Efficacité et vie privée : de BitTorrent à Skype

Arnaud Legout

UNIVERSITE DE NICE-SOPHIA ANTIPOLIS - UFR SCIENCES

Ecole Doctorale STIC

THESE

Présentée pour obtenir le titre de

Habilitation à Diriger des Recherches

de l'Université de Nice Sophia Antipolis

Spécialité : Informatique

par

Arnaud LEGOUT

Efficacité et vie privée : de BitTorrent à Skype

Soutenue publiquement le 20 janvier 2012 devant le jury composé de :

M.	Philippe	Nain	Inria	Président et tuteur
Mme	Anne-Marie	Kermarrec	Inria	Rapporteur
M.	Laurent	Massoulié	Technicolor	Rapporteur
M.	Laurent	Viennot	Inria	Rapporteur
M.	Refik	Molva	Eurecom	Examinateur

Remerciements

Je tiens à remercier avant tout Walid Dabbous sans qui je ne serais probablement pas retourné à la recherche après 4 années passées dans une start-up. Walid a toujours supporté mes travaux, en particulier dans les moments difficiles ; peu l'auraient fait à sa place et je lui en suis très reconnaissant. Il a su créer des conditions qui ont grandement facilité mon travail. Sa compréhension des problèmes et son esprit de synthèse m'ont toujours été d'une grande aide. Je tiens également remercier tous les autres membres de Planète Chadi Barakat, Thierry Turletti et Thierry Parmentelat. En particulier, Chadi par les nombreuses discussions que l'on a eues, son esprit critique et son aide m'a remis le pied à l'étrier de la recherche internationale. Je remercie Stevens Le Blond, mon premier étudiant en thèse, sans qui le troisième chapitre de cette thèse n'aurait pas été possible. Pour finir, je remercie mes parents et Christian qui ont fait une relecture attentive de cette thèse.

À Louis et Gabriel.

Table des matières

Chapitre 1

Introduction

Cette thèse présente les résultats de deux axes de recherche que j'ai menés depuis 2004. Le premier axe est sur l'étude de l'efficacité de BitTorrent et le deuxième sur les risques d'atteinte à la vie privée sur Internet. On présente dans la suite la genèse de ces deux axes.

1.1 Genèse des travaux sur BitTorrent

Lorsque j'ai rejoint Inria (que l'on appelait encore l'INRIA) fin 2004, mon programme de recherche était sur la disponibilité et la sécurité des protocoles pair-à-pair dans un environnement mobile ; j'ai très vite abandonné cette composante mobile (éloignée de la réalité), me rendant compte qu'il y avait encore énormément à découvrir dans un environnement fixe (et bien plus réel). En explorant la bibliographie, je me suis rendu compte qu'il n'existait aucune preuve expérimentale de l'efficacité des protocoles pair-à-pair. Il y avait l'excellent travail de Yang *et al.* [89] qui venait d'être publié et qui montrait analytiquement l'efficacité optimale des protocoles proches de BitTorrent. Cependant, les hypothèses de leur modèle sont fortes : répartition idéale des pièces, choix optimal et instantané des pairs avec qui échanger. Il me paraissait évident à l'époque qu'avec des dizaines de milliers de pairs repartis sur Internet ces hypothèses ne pouvaient pas être valides.

Ainsi, j'ai décidé de me lancer dans l'étude du protocole pair-à-pair qui semblait le plus efficace : BitTorrent. En collaboration avec Guillaume Urvoy Keller et Pietro Michiardi, nous avions décidé d'explorer plusieurs approches méthodologiques, dont l'expérimentation que je pris en charge. Il s'agissait de l'approche la plus risquée parce que personne n'avait essayé (ou réussi) avant nous d'instrumenter un client BitTorrent et de faire des expérimentations sur un grand nombre de vrais torrents. Je mis environ deux semaines à comprendre et entièrement instrumenter [6] le client python BitTorrent implémenté par Bram Cohen l'inventeur du protocole et j'ai commencé mes premières expérimentations en juin 2005. J'avais formulé deux hypothèses à cette période. Premièrement, les caractéristiques des torrents dépendent de la nature du contenu ; typiquement, lorsqu'un pair échange un contenu sous copyright sans autorisation, on peut s'attendre à ce que son comportement soit très différent que lorsqu'il échange un contenu libre de

droit. Cette différence de comportement doit grandement influer sur la performance globale d'un torrent. Deuxièmement, l'efficacité réelle de BitTorrent est éloignée de ce que prévoit les modèles de Yang *et al.* [89]. En effet, comment serait-il possible qu'en pratique il y ait une répartition optimale des pièces et un choix idéal des pairs.

Après quelques semaines d'expérimentations, je dus me rendre à l'évidence : je m'étais trompé sur ces deux hypothèses. L'espoir d'inventer un nouveau protocole qui supplanterait BitTorrent s'envolait. Je décidais alors de comprendre les raisons de cette incroyable efficacité. J'observais en effet que la répartition des pièces et le choix des pairs étaient proches de la prédiction théorique *presque* tout le temps, mais pas *tout* le temps ? Je découvris que le dimensionnement de la source initiale déterminait l'efficacité du torrent et que les problèmes de performance étaient toujours dus à une source trop lente, mais pas à une inefficacité de BitTorrent. L'année suivante, je montrai, en collaboration avec Nikitas Liogkas, Eddie Kohler et Lixia Zhang, que le dimensionnement de la source initiale déterminait également la dynamique des algorithmes de BitTorrent et en particulier l'incitation au partage.

Ainsi BitTorrent est un protocole qui offre une excellente efficacité pour les utilisateurs (lorsque la source est bien dimensionnée), mais qu'en est-il pour le réseau ? Les travaux de Karagiannis *et al.* [59] suggèrent une mauvaise efficacité du pair-à-pair pour le réseau due aux nombreux paquets redondants passant sur les liens. Ils proposent alors de garder le trafic pair-à-pair local pour réduire la redondance. Cependant, en gardant le trafic local, on crée une contrainte sur le graphe connectant les pairs qui peut se traduire par une importante baisse d'efficacité pour les utilisateurs. C'est pourquoi, lorsque j'ai commencé à travailler sur le sujet en définissant l'orientation de la thèse de Stevens Le Blond, les travaux existants considéraient tous une localité relativement faible de l'ordre de 80% [88, 34].

Je pensais qu'il existait un compromis facilement identifiable entre efficacité pour le réseau (avec une forte localité) et efficacité pour les utilisateurs (avec une faible localité). Les résultats de nos expérimentations montrèrent que l'efficacité de BitTorrent perçue par les utilisateurs est essentiellement indépendante de la localité. De nouveau, j'avais sous-estimé la robustesse remarquable de BitTorrent face à des conditions réseaux dégradées.

Les résultats présentés dans le chapitre 2 couvrent mes travaux de fin 2004 jusqu'à mes travaux sur la localité de BitTorrent.

1.2 Genèse des travaux sur les risques d'atteinte à la vie privée

Dans le cadre des travaux sur la localité de BitTorrent, on a effectué des mesures de répartition des pairs par AS. Pour ces mesures, on devait collecter les adresses IP des pairs dans des torrents, la qualité des résultats dépendant du nombre de torrents considérés. En développant les outils de mesure, nous nous sommes aperçus qu'il était possible d'obtenir l'adresse IP d'un grand nombre de pairs. Nous avons alors cherché à savoir s'il était possible d'obtenir l'adresse

IP des pairs sur tous les torrents publics existants.

À notre grande surprise, nous réussîmes à collecter l'adresse IP de tous les pairs (autour de 10 millions) sur tous les torrents publics (autour de 700 000) avec une fréquence de seulement 30 minutes, c'est-à-dire quasiment en temps réel. Ce résultat prouvait qu'il existait un problème majeur de protection de la vie privée sur BitTorrent. Bien sûr BitTorrent n'a pas été conçu pour protéger la vie privée de ses utilisateurs, bien sûr les communications pair-à-pair exposent l'adresse IP de chaque pair à n'importe qui, mais jusqu'ici on pensait qu'il était impossible de suivre l'activité de millions de pairs en temps réel sans une lourde infrastructure. Avec ce premier travail, on montrait que sans aucune infrastructure dédiée (d'un simple ordinateur de bureau) on pouvait en temps réel surveiller l'intégralité des pairs utilisant BitTorrent soit 148 millions d'adresses IP uniques sur 3 mois.

La nouvelle difficulté à publier des travaux classiques sur le pair-à-pair dans des conférences de haut niveau et l'importance du problème que l'on avait identifié me convainquirent de réorienter la thèse de Stevens et de démarrer un nouvel axe sur l'exploration des risques d'atteinte à la vie privée sur Internet : le projet *Bluebear* [8].

Il est important, à ce point de la description de la genèse du projet *Bluebear*, d'expliquer notre motivation et notre éthique dans ce projet. Pour cela il faut se poser deux questions : qu'est-ce qui nous différencie d'un pirate informatique et est-ce qu'en publiant des attaques on ne facilite ou n'ouvre pas la voie à une nouvelle forme de piratage informatique ? La principale différence (sur les aspects éthiques) entre un chercheur et un pirate informatique est qu'à aucun moment on ne va exploiter, contre l'intérêt des internautes, les données collectées, et on va valider que nos expérimentations ne les impactent d'aucune façon (par exemple, en créant un ralentissement ou une perte de service). De plus, on va prendre toutes les mesures nécessaires pour rendre impossible l'exploitation de ces données par un tiers : effacement de données facilitant l'identification, analyse à la volée pour ne conserver que des données agrégées, stockage sur des machines sécurisées et chiffrement des données, etc. C'est pour cette raison que j'ai décidé de ne pas rendre les traces d'expérimentations publiques, même rendues anonymes. En effet, on ne peut jamais garantir qu'une trace anonyme ne peut pas permettre de remonter à des individus après une analyse sophistiquée [92, 82].

Concernant la légitimité de publier des attaques, il existe deux voix dissonantes. La première affirme qu'une attaque ne doit jamais être publiée tant qu'elle n'est pas corrigée : il s'agit de la sécurité par l'obscurantisme. En informatique, cette stratégie n'a jamais marché et conduit toujours à une situation dans laquelle des pirates informatique exploitent des failles sans que les internautes ne s'en rendent compte ou puissent s'en prémunir puisqu'ils n'ont aucun moyen de savoir qu'un tel risque existe. C'est la voix des sociétés qui ne veulent pas investir dans la sécurité de leurs clients et qui préfèrent avoir des failles exploitées sans que personne ne le sache plutôt que de dépenser de l'argent à corriger des failles qui ne sont peut-être pas encore exploitées. La deuxième voix recommande une publication rapide des attaques après un petit délai de carence

pour offrir l'opportunité de corriger la ou les failles exploitées. Il s'agit du *responsible disclosure* que j'ai suivi dans mes travaux. Ma motivation est donc de documenter d'importants problèmes pouvant compromettre la vie privée de nombreux internautes de manière à ce que des solutions soient apportées avant l'exploitation par des pirates informatique.

Revenons maintenant à l'attaque sur BitTorrent. Qu'est-ce qui pourrait être pire que de construire une base de données exhaustive associant chaque adresse IP aux contenus qu'elle a téléchargés : associer une identité sociale (adresse email, nom, etc.) aux adresses IP. Ainsi, il deviendrait possible de faire une exploitation directe des données collectées contre l'intérêt des internautes concernés. Par exemple, on peut facilement imaginer des groupes mafieux, se faisant passer pour des ayants droits, envoyant par email à des internautes la liste exacte des contenus téléchargés et demandant une compensation financière pour ne pas engager une procédure judiciaire. Ce type d'emails serait très crédible et intimidant, et lorsque l'on parle de plus de 100 millions d'internautes potentiellement concernés, même 1% de réponses positives peut représenter des sommes très importantes, et donc une vraie incitation pour ces groupes.

Nous eûmes alors l'idée d'étudier si les systèmes de voix sur IP permettaient de faire le lien entre une identité sociale et une adresse IP. En effet, ces systèmes combinent un annuaire (qui contient l'identité sociale) et des communications pair-à-pair (qui permettent d'obtenir une adresse IP). Nous décidâmes de faire des expérimentations sur Skype, le système de voix sur IP le plus populaire. Cependant, il était loin d'être évident que nous réussirions à faire le lien entre une adresse IP et une identité sociale. En effet, Skype utilise un protocole propriétaire non documenté, toutes les communications sont chiffrées, chaque client Skype communique avec de nombreux autres clients, etc. En collaboration avec Chao Zhang et Keith Ross, nous découvrîmes qu'il était non seulement possible de lier une identité à une adresse IP, mais qu'on pouvait le faire sans que l'internaute ne puisse le détecter ni l'empêcher.

Une réaction naturelle aux attaques que l'on a décrites jusqu'à maintenant est de penser qu'en utilisant un réseau garantissant l'anonymat on peut y échapper. C'est pourquoi nous nous sommes penchés sur Tor, le réseau garantissant l'anonymat le plus populaire. Mon intuition initiale était qu'avec les multiples mécanismes de découverte des pairs implémentés dans BitTorrent, il devait être possible de retrouver l'adresse IP publique d'un pair BitTorrent qui utilise Tor. On réussit effectivement à retrouver l'adresse de pairs utilisant BitTorrent sur Tor, montrant ainsi que l'utilisation d'un réseau garantissant l'anonymat n'était pas une solution suffisante pour protéger des attaques sur la vie privée.

Les résultats présentés dans le chapitre 3 couvrent nos contributions liées à la vie privée.

Chapitre 2

BitTorrent : ruisseau ou torrent ?

2.1 Introduction

Les premiers modèles sur les systèmes pair-à-pair [89, 79, 25] (que l'on discutera avec plus de détails en section 2.2.1) ont montré la très grande efficacité de tels systèmes. Cependant, ces modèles font des hypothèses fortes. Ils considèrent des politiques idéales de sélection de pièces et de pairs, c'est-à-dire que tout pair à toujours une pièce à donner à un autre pair, qu'il sait trouver cet autre pair et qu'il accepte de lui donner cette pièce. La conséquence de ces hypothèses est que l'utilisation de la capacité d'envoi de chaque pair est maximale.

La grande innovation de BitTorrent a été de proposer des politiques de sélection de pièces et de pairs reposant sur des algorithmes totalement distribués dont le but est de maximiser l'utilisation de la capacité d'envoi des pairs. La politique de sélection de pièces, *rarest first*, a pour but de garantir une bonne diversité des pièces, c'est-à-dire que la probabilité que tout pair à toujours une pièce à donner à n'importe quel autre pair est forte. L'objectif est de minimiser les contraintes lors du choix des pairs avec qui échanger. Ce choix des pairs est effectué par la politique de sélection de pairs, *choke*, dont le but est d'inciter les pairs à contribuer et de maximiser l'utilisation des ressources.

Cependant, les politiques de BitTorrent furent remises en cause par plusieurs travaux. L'efficacité de la politique *rarest first* a été remise en cause par plusieurs auteurs. Gkantsidis *et al.* [47] ont montré, en utilisant des simulations, que la politique *rarest first* pouvait conduire à une mauvaise diversité des pièces sous certaines conditions (peu réalistes dans le contexte d'Internet). Ils ont alors proposé une solution élégante utilisant le *network coding*, mais conduisant à de nouveaux problèmes de performance et de sécurité. Kostić *et al.* [61] ont proposé une solution de codage à la source pour résoudre le même problème. Cependant, le codage à la source ne fait qu'augmenter le nombre de pièces en créant des pièces de parité à la source, mais ne supprime pas le besoin d'avoir une politique de sélection de pièces.

