext.srcPort = srcP;
  newContext.dstAdd = dstA;
  newContext.dstPort = dstP;
  newContext.msgSize = size;
  for (i = 0 ; i < sizePart ; i++)
    {
      newContext.message[i] = *(data+i);
    }
  newContext.offset = sizePart;
  newContext.nextSeqNum = nextSN;
  newContext.ts = ts;
  return newContext;
}
```

C.2 Trouver un contexte existant

```
//
// returns the index of the context (srcA, srcP, dstA, dstP)
// returns -1 if not found
//
static int findOFContext (
   unsigned long srcA, // source address of the TCP flow
   unsigned int srcP,  // source port of the TCP flow
   unsigned long dstA, // destination address of the TCP flow
   unsigned int dstP   // destination port of the TCP flow
```

```
   )
{
int i;
for (i = 0 ; i < MAX_NB_OFFER_FILES_CONTEXTS ; i++)
{
if ( (context[i].srcAdd == srcA) && (context[i].srcPort == srcP)
     && (context[i].dstAdd == dstA) && (context[i].dstPort == dstP) )
return i;
}
return -1;
}
```

C.3 Ajout des données d'un paquet dans un contexte existant

```
int addOFPacket(
      unsigned char *data, // data to add
      int size,            // size of data to add
      int index,           // index of the context
        // to which the data is added
      double ts            // timestamp of new packet
      )
{
  int i;
  int offset = context[index].offset;

if (size + offset > MAX_SIZE_OFFER_FILES_MESSAGE) return -1;
for (i = 0 ; i < size ; i++)
{
context[index].message[offset+i] = *(data+i);
}
context[index].offset += size;
context[index].nextSeqNum += size;
context[index].ts = ts;
return 1;
}
```

C.4 trouver le contexte le plus ancien

```
static int oldestOFContext()
{
  int i;
  double minTs = context[0].ts;
  int minTsIndex = 0;

  for (i = 0 ; i < MAX_NB_OFFER_FILES_CONTEXTS ; i++)
    {
      double ts = context[i].ts;
      if ( ts == 0 ) {
return i; // we will not find smaller than 0
      }
      else
if ( ts < minTs )
  {
    minTs = ts;
    minTsIndex = i;
  }
    }
  if (eDonkeyDebug) {
    fprintf(FileDonkey, "yyyydebug: no more available context,
     replacing nb %i (%.1f", minTsIndex, context[0].ts);
    for (i = 1 ; i < MAX_NB_OFFER_FILES_CONTEXTS ; i++)  {
      fprintf(FileDonkey, " %.1f", context[i].ts);
    }
    fprintf(FileDonkey, ")\n");
  }
  return minTsIndex;
}
```

Annexe D

Base de données pour stocker les paquets eDonkey

Les paquets eDonkey observés lors de l'expérimentation de supervision sont stockés dans une base de données relationnelle SQL (Simple Query Language) de la manière décrite dans cette annexe. Afin de pouvoir accéder rapidement aux données selon les différents usages que l'on souhaite en faire, nous avons créé 7 tables dans la base.

La table D.1 répertorie les clients observés ainsi que les différentes adresses IP utilisées par ces derniers. Ces adresses IP ne sont utilisées qu'a des fins de copmtage pour établir des statistiques, et en aucune façon pour identifier les clients. Le type *VARCHAR(100)* est une chaîne de caractères de taille variable et de longueur inférieure à 100.

Nom du champ	Type	Commentaire
id	VARCHAR(100)	Identifiant de couche 2
ip	VARCHAR(20)	

TABLE D.1 – TABLE *client*

La table D.2 enregistre pour chaque hash, le nombre de *parties* distinctes observées (incorrectement appelées chunk dans la table) ainsi que les différents noms de fichiers. Le type *CHAR(32)* représente une chaîne de 32 caractères de taille fixe. Le type *INT* représente un entier signé codé sur 32 bits.

La table D.3 stocke les informations relatives aux téléchargements, i.e. qu'ont téléchargé les clients supervisés. Le type *TINYINT* représente un entier signé compris entre -128 et 127 (codage sur 8 bits). Une entrée dans cette table correspond à un client ayant téléchargé un certain nombres de *parties* d'un hash.

Nom du champ	Type	Commentaire
hash	CHAR(32)	
nb_chunk	INT	
nom	VARCHAR(230)	

TABLE D.2 – TABLE *nbChunk*

Nom du champ	Type	Commentaire
client_id	VARCHAR(100)	
hash	CHAR(32)	
mois	TINYINT	
jour	TINYINT	
heure	TINYINT	
nbDl	INT	Nb de *parties* téléchargées en 3H

TABLE D.3 – TABLE *aTelecharge*

La table D.4 est le pendant de la table D.3, elle stocke en effet les informations relatives aux uploads.

Nom du champ	Type	Commentaire
client_id	VARCHAR(100)	
hash	CHAR(32)	
mois	TINYINT	
jour	TINYINT	
heure	TINYINT	
nbUl	INT	Nb de *parties* uploadées en 3H

TABLE D.4 – TABLE *aUploade*

La table D.5 enregistre les informations relatives aux partages de chaque clients. Ces informations proviennent des messages *OfferFiles* observés.

La table D.6 liste les fichiers Otarie qui ont déja été rentré dans la base. Il y a un fichier Otarie pour chaque tranche de 3H et pour chaque BRAS.

Enfin la table D.7 sauvegarde les adresses distantes pour les téléchargements et les uploads de chaque client et chaque hash. On a une entrée dans cette table pour chaque entrée soit dans la table *aTelecharge* soit dans *aUploade*.

Nom du champ	Type	Commentaire
client_id	VARCHAR(100)	
hash	CHAR(32)	
mois	TINYINT	
jour	TINYINT	
heure	TINYINT	
taille	INT	Taille du hash dans le message *OfferFiles*

TABLE D.5 – TABLE *partage*

Nom du champ	Type	Commentaire
mois	TINYINT	
jour	TINYINT	
heure	TINYINT	
bas	VARCHAR(10)	Nom du BRAS de provenance

TABLE D.6 – TABLE *fichierOtarie*

Nom du champ	Type	Commentaire
bas	VARCHAR(10)	
client_id	VARCHAR(100)	
hash	CHAR(32)	
sensMontant	BOOLEAN	vrai si sens montant
ip	VARCHAR(20)	

TABLE D.7 – TABLE *adressesDistantes*

Bibliographie

[1] *IP (Internet Protocol), IETF RFC 791*
http ://www.ietf.org/rfc/rfc791.txt (accédé en mars 2010).

[2] *ICMP (Internet Control Message Protocol), IETF RFC 792*
http ://www.ietf.org/rfc/rfc792.txt (accédé en mars 2010).

[3] *TCP (Transmission Control Protocol), IETF RFC 793*
http ://www.ietf.org/rfc/rfc793.txt (accédé en mars 2010).

[4] *«Napster's High and Low Notes»* Business Week Online, August 14, 2000 issue. http ://www.businessweek.com/2000/00_33/b3694003.htm (accédé en mars 2010).

[5] *International Organization for Standardization*
http ://en.wikipedia.org/wiki/International_Organization_for_Standardization (accédé en mars 2010).

[6] *Le modèle OSI (Open Systems Interconnection)*,
http ://fr.wikipedia.org/wiki/Modèle_OSI (accédé en avril 2010).

[7] *"Application layer Packet Classifier for Linux»*
http ://l7-filter.sourceforge.net (accédé en mars 2010).

[8] *SNORT, open source network intrusion prevention and detection system (IDS/IPS)*, http ://www.snort.org (accédé en mars 2010).

[9] *Sourcefire*, société à l'origine du développement de Snort.
http ://www.sourcefire.com/company (accédé en avril 2010).

[10] *p0f - passive OS fingerprinting tool*
http ://lcamtuf.coredump.cx/p0f.shtml (accédé en mars 2010).

[11] *OpenEpi*
http ://www.openepi.com (accédé en mars 2010).

[12] *«A Method for the Solution of Certain Non-Linear Problems in Least Squares»*, K. Levenberg. The Quarterly of Applied Mathematics 2 : 164–168, 1994.

[13] *GUID (Globally Unique Identifier)*
http ://fr.wikipedia.org/wiki/Globally_Unique_Identifier (accédé en mars 2010).

[14] *MTU (Maximum Transmission Unit)*
http ://fr.wikipedia.org/wiki/Maximum_Transmission_Unit (accédé en mars 2010)

[15] *John Snow, 1813–1858*
http ://fr.wikipedia.org/wiki/John_Snow (accédé en mars 2010).

[16] *Internet Assigned Numbers Authority (IANA)*
http ://www.iana.org/ (accédé en mars 2010)

[17] *C.S. Institute, Federal Bureau of Investigation*, in : Proceedings of the 10th Annual Computer Crime and Security Survey 10, 2005, pp. 1–23

[18] *«Internet Traffic and Content Consolidation»*, C. Labovitz, I. Johnson, D. McPherson, J. Oberheide, F. Jahanian. Presentation at IETF, March 2010. http ://www.ietf.org/proceedings/10mar/slides/plenaryt-4.pdf (accédé en 2010).