Plusieurs travaux [50, 57, 46, 23] ont discuté le manque d'équité et de robustesse de la politique *choke*. En particulier, des auteurs ont critiqué le fait que la politique *choke* cherche à garantir la réciprocité des débits et non la réciprocité stricte des bits échangés. En effet, il a

été considéré que la réciprocité des débits pouvait favoriser les *free riders*, c'est-à-dire les pairs qui ne contribuent pas ou peu. Des solutions utilisant la réciprocité stricte des bits échangés ont été proposées. Nous verrons en section 2.3 que la réciprocité stricte des bits échangés est sous-optimale, et que l'équité atteinte lorsqu'il y a réciprocité des débits est pertinente.

Dans ce chapitre, on va réévaluer les problèmes identifiés par les travaux précédents en utilisant le client BitTorrent original et des expérimentations avec soit un seul client BitTorrent que l'on contrôle et qui est connecté à de vrais torrents (pour avoir une vue non-intrusive de ce qui se passe en réalité), soit des torrents dans lesquels on contrôle tous les clients BitTorrent (pour avoir une vue globale de la dynamique de BitTorrent).

Nos contributions sont les suivantes [63, 64, 62, 27].

- Dans la section 2.3.1, on montre que la politique *rarest first* est très efficace et garantit une diversité des pièces proche de l'optimale. On explique pourquoi dans certains cas une mauvaise diversité des pièces peut être observée. En particulier, on montre que ce manque de diversité n'est pas dû à la politique *rarest first*, mais à un mauvais dimensionnement de la source initiale.
- Dans la section 2.3.2, on montre que la politique *choke* est équitable, conduit à la formation de clusters de pairs avec des capacités d'envoi équivalentes, offre une incitation au partage et est, par conséquent, robuste aux *free riders*, et garantit une bonne utilisation de la capacité d'envoi des pairs. On montre qu'un sous-dimensionnement de la source initiale peut complètement changer les propriétés de la politique *choke* avec, en particulier, l'absence d'incitation au partage.
- Dans la section 2.3.3, on montre que le trafic BitTorrent peut être gardé local (pour réduire le trafic de transit) sans baisse significative des performances. On montre également qu'une politique de localité pourrait réduire le trafic interFAI de 40% par rapport au trafic BitTorrent actuel.

Nos résultats diffèrent significativement des travaux antérieurs [47, 61, 50, 57, 46, 23]. On peut attribuer ces différences à trois raisons. Premièrement, on se place dans le contexte d'Internet où il n'y a pas de restriction de connexions entre pairs comme cela peut être le cas dans les réseaux *ad hoc* ou de capteurs. Par exemple, Gkantsidis *et al.* considèrent un scénario, avec une très mauvaise connexion entre deux groupes de pairs, qui n'est pas représentatif d'Internet aujourd'hui. Deuxièmement, on utilise une version expérimentale de la politique *choke* lorsque le pair est une source. Tous les autres travaux considèrent une version plus ancienne de cet algorithme qui est moins équitable et moins robuste aux *free riders* que la version expérimentale que l'on évalue. On discutera en section 2.3 les raisons pour lesquelles l'ancienne version est toujours utilisée aujourd'hui. Troisièmement, tous nos résultats se basent sur des expérimentations et non sur des simulations comme c'est le cas pour les travaux antérieurs [23, 47]. Simuler un système pair-à-pair nécessite de nombreuses simplifications qui peuvent avoir un impact important sur les résultats obtenus. Toutes les simulations dans les travaux antérieurs [23, 47, 44] considèrent

un petit voisinage ce qui a un impact majeur sur la performance de la politique *rarest first*. En effet, lorsqu'on réduit la taille du voisinage, le diamètre du graphe de connexion entre les pairs va augmenter, ce qui pénalise fortement l'efficacité de la politique *rarest first*.

En résumé, on montre dans ce chapitre que les politiques *rarest first* et *choke* ont un comportement proche de l'idéal lorsque les pairs sont connectés à Internet, c'est-à-dire sans restriction de connexions et avec un grand voisinage (de l'ordre de 80 dans nos expérimentations).

Dans la suite, on présente l'état de l'art dans la section 2.2, nos contributions dans la section 2.3 et on discute nos résultats dans la section 2.4.

2.2 État de l'art

BitTorrent a été pour la première fois décrit officiellement par Bram Cohen son inventeur en 2003 [35]. Cependant, cette publication se limite à une description des algorithmes et du protocole, mais ne fournit aucune évaluation. De nombreuses questions existant sur l'efficacité réelle de BitTorrent, plusieurs publications suivirent pour aborder ces questions. Nous classons ces publications en cinq catégories : les modèles, les simulations, les mesures, les études sur l'incitation au partage et les études sur la localité.

2.2.1 Modèles

Le premier modèle à avoir eu un impact majeur sur la compréhension du fonctionnement de BitTorrent (et du pair-à-pair en général) est dû à Yang *et al.* [89]. Dans cette étude, ils présentent un modèle déterministe et un modèle stochastique qui montrent que la capacité de service augmente de manière exponentielle avec le temps et que le temps de téléchargement est une fonction logarithmique du nombre de pairs. Ils étudient également les logs d'un tracker pour valider leurs modèles.

Qiu et Srikant [79] étendent le modèle de Yang *et al.* en proposant une solution analytique à un modèle fluide. Ils utilisent également la théorie des jeux pour analyser le comportement de l'algorithme de sélection de pairs de BitTorrent (l'algorithme *choke*). Ils montrent la stabilité et l'efficacité de cet algorithme.

Biersack *et al.* [25] proposent une analyse de trois modèles de distributions synthétiques : une chaîne, un arbre et une forêt d'arbres. Ils étudient en particulier l'impact du nombre de pièces et du nombre d'envois parallèles (c'est-à-dire, le nombre de pairs vers lesquels un pair donné envoie simultanément des pièces). Ils montrent qu'un grand nombre de pièces et un petit nombre d'envois parallèles améliorent la performance.

Massoulie *et al.* [68] proposent un modèle probabiliste d'un système pair-à-pair et montrent que l'efficacité du système ne dépend pas fondamentalement de la contribution en tant que source des pairs, ni de la politique de sélection de pièces. Ce dernier point suggère que la politique de sélection de pièces *rarest first* n'apporte pas de bénéfices significatifs par rapport à la politique

aléatoire, ce qui est contredit par d'autres travaux [24, 44]. Les principales raisons de cette divergence sont deux hypothèses irréalistes considérées par Massoulie *et al.* dans leur modèle. Premièrement, ils supposent que chaque pair à déjà une pièce lorsqu'il joint le système et que la sélection de pairs se fait aléatoirement (soit dans tout le système ou dans une classe de pairs). Ces hypothèses améliorent artificiellement la diversité des pièces. Deuxièmement, ils ne considèrent par la notion de voisinage (également appelé *peer set*). En effet, le problème avec la politique aléatoire de sélection de pièces est que les pièces sont choisies au hasard parmi les pièces disponibles dans le voisinage. Par conséquent, il y a une plus forte probabilité de sélectionner les pièces les plus populaires. En permettant une sélection de pairs (et donc de pièces) en dehors du voisinage, la diversité des pièces est ici encore artificiellement améliorée.

Fan *et al.* [43] présentent un modèle de BitTorrent évaluant le compromis entre équité et efficacité. Plus particulièrement, ils regardent l'impact sur l'équité et l'efficacité de différentes politiques de sélection de pairs et de répartition de la capacité de ces pairs. Leur conclusion est qu'il existe de nombreux points opérationnels et que BitTorrent ne représente qu'un de ces points.

Tous ces travaux permettent une meilleure compréhension des systèmes pair-à-pair et de BitTorrent en particulier. Cependant, ils font tous des hypothèses fortes sur les politiques de sélection de pairs et de pièces. Ils considèrent qu'un pair à toujours une pièce à demander à un autre pair (c'est-à-dire une politique de sélection de pièces idéale) et qu'un pair peut toujours utiliser sa capacité d'envoi au maximum (c'est-à-dire une politique de sélection de pairs idéale). Sous ces hypothèses, ils montrent une grande efficacité des systèmes pair-à-pair.

Montrer une grande efficacité de BitTorrent en supposant des politiques idéales de sélection de pièces et de pairs est une contribution importante. En effet, avant 2004 (date des premiers travaux évaluant l'efficacité de BitTorrent), il n'y avait aucune preuve qu'un système pair-à-pair soit réellement efficace. Cependant, passer de politiques idéales de sélection de pairs et de pièces à une implémentation réelle et distribuée ayant les mêmes propriétés est un vrai challenge.

Un des objectifs de ce chapitre est de montrer expérimentalement que les politiques de sélection de pairs et de pièces de BitTorrent sont proches de politiques idéales et ainsi valider les hypothèses faites par les nombreux modèles existants.

2.2.2 Simulations

Felber *et al.* [44] proposent une des premières évaluations de l'impact de différentes politiques de sélection de pairs et de pièces dans un système pair-à-pair. Ils montrent que le choix de ces politiques a un impact majeur sur l'efficacité du système.

Bharambe *et al.* [23] proposent une étude par simulation spécifique à BitTorrent. Ils étudient l'utilisation de la capacité d'envoi des pairs ainsi que le volume de données échangées par chaque pair ce qui leur permet de conclure sur l'efficacité et l'équité de BitTorrent. Ils étudient également le rôle de la source initiale dans l'efficacité du démarrage d'un torrent.

Peu de travaux de simulations ont été publiés sur BitTorrent. La principale raison est que les travaux expérimentaux (dont ce chapitre fait parti) ont supplanté les simulations ; les expérimentations, bien que plus complexes à mettre en œuvre, permettent d'obtenir des résultats beaucoup plus réalistes.

En effet, si le simulateur est à base d'un modèle fluide, il permet de faire des simulations à grande échelle, mais il ne permet pas de capturer la dynamique très importante qui existe au niveau des paquets. Si le simulateur est à évènements discrets, le nombre d'évènements à gérer augmente de manière polynomiale avec le nombre de pairs. Par conséquent, on doit non seulement limiter la taille du voisinage de chaque pair, ce qui a un impact sur l'efficacité de la politique *rarest first* et donc biaise les résultats, mais on doit également limiter le nombre de pairs simulés. Il n'y a pas à notre connaissance de simulations événementielles considérant un voisinage supérieur à 15 pairs alors qu'il est de 50 à 80 pairs en réalité.

Tous les résultats présentés dans ce chapitre sont obtenus à partir d'expérimentations utilisant un vrai client BitTorrent et de vrais torrents.

2.2.3 Mesures

Quelques travaux de mesures ont eu un impact majeur sur la compréhension de BitTorrent. Ces travaux ont quasi exclusivement porté sur des logs de trackers.

Pouwelse *et al.* [77] étudient la popularité et la disponibilité des torrents, la performance de téléchargement, la durée de vie de torrents et le degré de pollution sur Suprnova le site BitTorrent le plus populaire à l'époque de l'étude. Tous ces résultats ont été obtenus à partir des trackers et portails Web de Suprnova.

Izal *et al.* [55] étudient le log d'un tracker pour un torrent de la distribution d'une image ISO de Linux Redhat 9. Ils fournissent des résultats sur la proportion de sources et de *leechers*, la distribution géographique des pairs, la corrélation entre vitesses agrégées d'envoi et de réception. Cette corrélation semble suggérer que la politique *choke* fonctionne correctement. Ils présentent également une étude succincte du comportement d'un pair en se basant sur une mesure locale d'un pair.

Andrade *et al.* [20] regardent les communautés de partage BitTorrent. Ils trouvent qu'une contrainte sur le rapport envoi/réception et que l'utilisation de flux RSS pour publier de nouveaux torrents pouvaient améliorer la contribution des pairs. Ils concluent que les torrents avec un large nombre de sources offrent des opportunités aux *free riders*.

La présence de *free riders* lorsqu'il y a un grand nombre de sources a souvent été interprétée à tort comme un problème ; il faut considérer ces sources comme de la capacité disponible dans les torrents. En effet, BitTorrent a été conçu pour garantir une incitation au partage dans les phases critiques de vie d'un torrent, c'est-à-dire lorsqu'il y a beaucoup plus de *leechers* que de sources. Comme on le verra en section 2.3, BitTorrent réussi parfaitement cela. Lorsque le nombre de sources s'approche ou devient plus grand que le nombre de *leechers*, la capacité disponible dans

le système devient très importante et le torrent se rapproche d'un modèle client serveur puisque chaque *leecher* a une source pour le servir. Il n'est plus dans ce cas critique pour l'efficacité du système d'inciter les *leechers* à contribuer.

Guo *et al.* [51] montrent que les taux d'arrivée et de départ des pairs suivent une loi exponentielle, et que la performance fluctue beaucoup dans les petits torrents. La collaboration intertorrent est présentée comme une solution pour inciter les *leechers* à rester en tant que sources dans les torrents.

Tous ces travaux se basent sur des études de logs de trackers. Ils présentent donc une vision agrégée du comportement des pairs, mais ils ne peuvent pas, en se limitant aux logs de trackers, aborder la dynamique des politiques *rarest first* et *choke*. Ils sont, par conséquent, orthogonaux aux résultats présentés dans ce chapitre.

2.2.4 Incitation au partage

Plusieurs travaux ont abordé la robustesse de BitTorrent aux *free riders*, c'est-à-dire aux pairs qui contribuent peu ou pas.

Dans la littérature, on dit que BitTorrent n'est pas robuste aux *free riders* lorsqu'un pair peut augmenter sa vitesse de téléchargement sans augmenter sa contribution.

Shneidman *et al.* [84] ont été les premiers à montrer que BitTorrent n'était pas robuste aux *free riders*. Ils décrivent brièvement trois attaques basiques comme de se déclarer de multiples fois au tracker pour avoir un plus grand nombre de voisins (on note qu'il suffit de faire de multiples requêtes au tracker avec la même identité pour avoir plus de voisins) ou de prétendre avoir des pièces et d'envoyer à la place de fausses pièces dans le but d'augmenter le nombre d'*unchokes* que les voisins vont faire. Ces attaques peuvent permettre une amélioration transitoire minime, mais elles auront un effet catastrophique sur le téléchargement du pair qui fait cette attaque après quelques minutes. En effet, les clients BitTorrent bannissent systématiquement les pairs qui ont un comportement suspect ou néfaste (comme d'envoyer des pièces corrompues).

Jun *et al.* [58] affirment sans le montrer que la politique *choke* n'est pas suffisante pour pénaliser les *free riders*. Ils proposent alors une politique de réciprocité stricte et montrent sur des expérimentations que leur politique est meilleure que la politique *choke*. Cependant tous leurs résultats se basent sur une erreur expérimentale. En effet, comme on le montrera dans la section 2.3, le dimensionnement de la source initiale est critique pour l'incitation au partage de BitTorrent. Hors, dans les simulations présentées, la source initiale partage sa capacité de service avec tous les *leechers* (c'est un choix des auteurs, mais ça ne correspond pas au fonctionnement normal de BitTorrent) ce qui conduit à un comportement pathologique. Malgré une vitesse suffisante d'envoi de la source initiale, le débit par pair est très faible.