[19] *Gnuplot, utilitaire de dessin de graphes*, http ://www.gnuplot.info/ (accédé en avril 2010).

[20] *«Assessing the accuracy of using aggregated traffic traces in network engineering»*, L. Janowski, P. Owezarski, Telecommunication System Journal, October 2009.

[21] *IPFIX (IP Flow Information Export), IETF RFC 5470*
http ://www.ietf.org/rfc/rfc5470.txt (accédé en mars 2010).

[22] *«Netflow v9, IETF RFC 3954»* - http ://www.ietf.org/rfc/rfc3954.txt (accédée en mars 2010)

[23] *«Contribution de la métrologie Internet à l'ingénierie des réseaux»*, P. Owezarski, rapport d'habilitation à diriger les recherches.

[24] *Carte DAG (Data Acquisition and Generation)*
http ://en.wikipedia.org/wiki/DAG_Technology (accédé en mars 2010).

[25] *«Correlation of Intrusion Symptoms : an Application of Chronicles»*, B. Morin, H. Debar., Proc. of 6th International Conference on Recent Advances in Intrusion Detection (RAID'03), 2003.

[26] *«M4D4 : a Logical Framework to Support Alert Correlation in Intrusion Detection»*, Benjamin Morin, Ludovic Mé, Hervé Debar and Mireille Duccassé. Information Fusion. Volume 10, Issue 4, pp. 285–299. Elsevier. October 2009.

[27] *«Automated Classification of Network Traffic Anomalies»*, Guilherme Fernandes et Philippe Owezarski. Proc. of SecureComm'09, pp. 91–100, 2009.

[28] *«A serial combination of anomaly and misuse IDSes applied to HTTP traffic»*, E. Tombini, H. Debar, L. Mé, M. Ducassé, in Proceedings of the 20th Annual Computer Security Applications Conference, (ACSAC'04), 2004.

[29] *«Spam and the ongoing battle for the inbox»*, J. Goodman, G. Cormack, D. Heckerman. Communications of the ACM 50, 2 (2007), pp. 24–33.

[30] *«Tutorial on junk email filtering»*, E. Hulton, J. Goodman. Tutorial in the 21st International Conference on Machine Learning (ICML) (2004).

[31] *«Detecting spammers with SNARE : spatio-temporal network-level automatic reputation engine»*, S. Hao, N.A. Syed, N. Feamster, A.G. Gray, S. Krasser. 18th USENIX Security Symposium, Montreal, Canada, août 2009.

[32] *SpamHaus IP Blocklist*. http ://www.spamhaus.org (accédé en mars 2010).

[33] *SpamCop*. http ://www.spamcop.net (accédé en mars 2010).

[34] *«A learning approach to spam detection based on social networks»* LAM, H., AND YEUNG, D. . In 4th Conference on Email and Anti-Spam (CEAS) (2007)

[35] *«Code Red Worm Propagation Modelling and Analysis»*, C. C. Zou, W. Gong, and D. Towsley, in 9th ACM Conference on Computer and Communication Security, Nov 2002.

[36] *«Virus Throttling»*, M.M. Williamson, J. Twycross, J. Griffin, A. Norman, Virus Bulletin March 2003, pp. 8–11.

[37] *«The emerging threat of peer-to-peer worms»*, N. Khiat, Y. Carlinet, N. Agoulmine, MonAM'06 (Monitoring, Attack Detection and Mitigation), Tubingen, September 2006.

[38] *«An Inquiry into the Nature and Causes of the Wealth of Internet Miscreants»*, J. Franklin, V. Paxson, A. Perrig, and S. Savage, in proceedings of ACM CCS, October 2007.

[39] *«A network security monitor»*, L.T. Heberlein, G.V. Dias, K.N. Levitt, B. Mukherjee, J. Wood et D. Wolber. Proc. IEEE Symp. Research in Security and Privacy, pp. 296–304, May 1990.

[40] *«GrIDS - a graph based intrusion detection system for large networks»*, S. Staniford-Chen et al. In Proceedings of the 19th National Information Systems Security Conference, volume 1, pages 361–370, October 1996.

[41] *«Bro : A System for Detecting Network Intruders in Real-Time»*, Vern Paxson, Proc. of RAID '98 workshop, vol. 31, no 23-24, 1998, pp. 2433–2487.

[42] *«Practical Automated Detection of Stealthy Portscans»*, S. Staniford, J.A. Hoagland et J.M. McAlerney. Journal of Computer Security, Volume 10, Issue 1-2 (2002), Pages 105-136.

[43] *«Connection-history Based Anomaly Detection»*, T. Toth and C. Kruegel, in Proceedings of the IEEE Workshop on Information Assurance and Security, June 2002.

[44] *«The earlybird system for the real-time detection of unknown worms»*, S. Singh, C. Estan, G. Varghese et S. Savage, Technical Report CS2003-0761, Aug. 2003.

[45] *Empreinte de Rabin*
http ://en.wikipedia.org/wiki/Rabin_fingerprint (accédé en mars 2010).

[46] *«Design of a System for Real-Time Worm Detection»*, B. Madhusudan, J. Lockwood. Proceedings of the 12th Annual IEEE Symposium on High Performance Interconnects, 2004. Pages 77-83, 2004.

[47] *«Internet Threat Detection System Using Bayesian Estimation»*, Masaki Ishiguro, Hironobu Suzuki, Ichiro Murase, Hiroyuki Ohno, FIRST 16th Annual Conference, Budapest, June 2004.

[48] *«Anomalous Payload-based Network Intrusion Detection»*, K. Wang S.J. Stolfo, Sept 2004. Columbia University. Proceedings of International symposium on recent advances in intrusion detection, vol. 3224, 2004, pp. 203–222.

[49] *Distance de Mahalanobis*
http ://fr.wikipedia.org/wiki/Distance_de_Mahalanobis (accédé en mars 2010).

[50] *«Fast Portscan Detection Using Sequential Hypothesis Testing»*, Jaeyeon Jung, Vern Paxson, Arthur W. Berger et Hari Balakrishnan. Proc. of IEEE symposium on security and privacy 2004, pp. 211-225.

[51] *«Scan Detection Based Identification of Worm Infected Hosts»*, C. Göldi et R. Hiestand. Thesis Report, Swiss Federal Institute of Technology, Zurich, 18 April 2005.

[52] *«Entropy Based Worm and Anomaly Detection in Fast IP Networks»*, Arno Wagner et Bernhard Plattner. Proceedings of the 14th IEEE International Workshops on Enabling Technologies : Infrastructure for Collaborative Enterprise, pages 172-177, 2005.

[53] *Le ver WITTY*
http ://www.caida.org/research/security/witty/ (accédé en mars 2010).

[54] *«An Effective Architecture and Algorithm for Detecting Worms with Various Scan Techniques»*, Jiang Wu, Sarma Vangala, Lixin Gao, Kevin Kwiat. Journal of Computer Security, Volume 14, Issue 4 (July 2006), pages 359-387.

[55] *«Analysis of computer infection risk factors based on customer network usage»*, Yannick Carlinet, Ludovic Mé, Hervé Debar, Yvon Gourhant. Securware'08, Cap Esterel août 2008.

[56] *«Bayesian event classification for intrusion detection»*, C. Kruegel, D. Mutz, W. Robertson, F. Valeur. In Proc. of the 19th Annual Computer Security Applications Conference, 2003.

[57] *«A novel anomaly detection scheme based on principal component classifier»*, M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, L. Chang, in Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop, 2003, pp. 172–179.

[58] *«Efficient intrusion detection using principal component analysis»*, Y. Bouzida, F.e.e. Cuppens, N. Cuppens-Boulahia, S. Gombault, in Proceedings of the 3ème Conférence sur la Sécurité et Architectures Réseaux (SAR), Orlando, FL, USA, 2004.

[59] *«Identifying intrusions in computer networks with principal component analysis»*, W. Wang, R. Battiti, in The First International Conference on Availability, Reliability and Security, Vienna, Austria, 2006, pp. 270–279.

[60] *«Détection de vers P2P passifs»*, N. Khiat, Y. Carlinet, N. Agoulmine, SAR'08 (Sécurité des Architectures Réseaux et des Systèmes d'Information), Loctudy, Octobre 2008.

[61] *«A taxonomy of DDoD defense mechanisms»*, J. Mirkovic, P. Reiher. ACM SIGCOMM Computer Communication review 2004, volume 34(2), pp. 39–53, 2004.

[62] *«Inferring Internet denial-of-service activity»*, David Moore, Colleen Shannon, Douglas J. Brown, Geoffrey M. Voelker, Stefan Savage. ACM Transactions on Computer Systems (TOCS), volume 24, Issue 2 (May 2006) pp. 115–139, 2006.

[63] *«A survey of coordinated attacks and collaborative intrusion detection»*, C.V. Zhou, C. Leckie, S. Karunasekera. Computers & Security volume 29 (2010), Elsevier, pp. 124–140, 2010.

[64] *«Botnets - The Silent Threat»*, D. Barroso, ENISA position paper, November 2007. http ://www.enisa.europa.eu/act/res/other-areas/botnets/botnets-2013-the-silent-threat (accédé en mars 2010).