Liogkas *et al.* [66], Locher *et al.* [67] et Sirivianos *et al.* [86] ont conçu et implémenté des attaques permettant à un *free rider* d'améliorer sa vitesse de téléchargement dans certaines circonstances. Cependant, ces attaques ont un impact pratique limité. En effet, elles se basent

toutes sur les deux mêmes principes : augmenter la tailler du voisinage, ou se connecter uniquement à des sources.

L'augmentation de la taille du voisinage n'a qu'un effet limité. Supposons que la vitesse d'envoi moyenne soit de 20 kB/s, que chaque pair donne en parallèle à 4 autres pairs, que la taille du voisinage soit de 50 pairs, et que l'*optimistic unchoke* ait lieu toutes les 30 secondes. Sous ces hypothèses, un pair qui contribue va recevoir en moyenne à 20 kB/s. Comme l'*optimistic unchoke* ne représente qu'un quart de la capacité d'envoi d'un pair, un *free rider* doit avoir un voisinage 4 fois plus grand pour recevoir à la même vitesse qu'un pair moyen, c'est-à-dire 200 voisins. Cependant, Zhang *et al.* [91] ont montré que moins de 1% des torrents avaient plus de 100 pairs. Par conséquent, augmenter suffisamment la taille du voisinage est infaisable dans plus de 99% des cas. On souligne également que plus un *free rider* a une capacité d'envoi importante et plus il aurait pu recevoir rapidement en contribuant. En effet, on montre dans la section 2.3 qu'un pair reçoit à une vitesse comparable à sa vitesse de contribution dans la phase de démarrage du torrent.

Se connecter uniquement à des sources pour recevoir sans contribuer ne représente pas une réelle attaque comme on l'a discuté en section 2.2.3. En effet, plus il y a de sources et plus la capacité de service librement disponible dans le torrent est grande. De plus, on discutera en section 2.3 les deux politiques de sélection de pairs pour les sources et on montrera qu'il existe une politique qui est robuste aux *free riders*.

Piatek *et al.* [72] montrent que les pairs très rapides sont pénalisés parce qu'ils contribuent beaucoup plus qu'ils ne reçoivent. Ils introduisent ainsi BitTyrant qui est une modification de BitTorrent qui cherche à minimiser la vitesse d'envoi d'un pair tout en maximisant sa vitesse de réception. Les auteurs montrent que BitTyrant est plus efficace que l'implémentation Azureus de BitTorrent. Cependant, cette optimisation de la vitesse d'envoi est contraire à la philosophie de BitTorrent qui cherche à inciter au partage et consacre une partie de la capacité de chaque pair, avec l'*optimistic unchoke*, pour les nouveaux pairs sans pièce. Le résultat de cette optimisation est une baisse globale d'efficacité et une difficulté pour les nouveaux pairs de joindre des torrents. De plus, on verra en section 2.3.2 qu'en pratique cette optimisation peut avoir un effet désastreux en cas de sous-dimensionnement de la source initiale ; dans ce cas, la vitesse de téléchargement n'est plus une fonction de la vitesse d'envoi, donc BitTyrant convergera vers une capacité d'envoi minimale pour tous les pairs.

Les travaux que l'on présente dans ce chapitre expliquent pourquoi sous certaines conditions BitTorrent semble ne pas fonctionner efficacement. Ils montrent en particulier l'importance du dimensionnement de la source initiale dans la robustesse de BitTorrent aux *free riders*.

2.2.5 Localité

Alors que BitTorrent permet à un fournisseur de contenus d'économiser le coût d'une infrastructure de distribution, les opérateurs réseaux sont fortement pénalisés à cause d'une aug-

mentation de trafic qui est difficile à planifier. En effet, un torrent est une session de transfert spontanée avec potentiellement un grand nombre de pairs interconnectés aléatoirement. La spontanéité rend difficile la planification de trafic pour un fournisseur d'accès à Internet (FAI) (ou *internet service providers (ISP)*), et les interconnexions aléatoires favorisent les communications interFAI redondantes.

La premier travail qui a introduit le concept de localité et montré son bénéfice a été réalisé par Karagiannis *et al.* [59] en 2005. Ils montrent avec une capture de trafic sur un lien d'accès d'un réseau et avec des simulations se basant sur un log de tracker BitTorrent [55] qu'en localisant le trafic pair-à-pair il y a un réel potentiel de réduction de trafic sur les liens interFAI sans pénaliser les utilisateurs.

D'autres travaux ont suivi sur l'évaluation de la localisation du trafic pair-à-pair. Bindal *et al.* [26] étudient l'impact de la localisation sur la charge des liens de peering des FAI et sur la performance des utilisateurs. Les auteurs considèrent un scénario avec 14 FAI contenant 50 pairs chacun, soit un torrent de 700 pairs. Lin *et al.* [65] introduisent une politique de localisation appelée ELP. Ils fournissent un modèle donnant des bornes sur le trafic interFAI et valident expérimentalement leur approche avec un déploiement de 60 pairs sur PlanetLab.

Plusieurs travaux ont été consacrés à la conception d'une solution de localisation du trafic pouvant être largement déployée. P4P [88] est un projet visant à concevoir une infrastructure légère permettant la coopération entre les applications pair-à-pair et les FAI. Xie *et al.* présentent des expérimentations à petite échelle (entre 53 et 160 pairs sur PlanetLab) sur deux scénarios spécifiques. Ils présentent également une expérimentation sur un torrent réel pour lequel les pairs d'un seul FAI utilisent P4P. Ils montrent que cet FAI peut réduire son trafic interFAI de 60%.

Aggarwal *et al.* [19] présentent une architecture similaire par certains aspects à celle de P4P. Les auteurs définissent la notion d'oracle qui est offert par chaque FAI pour fournir à chaque nouveau pair une liste de voisins dans le même FAI. Ils évaluent leur solution en utilisant des simulations et des expérimentations à petite échelle sur 45 nœuds Gnutella.

Une autre approche ne nécessitant pas d'infrastructure dédiée propre est proposée par Ono [34]. Ono utilise en fait l'infrastructure d'un CDN en faisant l'hypothèse que les pairs redirigés vers le même serveur du CDN sont des pairs proches. Ono est implémenté sous la forme d'un plugin du client BitTorrent Vuze et est largement déployé. Les auteurs rapportent les mesures collectées sur 120 000 utilisateurs d'Ono sur une période de 10 mois. Ils montrent une amélioration de la vitesse moyenne de téléchargement de 207%. Cependant, ils ne donnent aucun résultat direct sur l'impact d'Ono sur le trafic interFAI, ils montrent uniquement une réduction du chemin moyen entre pairs utilisant Ono (ce qui ne peut pas être directement corrélé à une réduction de trafic interFAI).

Contrairement à des travaux tels que P4P [88] ou Ono [34] qui présentent des résultats sur des déploiements réels, les résultats que l'on présente en section 2.3.3 fournissent une étude

systématique et rigoureuse de l'impact de la localisation du trafic BitTorrent dans le cadre d'expérimentations contrôlées. Nos travaux diffèrent de l'état de l'art en répondant à deux questions fondamentales : jusqu'où peut-on localiser le trafic pair-à-pair ? quelle réduction de trafic interFAI peut être obtenue à l'échelle d'Internet ? Ils diffèrent également par l'échelle des expérimentations et des mesures, et par l'évaluation systématique de nombreux paramètres importants. En particulier, on considère 214 443 vrais torrents répartis sur 9 605 AS (un seul torrent et un seul AS étaient considérés dans l'expérimentation faite sur P4P [88]) et on montre qu'en utilisant seulement 4 connexions interFAI (l'expérimentation faite sur P4P considère 20% de connexions interFAI) on peut réduire le trafic interFAI à l'échelle d'Internet de 40% sans dégradation notable de la performance des utilisateurs.

Piatek *et al.* [75] présentent trois pièges dans la conception de politiques de localisation qui minimisent le trafic interFAI : la localisation se basant sur le support des utilisateurs peut ne pas fonctionner correctement ; la localisation peut dégrader l'efficacité et la robustesse ; les FAI ont des intérêts différents suivant leur rôle. Les deux premiers pièges ne s'appliquent pas à notre travail puisque l'on utilise une politique de localisation utilisant un tracker et que nous avons conçu et évalué une politique de réincorporation des partitions qui prévient les problèmes de robustesse. Le dernier piège concernant les intérêts divergents des FAI est orthogonal à notre travail. Notre travail se limite à montrer que si un FAI a un intérêt à localiser le trafic, alors il peut le faire de manière efficace et transparente pour les utilisateurs.

L'étude la plus proche de ce que l'on présente en section 2.3.3 a été faite par Cuevas *et al.* [37]. En effet, ces auteurs ont également collecté des grandes traces BitTorrent et ont exploré l'impact de la localisation du trafic. Cependant, il y a également d'importantes différences avec notre travail. Notre approche méthodologique est fondamentalement différente : le cœur de leur travail est un modèle mathématique, alors que notre évaluation repose sur des expérimentations à grande échelle. De plus, nous étudions l'impact de paramètres différents et nous répondons à l'importante question qui n'est pas abordée par le travail de Cuevas *et al.* : quelle réduction de trafic interFAI peut être obtenue à l'échelle d'Internet ? Par conséquent notre travail est complémentaire de celui de Cuevas *et al.* parce qu'il valide certaines hypothèses de leur modèle (en particulier la bonne diversité des pièces en cas de localisation du trafic qui est le prérequis nécessaire à la stratification des pairs) et répond à des questions importantes qui ne sont pas couvertes par ce travail.

2.3 BitTorrent est-il efficace ?

Les politiques de sélection de pièces et de pairs sont au cœur de l'efficacité de tout protocole pair-à-pair de réplication de contenus. Le but de la politique de sélection de pièces est de garantir une bonne entropie des pièces, c'est-à-dire une bonne diversité des pièces. Une entropie élevée signifie que la probabilité est élevée que n'importe quel pair ait au moins une pièce qui intéresse

n'importe quel autre pair. Dans la section 2.3.1, on montre que la politique de sélection de pièces de BitTorrent garantit une entropie élevée. On montre également que lorsque dans certains cas on observe une mauvaise entropie, elle n'est pas due à un problème de la politique de sélection de pièces, mais à un problème de dimensionnement de la source initiale.

Le but de la politique de sélection de pairs est d'utiliser au maximum la capacité d'envoi des pairs dans un environnement égoïste, c'est-à-dire lorsque chaque pair ne cherche qu'à maximiser son propre intérêt. Par conséquent, la politique de sélection de pairs doit inciter les pairs au partage et donc être robuste aux *free riders*. On montre dans la section 2.3.2 que la politique de sélection de pairs de BitTorrent fournit une réelle incitation au partage, mais qu'un mauvais dimensionnement de la source initiale supprime cette incitation.

Alors que l'on discute le problème de l'efficacité de BitTorrent pour les pairs des politiques de sélection de pièces et des pairs dans les section 2.3.1 et 2.3.2, on aborde dans la section 2.3.3 le problème de l'efficacité de BitTorrent pour les FAI. BitTorrent crée une charge importante sur les liens interFAI, mais on montre qu'une localisation du trafic est possible avec des gains importants sur les liens interFAI sans pénalité significative pour les pairs.

2.3.1 Sélection des pièces

Un contenu est découpé en pièces et chaque pièce est découpée en blocs. Le découpage en pièces est utilisé pour améliorer l'efficacité de BitTorrent ; Yang *et al.* [89] montrent que le temps de téléchargement diminue en fonction de l'inverse du nombre de pièces. La taille des pièces est fixée pour un torrent donné et varie en général de 128 kB à 4 MB. Le découpage des pièces en blocs permet de faire du traitement en pipeline des requêtes de blocs et, par conséquent, de compenser l'effet du RTT. La taille d'un bloc est toujours fixée à 16 kB.

Une pièce est appelée unité de réplication parce qu'un pair ne peut retransmettre une pièce que s'il a reçu tous les blocs de cette pièce et si l'intégrité de la pièce a été vérifiée. Un bloc est appelé une unité de transmission parce qu'un pair ne peut faire que des requêtes qui ont la granularité d'un bloc.

La politique de sélection de pièces consiste à choisir la pièce de laquelle des blocs vont être demandés. Cependant, une fois la pièce choisie, l'ordre de demande des blocs dans la pièce n'est pas spécifié par le protocole. Cet ordre n'a aucune importance puisque du moment qu'un pair offre une pièce, il a obligatoirement tous les blocs de cette pièce.

La politique de sélection de pièces de BitTorrent regroupe quatre stratégies : sélection *rarest first*, priorité stricte, sélection aléatoire, et *end game mode*.

La stratégie *rarest first* est la stratégie la plus connue de BitTorrent. Elle consiste pour un pair à télécharger en priorité les pièces les plus rares dans son voisinage. Le but de cette stratégie est de maximiser l'entropie des pièces.

La stratégie de priorité stricte consiste à toujours télécharger en priorité les blocs des pièces qui ont été partiellement téléchargées. Le but est de minimiser le temps pour obtenir une pièce

entière et donc minimiser le temps pour pouvoir retransmettre cette pièce. Cette politique préempte sur les trois autres, mais elle n'est pas bloquante, c'est-à-dire que si aucun voisin ne peut offrir les blocs des pièces partiellement téléchargées, alors le pair téléchargera les blocs d'autres pièces.

La stratégie de sélection aléatoire n'est utilisée que pour les quatre premières pièces téléchargées par un pair. Le but de cette stratégie est également de minimiser le temps pour obtenir une pièce. Cependant cette stratégie se focalise sur le début du téléchargement qui est critique pour l'efficacité de BitTorrent. En effet, lorsqu'un pair joint un torrent, il ne possède aucune pièce, il ne peut donc pas commencer un échange avec un autre pair puisqu'il n'a rien à offrir. Dès qu'il possédera quelques pièces à offrir, il pourra alors entrer dans une relation d'échange avec d'autres pairs, relation qui, comme nous le verrons en section 2.3.2, est le garant de l'efficacité et de l'incitation au partage.

Cette stratégie augmente la probabilité de compléter les premières pièces par rapport à la stratégie *rarest first*. En effet, nous expliquerons en section 2.3.2 que chaque pair donne pendant 30 secondes à un autre pair choisi au hasard (c'est l'*optimistic unchoke*). Si un pair envoie à 4 pairs en parallèle à une vitesse agrégée de 20 kB/s (ce qui correspond aux valeurs par défaut d'un client BitTorrent standard), cela représente 5 kB/s en réception par pair. Pour transmettre une pièce standard de 256 kB il faut 51 secondes. Il faudra donc un minimum de deux *optimistic unchokes* pour qu'un pair reçoive une pièce en entier. Les *optimistic unchokes* venant de pairs différents, si la pièce en cours de téléchargement est une pièce rare, il y a moins de chances que cette pièce puisse être servie par deux pairs différents faisant des *optimistic unchokes* consécutifs que si la pièce est choisie au hasard. La stratégie aléatoire n'est pas optimale, mais est un bon compromis entre simplicité et efficacité.