[65] *«Botnets as a vehicle for online crime»*, N. Ianelli, A. Hackworth, CERT Request for Comments (RFC) 1700, December 2005.

[66] *«Home page du projet WASTE»*
http ://waste.sourceforge.net (accédé en mars 2010).

[67] *«A Multifaceted Approach to Understanding the Botnet Phenomenon»*, Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, Andreas Terzis. Proc. of Internet Measurement conference (IMC), pages 41-52, 2006.

[68] *Le pot de miel Nepenthes*
http ://nepenthes.carnivore.it (accédé en mars 2010).

[69] *Le testbed PlanetLab*
http ://www.planet-lab.org (accédé en mars 2010).

[70] *«A Survey of Botnet and Botnet Detection»*, M. Feily, A. Shahrestani, S. Ramadass, in Proc. of Third International Conference on Emerging Security Information, Systems and Technologies (SECURWARE), pp. 268–273, 2009.

[71] *«Botnet Detection and Response, The Network is the Infection»*, D. Dagon, in DNS Operations, Analysis, and Research Center (DNS-OARC) Workshop, 2005.

[72] *«An Algorithm for Anomaly-based Botnet Detection»*, James R. Binkley et Suresh Singh, Computer Science, PSU, USENIX SRUTI'06, 2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet, July 7, 2006.

[73] *«Revealing botnet membership using dnsbl counter-intelligence»*, A. Ramachandran, N. Feamster et D. Dagon, in Proc. 2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI'06), 2006.

[74] *«Detecting Botnets with Tight Command and Control»*, W. Timothy Strayer, Robert Walsh, Carl Livadas, David E. Lapsley. Proc. of the 31st Annual IEEE Conference on Local Computer Networks (LCN) 2006, pp. 195–202.

[75] *«Botnet Detection Based on Network Behavior»*, W. Strayer, D. Lapsley, B. Walsh et C. Livadas, Advances in Information Security. Springer, 2008, pp. 1–24.

[76] *«A Proposal of Metrics for Botnet Detection Based on Its Cooperative Behavior»*, M. Akiyama, T. Kawamoto, M. Shimamura, T. Yokoyama, Y. Kadobayashi, S. Yamaguchi. In proc. of International Symposium on Applications and the Internet (SAINT) 2007.

[77] *«Wide-scale botnet detection and characterization»*, A. Karasaridis, B. Rexroad et D. Hoeflin, in Proc. 1st Workshop on Hot Topics in Understanding Botnets, 2007.

[78] *«Botnet Detection by Monitoring Group Activities in DNS Traffic»*, H. Choi, H. Lee, H. Lee, and H. Kim, in Proc. 7th IEEE International Conference on Computer and Information Technology (CIT 2007), 2007, pp. 715–720.

[79] *«Rishi : Identify bot contaminated hosts by irc nickname evaluation»*, J. Goebel et T. Holz, in Proc. 1st Workshop on Hot Topics in Understanding Botnets, 2007.

[80] *«Ngram-based text categorization»*, W. B. Cavnar et J. M. Trenkle. In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, pp. 161–175, Las Vegas, US, 1994.

[81] *«Flow-based identification of botnet traffic by mining multiple log file»*, M. M. Masud, T. Al-khateeb, L. Khan, B. Thuraisingham, K. W. Hamlen, in Proc. International Conference on Distributed Frameworks & Applications (DFMA), 2008.

[82] *«Botminer : Clustering analysis of network traffic for protocol- and structure independent botnet detection»*, G. Gu, R. Perdisci, J. Zhang et W. Lee, in Proc. 17th USENIX Security Symposium, 2008.

[83] *«Botsniffer : Detecting botnet command and control channels in network traffic»*, G. Gu, J. Zhang, and W. Lee, in Proc. 15th Annual Network and distributed System Security Symposium (NDSS'08), 2008.

[84] *«Flow clustering using machine learning techniques»*, A. McGregor, M. Hall, P. Lorier et J. Brunskill. In C. Barakat and I. Pratt, editors, Proc Fifth International Workshop on Passive and Active Network Measurement (PAM 2004), volume 3015 of LNCS, pages 205-214.

[85] *«Class of Service Mapping for QoS : A Statistical Signature Based Approach To IP Traffic Classification»*, M. Roughan, S. Sen, O. Spatscheck et N.G. Duffield. Proceedings of the Internet Measurement Conference 2004, Taormina, Sicily, Italy, October 25-27, 2004.

[86] *«Traffic classification using a statistical approach»*, D. Zuev, A.W. Moore. Proc. of the 6th International workshop on passive and active network measurement (PAM'05), vol. 3431, pp. 321–324, 2005.

[87] *«BLINC : Multilevel traffic classification in the dark»*, T. Karagiannis, K. Papagiannaki et M. Faloutsos. ACM SIGCOMM 2005, 35 (4), pp. 229–240.

[88] *«Early Application Identification»*, L. Bernaille, R. Teixeira et K. Salamatian, in proceedings of CoNEXT, December 2006.

[89] *«On Inferring Application Protocol Behaviors in Encrypted Network Traffic»*, C. V. Wright, F. Monrose, G. M. Masson, in Journal of Machine Learning Research 7 (2006) p. 2745-2769.

[90] *«On the Validation of Traffic Classification Algorithms»*, G. Szabó, D. Orincsay, S. Malomsoky, I. Szabó, in Lecture Notes in Computer Science, Passive and Active Network Measurement, Volume 4979, pp. 72–81, 2008.

[91] *«Internet Traffic Classification Demystified : Myths, Caveats, and the Best Practices»*, H. Kim, K.C. Claffy, M. Fomenkova, D. Barman, M. Faloutsos, K.Y. Lee, proc. of ACM CoNEXT 2008.

[92] *«A survey of techniques for internet traffic classification using machine learning»*, T. T. T. Nguyen, G. Armitage. In Communications Surveys & Tutorials, IEEE, Vol. 10, No. 4. (2008), pp. 56–76.

[93] *«KISS : Stochastic Packet Inspection for UDP Traffic Classification»*, A. Finamore, M. Mellia, M. Meo, D. Rossi, in Antonio Pescape and Carlo Sansone (Editor), RECIPE. Robust and Efficient Traffic Classification in IP Networks, Fridericiana Editrice Universitaria, Napoli, July 2009.

[94] *«Stochastic Packet Inspection for TCP Traffic»*, G. La Mantia, D. Rossi, A. Finamore, M. Mellia, M. Meo, IEEE International Conference on Communication (ICC), Cape Town, South Africa, May 2010.

[95] *«A Traffic Identification Method and Evaluations for a Pure P2P Application»*, S. Ohzahata, Y. Hagiwara, M. Terada, and K. Kawashima, 6th International Workshop on Passive and Active Measurements 2005, Boston, MA, USA, 2005.

[96] *Winny, logiciel d'échange de fichiers en P2P*
http ://fr.wikipedia.org/wiki/Winny (accédé en mars 2010).

[97] *«Protecting Free Expression Online with Freenet»*, I. Clarke, S. G. Miller, T. W.Hong, O. Sandberg, B. Wiley. IEEE INTERNET COMPUTING, pp. 40–49, janvier-février 2002.

[98] *«Finding Peer-to-Peer File-Sharing Using Coarse Network Behaviors»*, M. P. Collins, M. K. Reiter, in proceedings of ESORICS 2006, pp. 1–17.

[99] *«Comparing P2PTV Traffic Classifiers»*, A. Finamore, M. Mellia, F. Risso, N. Cascarano, L. Salgarelli, A. Este, F. Gringoli, International Conference on Communication (ICC), Cape Town, South Africa, May 2010.

[100] *«Lightweight, Payload-Based Traffic Classification : An Experimental Evaluation»*, F. Risso, A. Baldini, M. Baldi, P. Monclus, O. Morandi, IEEE International Conference on Communications (ICC 2008) - Advances in Networks & Internet Symposium, pp. 5869–5875, May 2008.

[101] *«Support Vector Machines for TCP traffic classification»*, A. Este, F. Gringoli, L. Salgarelli, Elsevier Computer Network, 53(14), pp. 2476–2490, Sept. 2009.

[102] *«An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol»*, S. A. Baset, H. G. Schulzrinne. Proceedings In INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings (2006), pp. 1–11.

[103] *«eDonkey Protocol v0.3»*, A. Klimkin, 2002 http ://cpansearch.perl.org/src/KLIMKIN/P2P-pDonkey-0.01/doc/eDonkey-protocol (accédé en mars 2010).

[104] *«The eMule Protocol Specification»*, Y. Kulbak, D. Bickson, rappotr technique de la Hebrew University of Jerusalem, Janvier 2005.

[105] *«Computer Viruses Theory and Experiments»*, F. Cohen, Computers and Security 1987, vol. 6, pp. 22–35.

[106] *«Biological Models of Security for Virus Propagation in Computer Networks»*, Sanjay Goel and Stephen F. Bush, ;Login : Dec 2004.