La dernière stratégie est le *end game mode*. Cette stratégie ne s'applique qu'à la fin du téléchargement lorsque le nombre de blocs restant à télécharger est inférieur au nombre de pairs pouvant les servir. Le but de cette stratégie est d'éviter le *terminaison idle time* décrit par Rodriguez *et al.* [83] qui se traduit par un ralentissement à la fin du téléchargement sur les derniers blocs. Cependant, cette stratégie est parfois assimilée à une solution au problème des dernières pièces (*last pieces problem*), ce qui est faux. La stratégie qui permet d'éviter ce problème est la stratégie *rarest first*.

On va dans la suite présenter la méthodologie que nous avons utilisée pour évaluer l'efficacité des quatre stratégies de la politique de sélection de pièces dans BitTorrent. Puis, nous allons présenter un résumé de nos résultats ; l'intégralité de nos résultats sur le sujet est disponible dans un rapport technique [63] et dans une publication à la conférence IMC'06 [64].

Méthodologie

Afin d'évaluer l'efficacité de la politique de sélection de pièces de BitTorrent, nous avons choisi une approche purement expérimentale. Comme nous l'avons discuté lors de la présentation

de l'état de l'art en section 2.2, l'approche expérimentale présente de nombreux avantages. Elle permet de valider les hypothèses faites dans les modèles et d'éviter les limitations des simulations. Lorsque ces expérimentations sont effectuées sur de vrais torrents, elles permettent, en outre, de mesurer le comportement de BitTorrent dans la réalité.

L'approche expérimentale est cependant difficile à mettre en œuvre. Dans cette étude, on a dû entièrement instrumenter un client BitTorrent. Nous avons choisi le client *mainline* dans sa version 4.0.2 de mai 2005 — on note que les politiques de sélection de pièces et de pairs de ce client sont toujours les mêmes dans les clients récents. À l'époque de ce travail, il n'existait, à notre connaissance, aucun client instrumenté ni aucune documentation sur le fonctionnement de BitTorrent autre que des descriptions succinctes [35] [5]. De plus, le code source du client ne contenait aucun commentaire. Le client que nous avons instrumenté et documenté est à notre connaissance le seul client BitTorrent entièrement instrumenté, commenté et librement disponible [6].

Lorsque l'on fait des expérimentations sur des torrents réels, il y a deux difficultés supplémentaires. La première difficulté est qu'il ne faut pas que la mesure perturbe le torrent mesuré ; un torrent dans lequel un grand nombre de clients BitTorrent sont instrumentés n'est plus un torrent réel, mais un torrent grandement contrôlé par l'expérimentateur. Afin que la mesure soit non-intrusive, nous avons décidé de n'avoir qu'un seul client instrumenté par torrent. Comme chaque client est connecté à un sous-ensemble aléatoire de 80 autres pairs, on considère qu'un seul client donne une vue représentative de la dynamique d'un torrent. On a également vérifié pour quelques torrents que toutes nos conclusions sont consistantes si on utilise trois clients instrumentés avec des voisinages disjoints. La deuxième difficulté est l'impossibilité de reproduire des expérimentations qui ont été faites sur des torrents réels. Pour remédier à ce problème, nous avons considéré un grand ensemble de torrents (voir table 2.1) que nous considérons représentatifs parce qu'avec une grande diversité de caractéristiques.

Nous avons mesuré chacun des torrents présentés dans la table 2.1 avec un seul client instrumenté. La mesure a duré huit heures soit environ quatre heures lorsque le client instrumenté est un *leecher* et quatre heures lorsqu'il est une source.

Résultats

Le but de ce travail est d'évaluer le niveau d'entropie qu'offre la politique de sélection de pièces de BitTorrent.

On appelle dans la suite une pièce *rare* une pièce qui n'existe que sur la source initiale, et une pièce *disponible* une pièce qui existe en au moins deux copies dans le torrent. Ainsi, un torrent dans un état *transitoire* est un torrent dans lequel il y a des pièces rares, et un torrent *stable* est un torrent dans lequel il n'y a plus de pièces rares, c'est-à-dire qu'il n'y a que des pièces disponibles.

Nous caractérisons l'entropie en utilisant une métrique que l'on appelle *disponibilité des pairs*.

TABLE 2.1 – Caractéristiques des torrents utilisés pour les expérimentations. **Colonne 1 (ID)** : Identifiant des torrents, **colonne 2 (# de S)** : nombre de sources au début de l'expérimentation, **colonne 3 (# de L)** : nombre de *leechers* au début de l'expérimentation, **colonne 4 (Rapport $\frac{S}{L}$)** : rapport (nombre de sources)/(nombre de *leechers*), **colonne 5 (PS max.) :** Taille maximale du voisinage lorsque le client instrumenté est un *leecher*, **colonne 6 (Taille)** : taille du contenu en MB.

ID	# de S	# de L	Rapport $\frac{S}{L}$	PS max.	Taille
1	0	66	0	60	700
2	1	2	0.5	3	580
3	1	29	0.034	34	350
4	1	40	0.025	75	800
5	1	50	0.02	60	1 419
6	1	130	0.007 8	80	820
7	1	713	0.001 4	80	700
8	1	861	0.001 2	80	3 000
9	1	1 055	0.000 95	80	2 000
10	1	1 207	0.000 83	80	348
11	1	1 411	0.000 71	80	710
12	3	612	0.004 9	80	1 413
13	9	30	0.3	35	350
14	20	126	0.16	80	184
15	30	230	0.13	80	820
16	50	18	2.8	40	600
17	102	342	0.3	80	200
18	115	19	6	55	430
19	160	5	32	17	6
20	177	4 657	0.038	80	2 000
21	462	180	2.6	80	2 600
22	514	1 703	0.3	80	349
23	1 197	4 151	0.29	80	349
24	3 697	7 341	0.5	80	349
25	11 641	5 418	2.1	80	350
26	12 612	7 052	1.8	80	140

La disponibilité du pair A d'après le pair B est le temps que B a été intéressé par A divisé par le temps que A a passé dans le voisinage de B. Une disponibilité de 1 signifie que B a toujours été intéressé par A. Lorsque tous les voisins de B ont une disponibilité de 1 on dit que l'on a une entropie idéale d'après B. Notons qu'une entropie idéale est impossible à atteindre en pratique ; nous cherchons cependant à savoir si la politique de sélection de pièces de BitTorrent est proche de cette entropie.

La figure 2.1 représente la disponibilité pour chacun des 26 torrents donnés dans la table 2.1 de chacun des voisins du pair instrumenté lorsqu'il est un *leecher* ; lorsqu'il est une source, la disponibilité de ses voisins est zéro puisqu'il n'est plus intéressé par aucune pièce. Les torrents sont ordonnés de celui qui a le moins de sources au début de la mesure à celui qui en a le plus.

On voit dans la figure 2.1 que l'entropie est proche de 1 pour tous les torrents qui ont

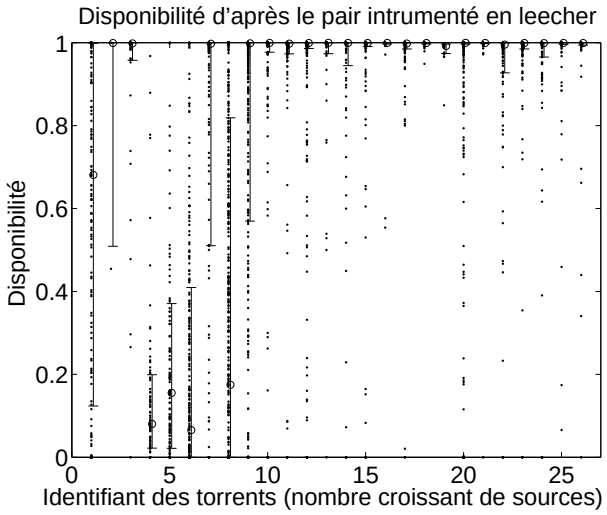

FIGURE 2.1 – Caractérisation de l'entropie. *Pour chaque voisin du pair instrumenté dans un torrent donné, un point représente la disponibilité de ce voisin d'après le pair instrumenté. Chaque ligne verticale représente le 2^e décile (bas de la ligne), la médiane (identifiée par le cercle) et le 8^e décile (haut de la ligne) de la disponibilité des voisins pour un torrent donné.*

strictement plus d'une source au début de la mesure (à partir du torrent 12), mais qu'elle est basse pour certains torrents avec au plus une source (torrents 1, 2, 4, 5, 6, 7, 8 et 9) ; on verra dans la suite que tous les torrents avec une mauvaise entropie sont des torrents dans un état transitoire. On va maintenant se concentrer sur le torrent 8 pour expliquer pourquoi l'entropie est mauvaise pour certains torrents et montrer qu'il ne s'agit pas d'un problème de la politique de sélection de pièces, mais de dimensionnement de la source initiale.

Le torrent 8 est un torrent avec une mauvaise entropie. Il a une source et 861 *leechers* au début de la mesure, et il est découpé en 863 pièces. On observe sur la figure 2.2 que l'évolution de la pièce la plus rare (courbe Min.) est à 0 pendant la majorité de la mesure. Cela signifie qu'il y a des pièces manquantes dans le voisinage du client instrumenté. La figure 2.3 donne l'évolution du nombre de pièces manquantes avec le temps. On observe que ce nombre diminue linéairement avec le temps. La pente de cette courbe donne la vitesse à laquelle de nouvelles pièces (les pièces rares) arrivent dans le voisinage du pair instrumenté. Comme la taille des pièces est de 4 MB, cette vitesse est de 36 kB/s.

Il est, par conséquent, très probable que le torrent 8 est dans un état transitoire, c'est-à-dire qu'il y a des pièces rares dans le torrent qui sont servies par la source initiale à 36 kB/s. Il existe donc le cycle suivant : dès que la source initiale sert une pièce rare, la pièce est rapidement répliquée par les pairs ; puis les pairs attendent une autre pièce de la source initiale. Durant cette période d'attente, aucun pair n'est intéressé par aucun autre pair, ce qui pénalise fortement la disponibilité globale.

On a donc montré que, paradoxalement, une mauvaise entropie pour les torrents dans un état transitoire — on a vérifié que tous les torrents avec une mauvaise entropie étaient effectivement

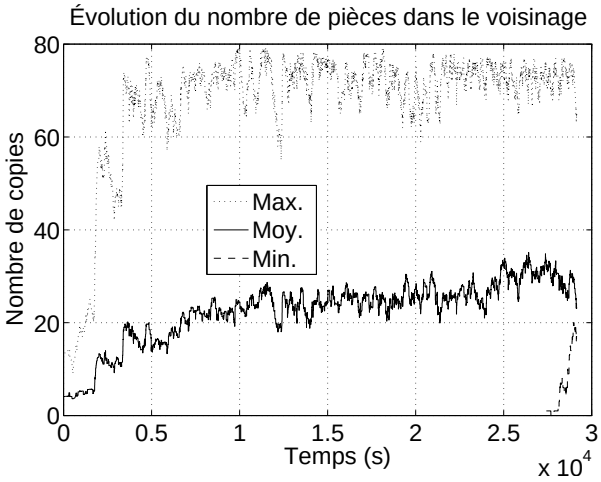

FIGURE 2.2 – Évolution du nombre de copies des pièces dans le voisinage du pair instrumenté lorsqu'il est un *leecher* dans le torrent 8. *La ligne en pointillés représente l'évolution de la pièce la plus répliquée dans le voisinage ; le ligne pleine représente l'évolution du nombre moyen de pièces dans le voisinage ; la ligne en tirets représente l'évolution de la pièce la plus rare dans le voisinage.*

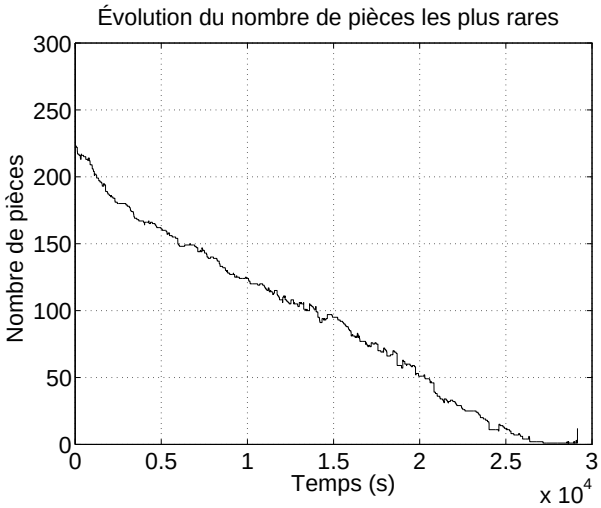

FIGURE 2.3 – Évolution du nombre de pièces les plus rares dans le voisinage du pair instrumenté lorsqu'il est un *leecher* pour le torrent 8. *Les pièces les plus rares forment l'ensemble des pièces les moins répliquées avec le même nombre de copies dans le voisinage.*

dans un état transitoire — est le signe d'une grande efficacité de BitTorrent. En effet, plus la réplication des pièces servies par la source initiale est efficace et plus la période d'attente sera longue, donc la disponibilité mauvaise.

Résumé des contributions

On a vu dans cette partie qu'il est possible d'observer des torrents avec une mauvaise entropie. Cependant, cette mauvaise entropie lorsqu'elle est observée pour des torrents dans un état

transitoire, et ce fut toujours le cas, est le signe d'une très bonne efficacité de BitTorrent. Il est cependant possible d'avoir des torrents dans un état transitoire avec une entropie proche de 1 (par exemple, les torrents 3, 10 et 11) lorsque la source initiale est suffisamment rapide. En conclusion, les problèmes d'entropie qui peuvent être observés sur de vrais torrents ne sont pas liés à une défaillance de la politique de sélection de pièces de BitTorrent, mais à un problème de dimensionnement de la source initiale.

2.3.2 Sélection de pairs

L'unique objectif de la politique de sélection de pairs de BitTorrent est de créer une incitation au partage. C'est en grande partie le succès de BitTorrent à atteindre cet objectif qui le différencie des autres protocoles pair-à-pair. Cet objectif, nous allons le voir, a pour conséquences une excellente utilisation des ressources disponibles et une grande robustesse aux *free riders*.

La politique de sélection de pairs va déterminer vers quels voisins un pair donné va accepter d'envoyer des blocs — on dit que chaque voisin sélectionné est *unchoke*. Ce sont ensuite les voisins sélectionnés qui choisissent les pièces à recevoir. La sélection des pairs est effectuée sur un cycle de 10 secondes, mais également à chaque fois qu'un pair sélectionné n'est plus intéressé ou quitte le voisinage, puisqu'il s'agit d'un évènement qui modifie la sélection.

La politique de sélection de pairs dans BitTorrent est différente suivant que le pair est un *leecher* ou une source. Mais, dans les deux cas, n voisins au maximum peuvent être *unchokes* et intéressés au même moment. Yang *et al.* [89] ont montré que l'optimal était autour de $n = 4$. Cependant, lorsqu'un pair a une capacité d'envoi supérieur à la moyenne, il est possible que $n = 4$ conduise à une sous-utilisation de sa capacité d'envoi. En conséquence, la valeur par défaut de n est 4, mais elle peut être augmentée en fonction d'une heuristique, spécifique à chaque client BitTorrent, dépendant de la capacité d'envoi du pair.