[107] *«Kolmogorov Complexity Estimates For Detection Of Viruses In Biologically Inspired Security Systems : A Comparison With Traditional Approaches»*, Sanjay Goel and Stephen F. Bush, Complexity Journal, vol. 9, issue 2, Nov-Dec 2003, Special Issue : «Resilient & Adaptive Defense of Computing Networks".

[108] *«Genetically Induced Communication Network Fault Tolerance»*, Stephen F. Bush, Complexity Journal, vol. 9, issue 2, Nov-Dec 2003, Special Issue : «Resilient & Adaptive Defense of Computing Networks".

[109] *«Concepts of Epidemiology, an integrated introduction to the ideas, theories, principles and methods of epidemiology»*, Raj Bhopal. Oxford University Press.

[110] *«Computers and epidemiology»*, Kephart et al. IEEE Spectrum. v30 n5, May 93, pp. 20–26.

[111] *«Statistique épidémiologie»*, Thierry Ancelle. Edition Maloine.

[112] *«An efficient program for computing conditional maximum likelihood estimates and exact confidence limits for a common odds ratio»*, D. Martin, H. Austin. Epidemiology 2, 359-362 (1991).

[113] *«Methods in Observational Epidemiology 2nd Edition»*, J. L. Kelsey, A. S. Whittemore, A. S. Evans, W. D. Thompson. Oxford University Press, USA (1996).

[114] *«Directed-graph Epidemiological Models of Computer Viruses»*, J. O. Kephart and S. R. White, in Proc. of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy, May 1991, pp. 343–359.

[115] *«The application of epidemiology to computer viruses»*, W.H. Murray, Computers and Security, vol. 7, pp. 130–150, 1988.

[116] *«Epidemiological models of P2P viruses and pollution»*, R. Thommes and M.J. Coates, In Proceedings of IEEE INFOCOM, April 2006.

[117] *«Statistical Methods for Rates and Proportions»*, J. Fleiss, B. Levin, M. Cho Paik. Published by Wiley-Interscience.

[118] *«Verizon Gets Cozy With P2P File-Sharers»*, Associated Press, http ://www.usatoday.com/tech/news/techinnovations/2008-03-14-verizon-p2p_N.htm (accédé en mars 2010).

[119] *«Application-Layer Traffic Optimization (ALTO) Problem Statement»*, J. Seedorf, E. Burger. IETF RFC 5693, december 2009, cf. http ://tools.ietf.org/html/rfc5693

[120] *«Accurate, scalable in-network identification of p2p traffic using application signatures»*, S. Sen, O. Spatscheck, D. Wang. Proceedings of the 13th international conference on World Wide Web, 2004, pp. 512–521.

[121] *«A Study on Traffic Characteristics Evaluation for a Pure P2P Application»*, S. Ohzahata, K. Kawashima. Proceedings of 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing, 2008, pp. 483–490.

[122] *«Identification and Analysis of Peer-to-Peer Traffic»*, M. Perenyi, T. Dang, A. Gefferth, and S. Monlnar. In Journal of Communications 2006, Mars 2006.

[123] *«Revealing Skype Traffic : when randomness plays with you»*, D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli. In ACM SIGCOMM 2007, Aug 2007.

[124] *«Peer sharing behaviour in the eDonkey network, and implications for the design of server-less file sharing systems»*, S. B. Handurukande, A.-M. Kermarrec, F. Le Fessant, L. Massoulié, S. Patarin. Proceedings of the 2006 EuroSys conference, volume 40, Issue 4 (October 2006).

[125] *«The bittorrent P2P file-sharing system : measurements and analysis»*, J. A. Pouwelse, P.Garbacki, D. H. J. Epema, H. J. Sips. 4th International Workshop on Peer-to-Peer Systems (IPTPS) 2005.

[126] *«Understanding the Properties of the BitTorrent Overlay»*, A. Al-Hamra, A. Legout, C. Barakat. INRIA Technical Report, 2007.

[127] *«Modelling and Caching of Peer-to-Peer Traffic»*, O. Saleh, M. Hefeeda. Proceedings of IEEE International Conference on Network Protocols (ICNP), pp. 249–258, 2006.

[128] *«Cache Replacement Policies Revisited : The case of P2P Traffic»*, A. Wierzbicki, N. Leibowitz, M. Ripeanu, R. Wozniak. Proceedings of IEEE

International Symposium on Cluster Computing and the Grid (CCGrid), pp. 182–189, 2004.

[129] *«A Survey of Web Caching Schemes for the Internet»*, J. Wang. ACM Computer Communication Review, vol. 25, no. 9, pp. 36–46, 1999.

[130] *«GreedyDual-Size : A Cost-Aware WWW Proxy Caching Algorithm»*, P. Cao, S. Irani. Proceedings of the 2nd Web Caching Workshop, 1997.

[131] *«Cache replacement policies for P2P file sharing protocols»*, A. Wierzbicki, N. Leibowitz, M. Ripeanu, R. Wozniak. European Transactions on Telecommunications, Vol. 15, No. 6, Nov. 2004, pp. 559–569.

[132] *«Economics of peering»*, S. Gibbard. Switch and Data Peering Forum, October 2004.

[133] *«Traffic Modeling and Proportional Partial Caching for Peer-to-Peer Systems»*, M. Hefeeda, O. Saleh. IEEE/ACM Transactions on networking, vol. 16, no. 6, December 2008.

[134] *La team Cymru*
http ://www.team-cymru.org/Services/ip-to-asn.html (accédé en avril 2010).

[135] *«Adaptive Peer Selection»*, D. S. Bernstein, Z Feng, B. N. Levine, S. Zilberstein. Proceedings of the 2nd International Workshop on Peer-To-Peer systems (IPTPS 2003).

[136] *«An Empirical Evaluation of WideArea Internet Bottlenecks»*, A. Akella, S. Seshan, A. Shaikh. Proceedings of ACM SIGCOMM, October 2003.

[137] *«Should ISPs fear Peer-Assisted Content Distribution ?»*, T. Karagiannis, P. Rodriguez, K. Papagiannaki. In ACM USENIX IMC, Berkeley 2005.

[138] *«Can ISPs and P2P systems co-operate for improved performance ?»*, V. Aggarwal, A. Feldmann, C. Scheideler. In ACM SIGCOMM Computer Communications Review (CCR), 37 :3, pp. 29–40.

[139] *«Taming the Torrent : A practical approach to reducing cross-ISP traffic in P2P systems»*, Choffnes, D. and F. Bustamante. Proceedings of ACM SIGCOMM, August 2008.

[140] *«Application-Layer Traffic Optimization (ALTO) Requirements»*, S. Kiesel, L. Popkin, S. Previdi, R. Woundy, Y. Yang. draft-ietf-alto-reqs-00 (work in progress), November 2009.

[141] *«P4P : Explicit Communications for Cooperative Control Between P2P and Network Providers»*, DCIA P2P Market Conference, March

2008, New-York. http ://www.dcia.info/documents/P4P_Overview.pdf (accédé en mars 2010).

[142] *«P4P : Provider Portal for Applications»*, H. Xie, Y. R. Yang, A. Krishnamurty, Y. Liu, A. Siverschatz. Proceedings of SIGCOMM'08, August 17-22, 2008.

[143] *«Analyzing peer-to-peer traffic across large networks»*, S. Sen, J. Wang. IEEE/ACM Transactions on Networking, pp. 219–232, 2002.

[144] *«The Effect of Peer Selection with Hopcount or Delay Constraint on Peer-to-Peer Networking»*, S. Tang, H. Wang, P. Van Mieghem. Proceedings of NETWORKING 2008, pp. 358-365, 2008.

[145] *«Topologically aware overlay construction and server selection»*, S. Ratnasamy, M. Handley, R. Karp, S. Shenker. INFOCOM 2002.

[146] *«Locality Prediction for Oblivious Clients»*, K. Shanahan, M. Freedman. IPTPS 2005.

[147] *«Optimal Selection of Peers for P2P Downloading and Streaming»*, M. Adler, R. Kumar, K. Ross, D. Rubenstein, T. Suel, D. Yao. INFOCOM 2005.

[148] *«Improving Traffic Locality in BitTorrent via Biased Neighbor Selection»*, R. Bindal, P. Cao, W. Chan, J. Medval, G. Suwala, T. Bates, A. Zhang. IEEE ICDCS, 2006.

[149] *«Caching Peer-To-Peer traffic»*, Y. Carlinet, L. Mé, H. Debar, Y. Gourhant. ICN 2010 (Internet Conference on Networking).

[150] *«Evaluation of P4P based on real traffic measurement»*, Y. Carlinet, L. Mé, H. Debar, Y. Gourhant. ICIMP 2010 (Internet Conference on Internet Measurement and Protection).

[151] *Xavier Niel critique YouTube* - http ://www.pcinpact.com/actu/news/56452-xavier-niel-freebox-v6-youtube.htm (accédé en mai 2010).

[152] *Cisco Visual Networking Index : Forecast and Methodology, 2008-2013*, «White paper» de Cisco, juin 2009.

Date et signature du directeur de thèse :

Ludovic Mé

Date et signature du président du jury de thèse :

César Viho