Lorsque le pair est un *leecher*, il suit l'algorithme suivant.

- Il ordonne ses voisins en fonction de la vitesse à laquelle ils lui envoient des blocs, du plus rapide au plus lent. Toutes les 10 secondes, il *unchoke* les $n - 1$ pairs les plus *rapides*, qui sont *intéressés*, c'est-à-dire qui ont besoin d'au moins une pièce que le pair a, et qui ont été *actifs* dans les trente dernières secondes.
- Tous les 3 cycles de 10 secondes, c'est-à-dire toutes les 30 secondes, un pair intéressé est choisi au hasard. Si ce pair ne fait pas partie des $n - 1$ plus rapides, il devient *unchoke*, sinon, un autre pair intéressé est choisi au hasard jusqu'à ce qu'il y en ait un qui ne fasse pas partie des $n - 1$ plus rapides ; il devient alors *unchoke*. On appelle le pair choisi au hasard et qui devient *unchoke* un *optimistic unchoke*.

L'*optimistic unchoke* a deux rôles. Le premier est de permettre aux nouveaux pairs d'obtenir leur première pièce. Sans *optimistic unchoke*, il n'y a aucun moyen pour un nouveau pair sans pièce d'obtenir une pièce puisqu'il n'a rien à offrir en échange. Le deuxième est de découvrir la capacité des voisins. Comme un pair ne donne que si on lui donne, le seul moyen de découvrir la

capacité d'un voisin est de lui donner gratuitement (c'est-à-dire sans rien demander en échange) pendant 30 secondes avec un *optimistic unchoke*. Il y a alors plusieurs cas possibles ; on suppose ici qu'un pair P fait un *optimistic unchoke* sur un voisin.

1. Si le pair P envoie assez vite pour être parmi les 3 pairs les plus rapides de son voisin, alors le voisin envoie en retour à P.
   (a) Si le voisin fait maintenant partie des trois pairs les plus rapides de P, alors P *unchoke* ce voisin et choisit un nouvel *optimistic unchoke*, les deux pairs sont ainsi entrés dans une relation de réciprocité stable ;
   (b) sinon, après la période de 30 secondes P choisit un nouvel *optimistic unchoke*.
2. Sinon, le voisin n'envoie jamais rien en retour et après la période de 30 secondes P choisit un nouvel *optimistic unchoke*.

On voit donc que la dynamique de la politique de sélection de pairs pour un *leecher* est complexe. On présente une analyse expérimentale de cette dynamique en section 2.3.2.

Lorsque le pair est une source, il existe deux versions de l'algorithme. La première version, qui est la version *initiale*, utilise le même algorithme que lorsque le pair est un *leecher* à la seule différence que les voisins sont ordonnés en fonction de la vitesse à laquelle ils reçoivent des blocs ; une source ne peut pas évaluer la vitesse d'envoi des voisins puisqu'elle n'a besoin d'aucun bloc. L'inconvénient de cet algorithme est que quelques *free riders* très rapides peuvent monopoliser une source et ainsi faire chuter l'efficacité du torrent. Pour remédier à ce problème, Bram Cohen a introduit expérimentalement dans le client *mainline* 4.0.2 un nouvel algorithme *équitable* qui se limite à faire un *unchoke* de 60 secondes sur n voisins choisis à tour de rôle et renouvelés en roulement. L'avantage de cette version est que tous les pairs reçoivent le même service de la source. Il n'est plus possible pour un pair de monopoliser la source. De plus, comme la capacité de la source est uniformément répartie dans le torrent, cela améliore la diversité des pièces et donc l'entropie.

Cependant, la version actuelle de l'algorithme pour une source est la version initiale. Même si cette version est moins robuste aux attaques que la version équitable, elle permet de favoriser les pairs qui sont très rapides. Il y a actuellement une forte corrélation entre la vitesse de réception d'un pair et sa vitesse de contribution. Par conséquent, il est plus important pour les clients BitTorrent actuels de donner aux pairs les plus rapides que d'être robustes aux *free riders*. Notons cependant que l'on ne s'attend pas à avoir une différence importante de performance entre les deux algorithmes dans les cas standards.

Dans la suite, on va exclusivement étudier la version équitable de l'algorithme pour les sources. On va maintenant présenter la méthodologie utilisée pour évaluer les propriétés de la politique de sélection de pairs, puis on présentera un résumé de nos contributions. L'intégralité de nos contributions sur le sujet est disponible dans un rapport technique [63], dans une publication à la conférence IMC'06 [64], et dans une publication à la conférence SIGMETRICS'07[62].

Méthodologie

Pour évaluer les propriétés de la politique de sélection de pairs, on a utilisé deux approches complémentaires.

Le première approche est la même que celle décrite en section 2.3.1. On a effectué des mesures sur 26 torrents avec un client instrumenté. Cette approche permet d'évaluer les propriétés de la politique de sélection de pairs sur de vrais torrents, mais avec une vision locale. Cependant, dans le cas de la politique de sélection de pièces, la représentativité globale de la vision locale de l'entropie dépend peu de la vitesse d'envoi du client instrumenté ; mais dans le cas de la politique de sélection de pairs, la dynamique globale dépend fortement des propriétés d'envoi des pairs. Pour cette raison, nous avons suivi une deuxième approche basée sur une instrumentation totale des torrents contrôlés.

Pour cette deuxième approche, nous avons créé des torrents synthétiques pour lesquels nous contrôlons tous les clients. Nous considérons des torrents de 41 clients instrumentés, 40 *leechers* et une source, que nous lançons sur PlanetLab [4], un client par nœud PlanetLab. Le fichier échangé fait 113 MB découpés en 453 pièces de 256 kB chacune. Tous les clients arrivent au même moment pour émuler un scénario de *flash crowd*. La source reste connectée durant toute la durée de l'expérimentation, mais les *leechers* se déconnectent dès qu'ils ont récupéré toutes les pièces du contenu.

On considère pour chaque expérimentation, c'est-à-dire pour chaque torrent, trois classes de *leechers* :

- une classe lente correspondant à 13 *leechers* ayant une vitesse d'envoi limitée à 20 kB/s,
- une classe moyenne correspondant à 14 *leechers* ayant une vitesse d'envoi limitée à 50 kB/s,
- une classe rapide correspondant à 13 *leechers* ayant une vitesse d'envoi limitée à 200 kB/s.

Pour évaluer l'impact de la vitesse d'envoi de la source initiale sur la dynamique de la politique de sélection de pairs, on effectue chaque expérimentation dans deux contextes différents. Dans le premier contexte, on considère une source initiale bien dimensionnée, c'est-à-dire avec une vitesse d'envoi maximale de 200 kB/s. Dans ce cas, la source est aussi rapide que les plus rapides des *leechers*. Dans le deuxième contexte, la source initiale est sous-dimensionnée ; sa vitesse d'envoi maximale est de 100 kB/s. Par conséquent, elle n'est pas assez rapide pour que les plus rapides des *leechers* puissent utiliser leur capacité d'envoi au maximum.

Résultats

Le principe de base de la politique de sélection de pairs dans BitTorrent est la réciprocité : lorsqu'un pair donne, il doit recevoir en retour. Nous avons étudié ce principe de réciprocité sur les 26 torrents présentés en table 2.1 (première approche méthodologique de la section 2.3.2).

La figure 2.4 montre que les voisins qui reçoivent le plus du pair instrumenté (figure du haut) sont également ceux qui donnent le plus au pair instrumenté (figure du bas) ; on en conclut une excellente réciprocité. On note également que les 5 pairs qui reçoivent le plus du pair instrumenté

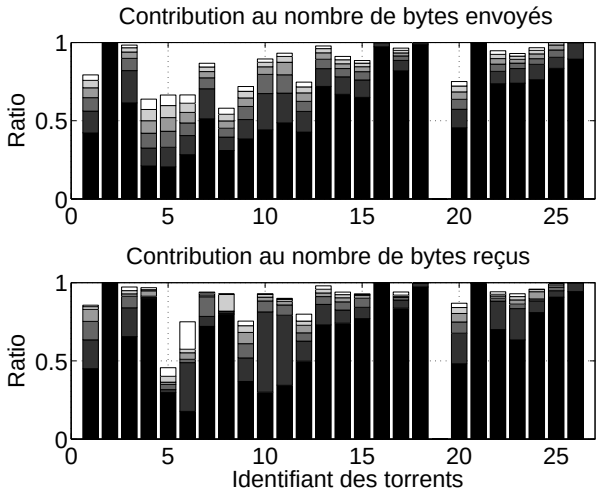

FIGURE 2.4 – Caractérisation de la réciprocité de la politique de sélection de pairs pour un leecher pour chacun des 26 torrents. ***Figure du haut :*** *Nombre de bytes envoyés par le pair instrumenté vers ses voisins. On a créé 6 groupes de 5 voisins chacun : le premier groupe (en noir) contient les 5 voisins qui ont reçu le plus, le deuxième groupe les 5 suivants, etc. jusqu'au cinquième groupe (en blanc) qui contient du 26^e au 30^e voisins qui ont reçu le plus.* ***Figure du bas :*** *Nombre de bytes reçus par le pair instrumenté de ses voisins. Les mêmes groupes que pour la figure du haut sont utilisés. Par conséquent, cette figure permet de voir combien les groupes qui reçoivent le plus donnent au pair instrumenté.*

(groupe noir) reçoivent une importante fraction de l'ensemble des données servies par le pair instrumenté ; cela montre que certains pairs sont choisis plus fréquemment. On note quelques exceptions à ces observations. Pour le torrent 19, le pair instrumenté n'envoie aucune donnée aux voisins ; il s'agit d'un résultat normal puisque tous les voisins sont dans ce cas des sources. Certains torrents ne présentent pas une excellente réciprocité (torrent 6) ou le groupe des 5 pairs qui reçoivent le plus ne reçoit pas significativement plus que les autres groupes (torrents 4, 5, 6, 8). Tous ces torrents sont dans un état transitoire, ce qui implique que les voisins ont une mauvaise disponibilité (au sens de la section 2.3.1). Par conséquent, l'équilibre vers lequel tend la politique de sélection de pairs est perturbé. Il s'agit ici encore d'un problème de dimensionnement de la source initiale et non d'un problème de la politique de sélection de pairs.

Afin de mieux comprendre comment sont choisis les voisins avec lesquels le pair instrumenté échange des blocs, nous nous concentrons sur le torrent 7. La figure 2.5 montre qu'il n'y a pas de corrélation entre le nombre d'*unchokes* que reçoit un voisin et le temps qu'il est intéressé par le pair instrumenté. On voit, en particulier, qu'un voisin qui est intéressé peu de temps (moins de 2 000 secondes) mais qui contribue reçoit beaucoup plus qu'un voisin qui est intéressé longtemps (plus de 8 000 secondes) mais qui ne contribue pas. On note cependant une corrélation linéaire de faible pente (voir les points en bas de la figure 2.5) qui est due à l'*optimistic unchoke* ; le nombre d'*unchokes* reçus par un *free rider* avec les *optimistic unchokes* est donc très inférieur au nombre d'*unchokes* reçus par un pair qui contribue.

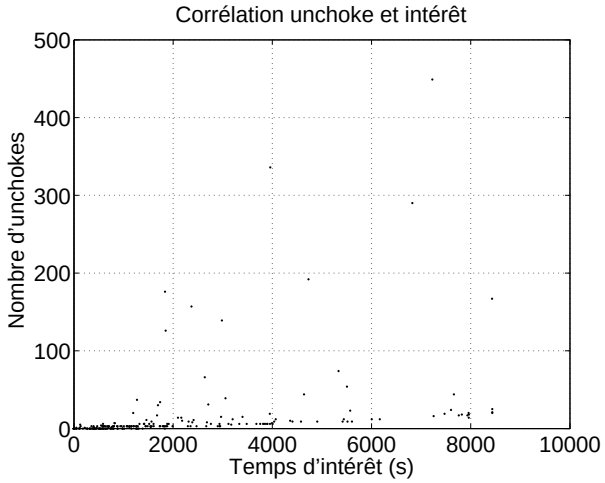

FIGURE 2.5 – Corrélation entre le nombre d'*unchokes* et la durée d'intérêt des voisins par le pair instrumenté pour le torrent 7. *Chaque point sur la figure représente la corrélation pour un voisin du pair instrumenté lorsque le pair instrumenté est un leecher.*

On peut donc conclure que la politique de sélection de pairs garantit une bonne réciprocité et que seul un petit nombre de pairs est sélectionné sur la durée d'un téléchargement, ce qui semble suggérer l'existence d'un équilibre.

Pour étudier cet équilibre et évaluer les propriétés globales (c'est-à-dire à l'échelle du torrent) de la politique de sélection de pairs, nous allons maintenant étudier des résultats d'expérimentations faites sur des torrents synthétiques tels que décrits dans la deuxième approche de la section 2.3.2.

On considère dans la suite que l'on a une source initiale bien dimensionnée, c'est-à-dire qui envoie au maximum à 200 kB/s. On observe dans la figure 2.6 que les pairs de même capacité se regroupent en clusters. Cela signifie que les pairs de même capacité préfèrent échanger des données entre eux.

On observe deux artefacts sur la figure 2.6. Le premier est que plus les pairs sont lents et plus les durées d'*unchokes* sont longues (les carrés sont plus foncés). En effet, les pairs lents mettent plus de temps à finir le téléchargement, ils ont donc plus de temps pour faire des *unchokes*. Le deuxième artefact est que le pair 27, qui est un pair moyen, *unchoke* souvent les pairs lents. Ce pair 27 était sur une machine fortement utilisée par d'autres applications (chaque pair est lancé dans une machine virtuelle sur un nœud PlanetLab). Il a souvent été contraint d'envoyer à moins de 50 kB/s, ce qui était insuffisant pour échanger avec les autres pairs moyens.

L'évaluation du temps de téléchargement en fonction de la capacité d'envoi des pairs donne une bonne indication de l'incitation au partage de la politique de sélection de pairs de BitTorrent. La figure 2.7 montre que le temps de téléchargement est fonction de la vitesse d'envoi des pairs : plus les pairs envoient rapidement et plus ils finissent leur téléchargement tôt. On voit donc que la politique de sélection de pairs fournit une bonne incitation au partage.

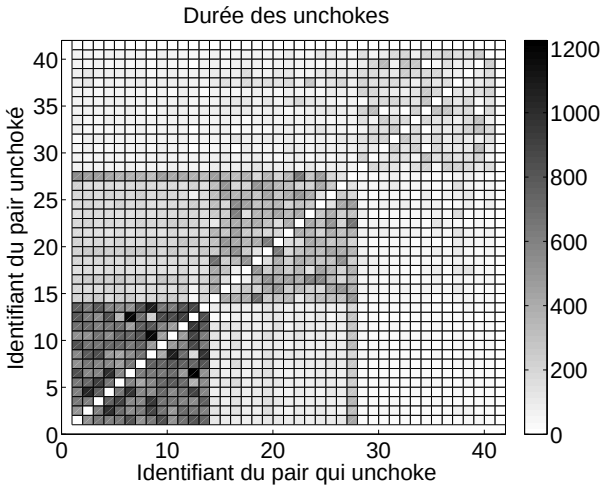

FIGURE 2.6 – Durée des *unchokes* de chaque pair vers chaque autre pair. *Les durées sont des moyennes sur 13 expérimentations indépendantes. Plus les carrés sont foncés et plus les durées d'unchokes sont longues (l'unité de la barre de couleur est la seconde). Les pairs 1 à 13 ont une capacité d'envoi maximale de 20 kB/s, les pairs 14 à 27 ont une capacité d'envoi maximale de 50 kB/s et les pairs 28 à 40 ont une capacité d'envoi maximale de 200 kB/s. La source initiale est le pair 41, c'est une source bien dimensionnée avec une capacité d'envoi maximale de 200 kB/s.*

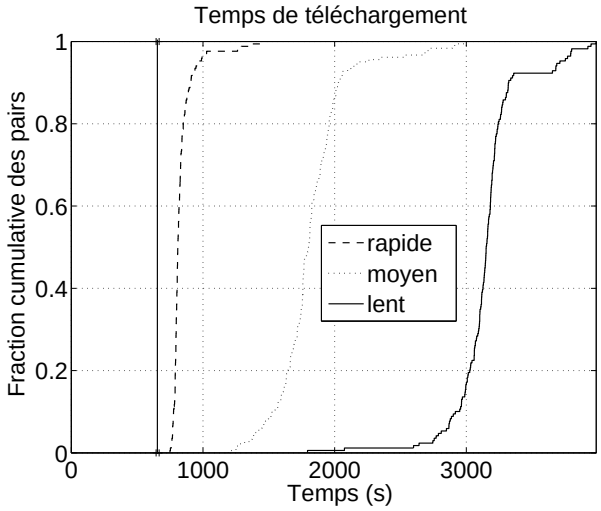

FIGURE 2.7 – Impact de la vitesse d'envoi sur le temps de téléchargement. *Distribution cumulative du temps de téléchargement pour chacune des trois classes de leechers sur 13 expérimentations indépendantes. La source initiale qui est bien dimensionnée envoie au maximum à 200 kB/s. La ligne noire verticale représente le temps d'envoi par la source initiale de la dernière pièce (c'est donc le temps minimum de téléchargement).*

L'incitation au partage est une contrainte sur le choix des pairs avec qui chaque autre pair va échanger des données. Cette incitation pourrait se traduire par une mauvaise utilisation de la capacité d'envoi des pairs. La figure 2.8 montre que l'incitation au partage n'a aucun impact négatif sur l'efficacité de BitTorrent. En effet, l'utilisation de la capacité d'envoi est proche de 1

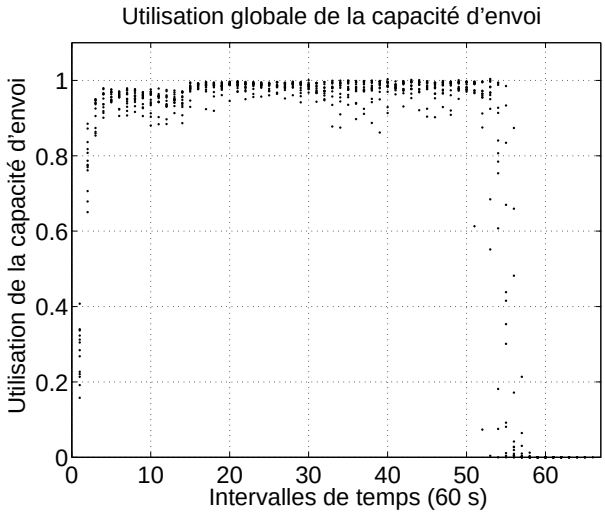

FIGURE 2.8 – Utilisation de la capacité d'envoi des pairs. *Chaque point représente l'utilisation moyenne de la capacité d'envoi maximale d'un pair sur un intervalle de 60 secondes. La source initiale qui est bien dimensionnée envoie au maximum à 200 kB/s.*

durant toute la durée de l'expérimentation.

On voit donc qu'avec une source initiale bien dimensionnée, les pairs de même capacité d'envoi se regroupent ensemble (ils forment des clusters), les pairs ont une incitation au partage et l'utilisation de la capacité d'envoi des pairs est très bonne. Cependant nous allons voir qu'un sous-dimensionnement de la source initiale va conduire à des propriétés de la politique de sélection de pièces totalement différentes.

On considère maintenant que l'on a une source initiale sous-dimensionnée, c'est-à-dire que sa capacité maximale d'envoi est de 100 kB/s. Cette source a donc une capacité inférieure aux plus rapides des pairs (200 kB/s), mais supérieure aux pairs lents (20 kB/s) et moyens (100 kB/s).

On voit dans la figure 2.9 qu'il n'y a plus de clusters de pairs. Cela signifie qu'il n'y a plus de regroupements de pairs en fonction de leur capacité d'envoi. Ce résultat surprenant illustre l'importance de dimensionnement de la source initiale. En effet, avec une source d'une capacité d'envoi juste inférieure aux plus rapides des pairs, on aurait pu s'attendre à observer des clusters de pairs lents et moyens.

Lorsque la source initiale n'est pas assez rapide pour que les pairs rapides utilisent leur capacité d'envoi au maximum, la disponibilité de ces pairs entre eux diminue ; elle est autour de 0.5 puisque la capacité de la source initiale est 50% plus petite que la capacité des pairs les plus rapides. Par conséquent, comme dans le cas des torrents dans un état transitoire, les pairs rapides vont faire des *unchokes* sur les pairs lents et moyens. Les pairs rapides vont être *unchokés* en retour ce qui va casser les clusters de pairs lents et moyens. Lorsque la source initiale envoie de nouvelles pièces, les pairs rapides vont de nouveau s'*unchoker* entre eux, ce qui empêche la formation de clusters entre les pairs rapides et les autres pairs. La répétition de ce cycle empêche

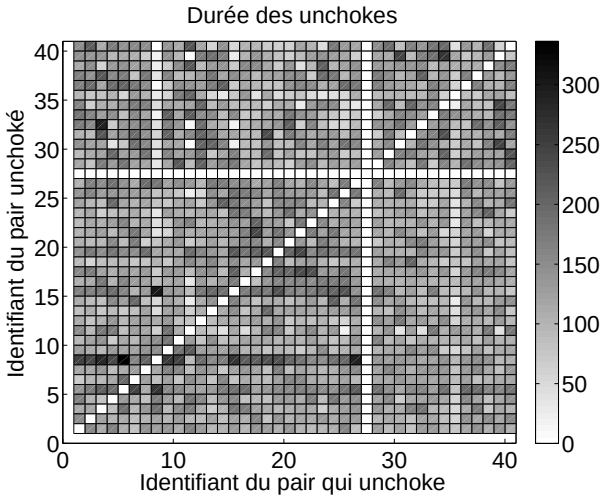

FIGURE 2.9 – Durée des *unchokes* de chaque pair vers chaque autre pair. *Les durées sont des moyennes sur 13 expérimentations indépendantes. Plus les carrés sont foncés et plus les durées des unchokes sont longues (l'unité de la barre de couleur est la seconde). Les pairs 1 à 12 ont une capacité d'envoi maximale de 20 kB/s, les pairs 13 à 26 ont une capacité d'envoi maximale de 50 kB/s et les pairs 28 à 40 ont une capacité d'envoi maximale de 200 kB/s. La source initiale est le pair 27, elle est sous-dimensionnée avec une capacité d'envoi maximale de 100 kB/s.*

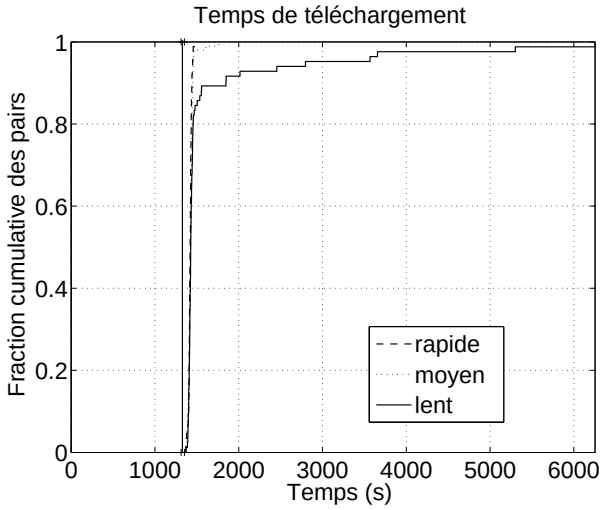

FIGURE 2.10 – Impact de la vitesse d'envoi sur le temps de téléchargement. *Distribution cumulative du temps de téléchargement pour chacune des trois classes de leechers sur 13 expérimentations indépendantes. La source initiale qui est sous-dimensionnée envoie au maximum à 100 kB/s. La ligne noire verticale représente le temps d'envoi par la source initiale de la dernière pièce (c'est donc le temps minimum de téléchargement).*

donc la formation de clusters entre n'importe quels pairs.

Il est donc suffisant que la source initiale soit d'une capacité juste inférieure aux plus rapides des pairs pour que plus un seul cluster ne se forme. Cependant, ce sous-dimensionnement de la source initiale a un impact bien plus important que l'absence de clusters (dont l'impact

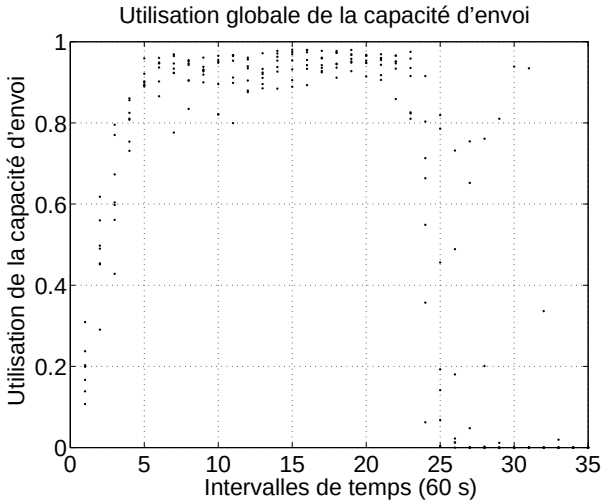

FIGURE 2.11 – Utilisation de la capacité d'envoi des pairs. *Chaque point représente l'utilisation moyenne de la capacité d'envoi maximale d'un pair sur un intervalle de 60 secondes. La source initiale qui est sous-dimensionnée envoie au maximum à 100 kB/s.*

sur les utilisateurs n'est pas évident) : un sous-dimensionnement de la source initiale supprime toutes incitations au partage. En effet, la figure 2.10 montre que le temps de téléchargement est maintenant indépendant de la capacité des pairs. Cependant, l'utilisation globale de la capacité d'envoi, voir figure 2.11, reste tout de même bonne (autour de 90% à l'équilibre) ; un torrent peut ne plus offrir aucune incitation au partage sans que cela ne se traduise par une baisse significative d'efficacité. Par contre, si l'on considère une source initiale avec une capacité maximale de 20 kB/s, l'utilisation globale de l'efficacité va chuter sans surprise à moins de 20%.

Il faut faire ici une distinction entre l'impact du manque d'incitation au partage sur l'efficacité théorique et sur l'efficacité pratique. On considère dans ces expérimentations des pairs qui collaborent sans incitation. Par conséquent, on a montré que l'efficacité théorique, c'est-à-dire l'efficacité maximale que l'on peut atteindre avec le comportement le plus favorable des pairs, peut rester bonne même s'il n'y a plus d'incitation au partage. Par contre, l'efficacité pratique est celle que l'on obtient avec des pairs réels qui ont un comportement égoïste (*selfish*). Dans ce cas, il est très difficile d'évaluer l'impact du manque d'incitation. En effet, comme les clients BitTorrent actuellement déployés ne cherchent pas à minimiser l'utilisation de la capacité d'envoi, le choix de cette capacité est exclusivement une action de l'utilisateur. L'impact de la capacité d'envoi sur le téléchargement est très difficile à évaluer pour les utilisateurs car il varie d'un torrent à l'autre (et même au cours de la vie du torrent) en fonction du dimensionnement de la source initiale et du nombre de sources disponibles par rapport au nombre de *leechers*.

Cependant, des clients comme BitTyrant [72] (voir section 2.2.4) qui cherchent à optimiser localement l'utilisation de la capacité d'envoi vont conduire à une très mauvaise efficacité en cas de sous-dimensionnement de la source initiale. En effet, dans ce cas, comme la vitesse d'envoi n'a pas d'impact direct sur la vitesse de réception, le client va diminuer cette vitesse d'envoi

jusqu'à la valeur minimum. Ce problème confirme que l'optimisation locale de la vitesse d'envoi est une mauvaise idée dans le contexte de BitTorrent.

Résumé des contributions

On a vu dans la section 2.3.1, que le bon dimensionnement de la source initiale était fondamental pour obtenir une bonne diversité des pièces et donc pour avoir une bonne efficacité. On a montré ici que ce dimensionnement déterminait la capacité de la politique de sélection de pairs à offrir une incitation au partage et, par conséquent, d'être robuste aux *free riders*. Ce résultat explique également pourquoi dans certains cas, des expérimentations peuvent montrer qu'un *free rider* a une vitesse de téléchargement équivalente à un pair qui contribue. Il montre aussi que la solution ne vient pas d'une tentative d'optimisation locale de la capacité d'envoi, mais d'une augmentation du dimensionnement de la source initiale.

2.3.3 Localisation du trafic

On a montré jusqu'ici que les politiques de sélection de pièces et de pairs de BitTorrent étaient très efficaces pour les fournisseurs de contenus et pour les utilisateurs. BitTorrent permet aux fournisseurs de contenus de pouvoir distribuer à grande échelle des contenus sans avoir à payer pour une coûteuse infrastructure ; et les utilisateurs perçoivent une excellente efficacité. Cependant, il y a un troisième acteur dont on n'a pas parlé jusqu'ici : l'opérateur réseau. Cet opérateur est celui qui fournit l'infrastructure réseau nécessaire à l'échange des données. Il peut s'agir d'un opérateur de transit ou d'un opérateur fournissant l'accès aux utilisateurs terminaux. Dans la suite, on utilisera de manière générique le terme fournisseur d'accès à Internet (FAI).

Les FAI sont interconnectés entre eux avec principalement deux types de connexions suivant le rôle respectif de chaque FAI. Les connexions de transit sont entre deux FAI qui ont un accord de transit, c'est-à-dire qu'un FAI est fournisseur d'accès (on dit de transit) pour un autre FAI. Dans ce cas, les flux de données sont asymétriques entre ces FAI, le FAI de transit fournissant plus de données qu'il n'en reçoit. Le FAI de transit est donc payé par l'autre FAI en fonction des données échangées [70]. Les connexions de *peering* sont entre deux FAI qui ont une relation d'égal à égal, c'est-à-dire que les flux de données entre eux sont équilibrés. Dans ce cas, il n'y a pas de facturation pour les données échangées tant que les flux entre les FAI restent équilibrés.

On voit donc que l'économie des FAI dépend grandement de la planification du trafic et du dimensionnement adéquate des infrastructures. Cependant, avec une distribution de contenus pair-à-pair, les connexions entre les pairs sont dynamiques (c'est-à-dire changent souvent) et établies au hasard ; les flux de données sont très volatiles et difficilement prédictibles. La planification et le dimensionnement devient difficile. Si la quantité de trafic sur une connexion de transit change significativement ou que l'équilibre sur une connexion de *peering* est rompu, l'impact financier sur le FAI peut être important.

À cause de ce problème, les FAI sont hostiles au trafic pair-à-pair. Ils ne veulent pas payer

le prix de l'augmentation de trafic générée par une distribution pair-à-pair qui est utilisée par les fournisseurs de contenus pour économiser sur l'infrastructure.

Le concept de localité pour le trafic pair-à-pair a été introduit par Karagiannis *et al.* [59]. Le principe est de favoriser les connexions entre pairs dans un même FAI afin de minimiser la redondance sur les liens (de transit ou de *peering*) interFAI. Les nombreux travaux qui ont suivi (voir section 2.2.5) ont amélioré la compréhension de la localisation du trafic pair-à-pair. Ils ont cependant laissé en suspens deux questions fondamentales :

– jusqu'où peut-on localiser le trafic pair-à-pair ?
– quelle réduction de trafic interFAI peut être obtenue à l'échelle d'Internet ?

Dans la suite, on va commencer par décrire la politique de localité que l'on utilise, puis on va présenter un résumé de nos réponses à ces deux questions. L'intégralité de nos contributions est disponible dans une publication journal Computer Networks [27].

Politique de localité

La politique de localité que l'on considère dans la suite est basée sur une modification du tracker. Suivant le scénario de déploiement d'une politique de localité, l'implémentation peut se baser sur une modification des clients [34] ou du tracker, ou sur l'utilisation d'une infrastructure dédiée [88]. Quelle que soit l'implémentation réelle de la politique de localité, ses effets sur la réduction de trafic interFAI et sur la vitesse de téléchargement des pairs seront les mêmes. Par conséquent, tous les résultats que l'on présente ici restent valables pour d'autres implémentations d'une politique de localité.

On décrit maintenant notre politique de localité. On dit qu'une connexion interFAI est sortante pour un FAI lorsque c'est un pair de ce FAI qui a établi la connexion. Cependant, rien ne distingue une connexion sortante d'une connexion entrante en pratique ; il s'agit ici d'une simple distinction logique. Le tracker maintient pour chaque FAI le nombre de connexions sortantes pour cet FAI, et un paramètre global pour l'ensemble des FAI : le nombre maximal de connexions sortantes par FAI. À chaque fois qu'un pair contacte le tracker, il est associé par le tracker à son FAI. Le tracker vérifie si le nombre de connexions sortantes pour cet FAI a atteint le maximum. Si c'est le cas, le tracker retourne au pair un sous-ensemble aléatoire de pairs du même FAI. Si ça n'est pas le cas, le tracker retourne un sous-ensemble aléatoire de pairs du même FAI et un pair à l'extérieur du FAI.

Cette politique de localité s'applique à tous les pairs, source ou *leecher*, mais pas à la source initiale. Comme on l'a vu précédemment (voir sections 2.3.1 et 2.3.2), le rôle de la source initiale est critique pour garantir une bonne efficacité du torrent durant son démarrage. C'est pourquoi, il est important que la source initiale puisse propager de nouvelles pièces dans tous les FAI où il y a des pairs pour le torrent de cette source.

En plus de cette politique, on introduit deux optimisations. La première s'appelle la stratégie de répartition de charge (RC). Avec la politique de localité, lorsque le nombre maximal de

connexions interFAI n'a pas encore été atteint, le tracker retourne un pair choisi au hasard dans n'importe quel autre FAI. Par conséquent, les gros FAI ont une plus forte probabilité d'avoir un de leurs pairs retourné par le tracker, donc d'avoir un plus grand nombre de connexions interFAI entrantes. Pour éviter ce problème, avec la stratégie de répartition de charge, le tracker va en premier choisir un FAI au hasard (parmi les FAI qui ont au moins un pair pour le torrent donné), et ensuite il va choisir un pair au hasard dans le FAI sélectionné. Ainsi, le nombre de connexions entrantes par FAI devient indépendant de la taille du FAI.

La deuxième stratégie s'appelle fusion de partitions (FP). L'apparition de partitions est en effet un des plus gros risques lorsqu'une politique de localité est utilisée. Lorsque des pièces du contenu sont manquantes dans une partition, aucun pair de la partition ne pourra finir le téléchargement. Pour résoudre ce problème, avec la stratégie de fusion de partitions, chaque pair surveille si ses voisins reçoivent de nouvelles pièces (grâce aux messages *HAVE* reçus). Si après un certain délai, aucun voisin ne reçoit de nouvelles pièces, le pair contacte le tracker pour lui demander un pair à l'extérieur de son FAI. Ainsi, les partitions sont rapidement détectées et fusionnées. Les mécanismes qui évitent une explosion des retours et une exploitation malicieuse pour contourner la politique de localité sont décrits dans la publication [27] (section 2.3).

Jusqu'où peut-on localiser le trafic pair-à-pair ?

Le seul paramètre à configurer dans notre politique de sélection de pairs est le nombre maximal de connexions interFAI sortantes. Ce paramètre est un compromis entre réduction de trafic interFAI et temps de téléchargement des pairs. En effet, plus le graphe d'interconnexion des pairs est contraint (en restreignant le nombre de connexions interFAI), plus il est difficile d'avoir une bonne diversité des pièces et donc un bon temps de téléchargement.

Les travaux précédents ont soit contourné ce problème en fixant à 20% le nombre de connexions interFAI [88] soit choisi arbitrairement un nombre fixe de connexions interFAI pour un scénario donné sans en évaluer les conséquences pour d'autres scénarios [26].

On va présenter ici des résultats que l'on a obtenus en faisant des expérimentations BitTorrent contrôlées à grande échelle. On utilise le client BitTorrent instrumenté décrit en section 2.3.1. Les pairs échangent un contenu de 100 MB découpé en pièces de 256 kB. Tous les pairs démarrent dans les 60 premières secondes de l'expérimentation. (On a également fait des expérimentations spécifiques sur le *churn* durant lesquelles les pairs peuvent démarrer sur de grandes périodes de temps. On ne présentera pas ici ces expérimentations ; elles sont décrites dans la version journal [27].)

Toutes les expérimentations sont réalisées sur la plate-forme d'expérimentation Grid'5000 [2]. Il s'agit d'une fédération de clusters qui peuvent être réservés pour faire du calcul scientifique ou des expérimentations réseaux. Les avantages de Grid'5000 par rapport à PlanetLab [4] sont nombreux. Grid'5000 permet une réservation totale des ressources dont une expérimentation a besoin. En particulier, les nœuds des clusters ne sont pas virtualisés comme dans PlanetLab, il n'y

a donc pas d'interactions entre expérimentations, les résultats sont reproductibles et consistants dans l'espace (c'est-à-dire indépendants des nœuds utilisés) et le temps. Grid5000 est donc une plate-forme d'expérimentation beaucoup plus adaptée que PlanetLab aux expérimentations réseaux dont le but est l'évaluation de performance. Il y a cependant quelques inconvénients importants à l'utilisation de Grid5000 : les délais réseaux présents dans un cluster sont inférieurs à la milliseconde, les délais ne sont donc pas réalistes ; le réseau interconnectant les clusters est dédié, il n'y a donc pas de trafic exogène ni de congestion. On a montré dans un travail publié à P2P'10 [80] (mais qu'on ne présente pas dans cette thèse) que ce manque de réalisme dans les délais et le trafic exogène n'avait aucun impact significatif sur les expérimentations BitTorrent.

Pour chaque expérimentation, on lance jusqu'à 100 clients BitTorrent sur un seul nœud de Grid5000 ; on a vérifié que l'on pouvait lancer jusqu'à 150 clients BitTorrent envoyant au maximum à 20 kB/s sans dégradation de performance significative.

Pour émuler la notion de FAI et de connexions interFAI, avant chaque expérimentation, on assigne chaque pair à un FAI logique. Lorsque l'expérimentation est terminée on utilise cette assignation logique pour calculer le trafic interFAI.

Dans les expérimentations suivantes, on va évaluer l'impact du nombre maximal de connexions sortantes par FAI sur deux métriques : *la redondance* qui représente le nombre de copies du contenu qui sortent par les connexions interFAI ; *le ralentissement* qui représente le temps de téléchargement des pairs normalisé par le temps optimal de téléchargement, c'est-à-dire lorsque tous les pairs utilisent leur capacité d'envoi au maximum.

On fait varier le nombre de connexions interFAI sortantes de 4 à 3 600 (qui correspond au cas de BitTorrent sans localité) dans un scénario avec 1 000 pairs et 10 FAI. On considère ensuite trois scénarios :

- un scénario homogène avec une source initiale lente dans lequel la vitesse maximale d'envoi de tous les pairs est 20 kB/s ;
- un scénario hétérogène dans lequel les pairs sont répartis en trois classes de vitesses maximales d'envoi (20 kB/s, 50 kB/s et 100 kB/s), la source initiale faisant partie de la classe la plus rapide ;
- un scénario homogène avec une source initiale rapide dans lequel la vitesse maximale d'envoi de tous les pairs est 20 kB/s sauf pour la source initiale qui a une vitesse de 100 kB/s.

La figure 2.12 montre que la redondance diminue avec le nombre maximal de connexions interFAI sortantes. De plus, par rapport au cas de BitTorrent sans localité (c'est-à-dire avec 3 600 connexions interFAI sortantes par FAI), on observe une réduction de la redondance de deux ordres de grandeur lorsque le nombre de connexions est seulement 4.

Une réduction de la redondance n'est pas surprenante lorsque l'on réduit le nombre de connexions interFAI sortantes puisque l'on contraint le graphe d'interconnexion des pairs pour éviter les échanges de données entre FAI. Cependant, il est important d'évaluer l'impact de cette

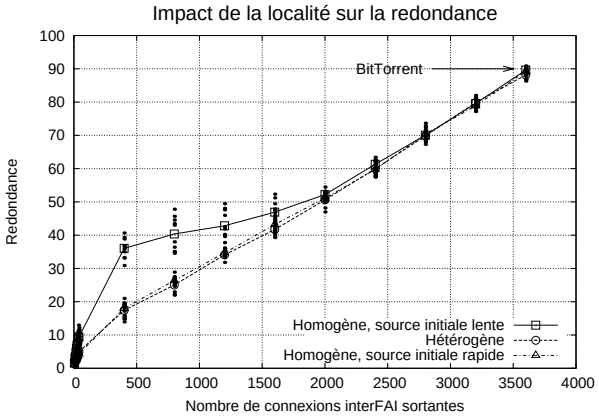

FIGURE 2.12 – Redondance avec 1 000 pairs et 10 FAI. *Chaque carré, cercle et triangle représente la redondance moyenne sur tous les FAI pour un scénario donné. Chaque point représente la redondance pour un FAI.*

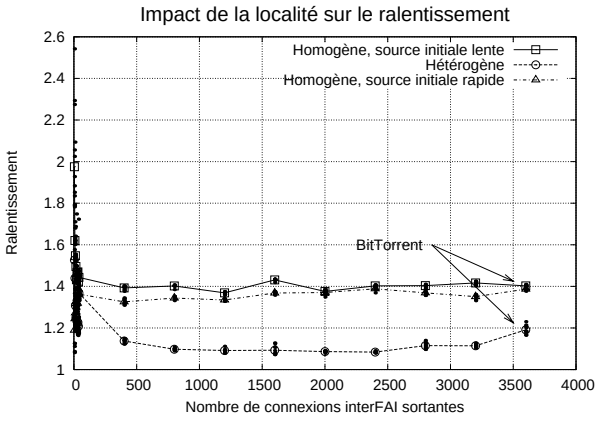

FIGURE 2.13 – Ralentissement avec 1 000 pairs et 10 FAI. *Chaque carré, cercle et triangle représente le ralentissement moyen sur tous les FAI pour un scénario donné. Chaque point représente le ralentissement pour un FAI.*

contrainte sur le ralentissement des pairs.

On voit sur la figure 2.13 que, dans la majorité des scénarios, la réduction du nombre de connexions n'a pas d'impact sur le ralentissement. On constate que dans le pire des cas, lorsqu'il n'y a que 4 connexions interFAI, le ralentissement augmente de 43%. Même si cette augmentation n'est pas négligeable, elle doit être évaluée avec la réduction de la redondance qui est proche de 99% dans le même scénario.

Le ralentissement observé est celui obtenu lorsqu'il n'y a pas de congestion sur les liens interFAI. En pratique, les liens interFAI sont des liens fortement chargés qui constituent des goulots d'étranglement pour les connexions interFAI. Par conséquent, une localisation du trafic pair-à-pair peut même améliorer le ralentissement en cas de forte congestion sur les liens interFAI (voir section 5.3 dans la publication Computer Networks [27]).

En résumé, BitTorrent est très robuste à la localisation de son trafic. On a en effet montré que l'on pouvait réduire de manière très importante le trafic sur les liens interFAI avec un impact faible sur le temps de téléchargement des pairs.

Réduction de trafic interFAI à l'échelle d'Internet ?

La réduction de trafic interFAI que l'on obtient en localisant les connexions pair-à-pair est une fonction du nombre de pairs dans les FAI. Si un FAI n'a jamais plus d'un pair par torrent, il n'y a aucun gain de trafic interFAI possible pour ce FAI. Pour comprendre l'impact pratique de la localisation du trafic BitTorrent, il est important de connaître la distribution des pairs BitTorrent par AS dans la réalité. On a pour cela conçu un *crawler* qui, dans une première phase de 6 heures, a collecté 214 443 fichiers `.torrent` uniques (sur www.mininova.org le 11 décembre 2008) correspondant à des torrents avec au moins une source et un *leecher*. On a ensuite, pour chaque torrent, collecté au minimum 90% des adresses IP des pairs présents dans le torrent. Cette deuxième phase qui a duré 12 heures a permis de collecter 6 113 224 adresses IP uniques réparties sur 9 605 AS différents. Étant donné que la collecte des adresses IP est faite de manière séquentielle sur l'ensemble des torrents et que le traitement pour un torrent est de l'ordre de la seconde, on peut considérer que pour un torrent donné toutes les adresses IP collectées le sont pour des pairs présents au même moment dans le torrent.

On a constaté que sur l'ensemble des torrents collectés 117 677 torrents et 6 643 AS ne pouvaient pas bénéficier d'une politique de localité parce qu'il y avait au plus un pair par AS et par torrent.

On a associé pour chaque adresse IP l'AS d'origine. Il est cependant très difficile de faire automatiquement le lien entre un AS et un FAI. Mais la granularité de l'AS donne une première approximation raisonnable des FAI. Dans la suite, on utilisera la notion de FAI et de connexions interFAI en faisant l'hypothèse (simplificatrice) qu'il y a une bijection entre l'ensemble des FAI et des AS.

Pour comprendre l'impact d'une politique de localité sur de vrais torrents — avec notamment une distribution réaliste de pairs par FAI, on va commencer par se focaliser sur 3 torrents représentatifs que l'on appelle torrents de référence dans la suite. Le torrent 1 distribue un film en langue anglaise, il représente donc un torrent largement réparti géographiquement. Il a 9 844 pairs répartis sur 1 043 FAI, le FAI le plus représenté a 386 pairs. Le torrent 2 distribue un film en italien, il est donc davantage localisé géographiquement que le torrent 1. Il a 4 819 pairs répartis sur 211 FAI, le FAI le plus représenté a 2 415 pairs. Le torrent 3 distribue un jeu ; il a 996 pairs sur 354 FAI, le FAI le plus représenté a 31 pairs. Ce torrent est représentatif des torrents de taille moyenne pour lesquels les FAI ne peuvent pas beaucoup réduire le trafic interFAI.

On a fait des expérimentations utilisant la distribution des pairs par FAI des trois torrents et les paramètres décrits en section 2.3.3 pour le scénario homogène avec une source initiale lente.

La redondance par FAI sans politique de localité (carrés dans la figure 2.14) augmente

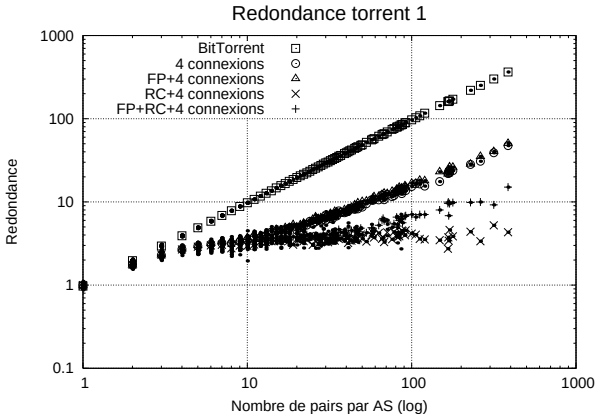

FIGURE 2.14 – Redondance pour le torrent 1. *Chaque symbole (rectangle, triangle, cercle, signe plus, signe multiplié) représente la redondance moyenne sur tous les FAI avec le même nombre de pairs pour un scénario donné. Chaque point représente la redondance pour un seul FAI.*

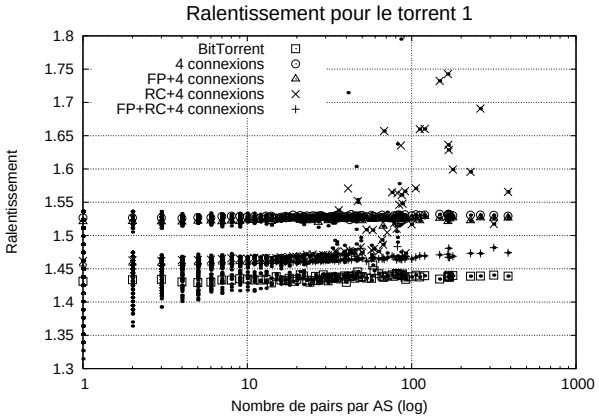

FIGURE 2.15 – Ralentissement pour le torrent 1. *Chaque symbole (rectangle, triangle, cercle, signe plus, signe multiplié) représente le ralentissement moyen sur tous les FAI avec le même nombre de pairs pour un scénario donné. Chaque point représente le ralentissement pour un seul FAI.*

linéairement avec le nombre de pairs dans cet FAI. La politique de localité avec 4 connexions interFAI sortantes permet une réduction de trafic importante, mais avec une augmentation de trafic qui est toujours linéaire en fonction du nombre de pairs par FAI. Comme expliqué en section 2.3.3, même s'il y a une contrainte sur le nombre de connexions interFAI sortantes, les FAI ont un nombre de connexions interFAI entrantes qui augmente avec le nombre de pairs dans le FAI. Pour résoudre ce problème, on a introduit l'optimisation de répartition de charge (RC) ; on voit dans la figure 2.14 (signe multiplié) que la redondance est maintenant indépendante du nombre de pairs par FAI.

Cependant, le ralentissement est plus important avec l'optimisation RC, voir figure 2.15 (signe multiplié), car la réduction du nombre de connexions interFAI entrantes augmente le risque de partitions. Pour résoudre ce second problème, on a introduit une deuxième optimisation :

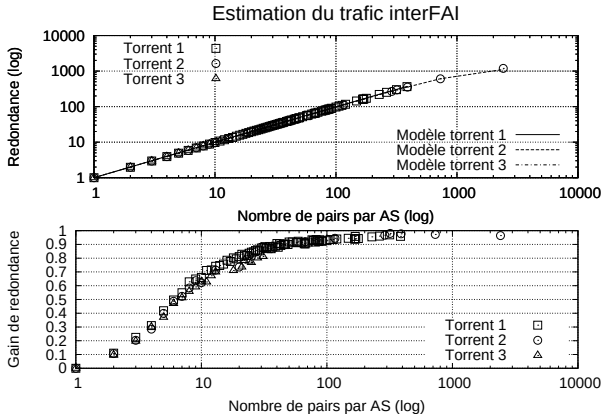

FIGURE 2.16 – La figure du haut montre la redondance expérimentale (symboles) et modélisée (lignes) pour les trois torrents de référence sans politique de localité. La figure du bas montre le gain de redondance qui est défini par $1 - \frac{R_L}{R_S}$ où R_L est la redondance avec 4 connections interFAI avec les optimisations RC et FP, et R_S est la redondance sans localité.

la fusion de partitions (FP). On voit dans la figure 2.14 (signe plus) que la redondance est légèrement supérieure avec les optimisations FP+RC, mais que le ralentissement est maintenant proche de celui obtenu par BitTorrent sans localité (figure 2.15, signe plus) quelle que soit la taille des FAI.

On va maintenant se concentrer sur la réduction de trafic interFAI possible à l'échelle d'Internet avec une politique de localité. On fait pour cela deux hypothèses sur les 6 113 224 pairs dont on a collecté les adresses IP ; ces hypothèses simplifient la quantification du trafic interFAI sur les torrents mesurés. Premièrement, on suppose que tous les pairs démarrent leur téléchargement simultanément et qu'ils restent connectés au torrent tout le temps de leur téléchargement. Cette hypothèse consiste essentiellement à supposer que tous les pairs d'un torrent donné téléchargent effectivement le contenu, ce qui est raisonnable. Deuxièmement, on suppose que n'importe quel pair peut faire un *unchoke* sur n'importe quel autre pair. En pratique, comme on l'a vu en section 2.3.2, les pairs se regroupent en clusters en fonction de leur vitesse d'envoi. Cependant, le voisinage est sélectionné aléatoirement sur l'ensemble des pairs d'un torrent donné. Les pairs qui reçoivent un *unchoke* sont donc en première approximation choisis au hasard sur l'ensemble des pairs ; le regroupement ne s'effectue qu'à l'intérieur du voisinage choisi aléatoirement.

On commence par évaluer le trafic interFAI sans politique de localité et ensuite on évalue la réduction de trafic avec la politique de localité. On suppose que le trafic envoyé par les pairs d'un FAI A vers d'autres FAI pour un torrent donné (distribuant un contenu de taille C) est uniquement fonction du nombre de pairs S_T dans le torrent et du nombre de pairs S_A dans le FAI ; ce trafic est $(1 - \frac{S_A}{S_T}) \cdot S_A \cdot C$.

Pour valider ce modèle, on a fait des expérimentations sur la plate-forme décrite en section 2.3.3 en utilisant la distribution de pairs par FAI des trois torrents de référence. On constate, dans la figure 2.16 (figure du haut), que bien que ce modèle soit simple la redondance par FAI

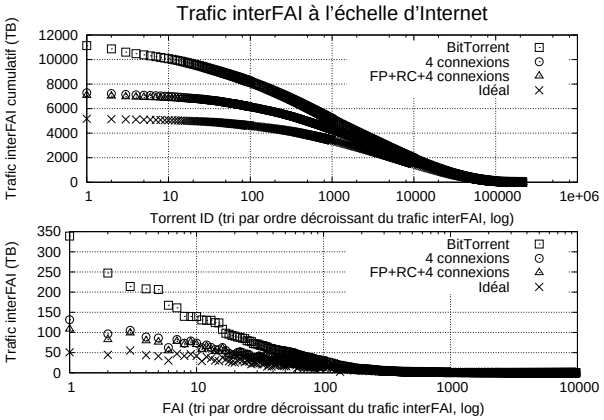

FIGURE 2.17 – Estimation du trafic cumulé interFAI pour tous les torrents (figure haut) et le trafic interFAI par FAI (figure du bas).

obtenue expérimentalement correspond fidèlement à celle obtenue par notre modèle. On va donc utiliser dans la suite ce modèle pour évaluer la quantité de trafic interFAI sans politique de localité.

Pour estimer la réduction de trafic interFAI avec une politique de localité, on a fait des expérimentations sur les 3 torrents de référence avec 4 connexions interFAI et les optimisations RC et FP. La figure 2.16 (figure du bas) montre que le gain de redondance est uniquement fonction du nombre de pairs par FAI, mais pas de la taille du torrent. On utilise dans la suite la moyenne du gain de redondance sur les trois torrents de référence.

Pour estimer le volume total de trafic interFAI généré par BitTorrent et la réduction de trafic avec une politique de localité pour les 214 443 torrents, on a calculé en utilisant les résultats précédents pour chaque torrent et chaque FAI le trafic interFAI : sans politique de localité, avec 4 connexions interFAI, avec 4 connexions interFAI et les optimisations RC et FP. On dit que le trafic interFAI est idéal lorsque la redondance est de 1 pour chaque torrent et chaque FAI.

La figure 2.17 (figure du haut) montre que le trafic interFAI sans localité est de 11.6 petabytes. Les 100 plus gros torrents génèrent 26% de tout ce trafic ; les 10 000 plus gros génèrent 82%. Il est étonnant de constater que peu de torrents génèrent une très grosse quantité de trafic. La politique de localité avec 4 connexions interFAI et les optimisations RC et FP permet une forte réduction du trafic interFAI ; dans ce cas, le trafic descend à 7 petabytes ce qui est seulement 35% plus grand que le cas idéal.

La représentation par FAI (figure 2.17, figure du bas) montre que les 50 plus gros FAI génèrent 45% de tout le trafic interFAI. Cependant, ces FAI bénéficient largement d'une politique de localité.

En résumé, la réduction de trafic à l'échelle d'Internet peut être très importante avec une politique de localité

Résumé des contributions

Il est possible de réduire le nombre de connexions interFAI à seulement 4 ; cela se traduit par une forte réduction du trafic sans impact notable sur les utilisateurs. De plus, on a montré que sur une distribution réelle de pairs par FAI, il était possible de réduire de 40% le trafic interFAI à l'échelle d'Internet.

On note cependant que la localisation du trafic pair-à-pair se traduit par une augmentation du trafic *intraFAI*. Dans certains cas, cette augmentation n'est pas souhaitable. Par exemple, les câblo-opérateurs utilisent comme boucle finale un médium partagé entre un grand nombre d'utilisateurs. Une augmentation du trafic local entre ces utilisateurs peut conduire à une saturation de la boucle locale. Les opérateurs historiques ne sont pas toujours mieux armés, même si l'évolution de leur réseau est plus simple et moins coûteuse que pour un cablô-opérateur. En effet, dans certaines infrastructures, tout le trafic entre utilisateurs doit passer par un routeur d'accès central ; dans ce cas le trafic entre utilisateurs peut conduire à une congestion de ce routeur d'accès.

Il y a autant de particularités dans l'infrastructure des FAI qu'il y a de FAI ; la décision de localiser le trafic peut donc grandement varier d'un FAI à l'autre. Notre travail montre que si l'infrastructure d'un FAI permet de localiser le trafic, alors le FAI peut largement réduire son trafic interFAI sans pénaliser ses utilisateurs (et même augmenter leur performance en cas de congestion) .

2.4 Discussion

Les travaux présentés dans ce chapitre ont été effectués entre 2005 et 2009, c'est-à-dire qu'ils ont débuté environ trois ans après la mise à disposition publique sur SourceForge du client BitTorrent *mainline* en version 3.0 (le premier client à implémenter les politiques de sélection de pièces et de pairs actuelles).

Début 2005, la dynamique de BitTorrent était mal comprise et BitTorrent inspirait une grande défiance ; une partie de la communauté essayait de montrer que BitTorrent était peu efficace et qu'il était possible de le remplacer par d'autres solutions très supérieures. L'histoire leur a donné tort. Il est plus facile de conspuer que de trouver des arguments pour défendre une idée nouvelle ; le groupe donnant de la force lorsque la rigueur fait défaut.

Pour comprendre cette défiance, il faut faire un rapide rappel de l'histoire de la distribution de contenus dans Internet. Dès la fin des années 1980, le problème de la distribution de contenus à grande échelle devint une préoccupation majeure pour la communauté (des chercheurs). En 1988, Steve Deering [39] pose les bases du routage multipoint. L'objectif est de permettre la distribution à grande échelle de contenus sans coût spécifique pour la source. Pendant dix années le multipoint a été l'un des principaux sujets de recherche ; cependant, à la fin des années 1990, il a fallu admettre que le multipoint n'était pas une solution viable pour les utilisateurs finaux [42].

Certains spécialistes du multipoint ont alors eu l'idée des réseaux de distribution de contenus, ou CDN pour *Content Delivery Network*. Ces réseaux reposent sur une infrastructure dédiée dont le but est de fournir les contenus d'un serveur (de cette architecture) proche de l'utilisateur. Ainsi, la congestion dans le réseau et les délais de propagation entre la source et le client ne sont plus un facteur limitant. Cependant, les CDN représentent une infrastructure coûteuse qu'un nombre limité de fournisseurs de contenus peuvent s'offrir. Pour contourner ce problème du coût de l'infrastructure, l'idée d'utiliser les ressources des utilisateurs dans une architecture pair-à-pair a germé. En 1999, la première application de distribution de contenus pair-à-pair, Napster, fait son apparition. Napster est plus proche d'un système client-serveur distribué que d'un système pair-à-pair tel qu'on le connaît aujourd'hui, mais il a le mérite de montrer pour la première fois que l'exploitation de la capacité d'envoi des pairs est faisable en pratique et à grande échelle. Deux ans plus tard, le premier prototype de BitTorrent fait sont apparition. Fin 2004, soit trois ans après son introduction, BitTorrent représente plus de 50% du trafic d'Internet. Jamais un protocole n'avait connu un succès aussi fulgurant.

Cependant, BitTorrent n'a pas été inventé par un membre du sérail ; Bram Cohen est le jeune développeur, sans diplôme de docteur ni même diplôme de master, qui a conçu seul BitTorrent tel qu'il existe aujourd'hui. Il s'agit d'un cas unique dans l'histoire d'Internet, très différent d'autres succès comme Google, Facebook, ou twitter. En effet, ces autres succès sont des services Web, alors que BitTorrent est un protocole réseau ; ils ont reçu des sommes astronomiques pour être tels qu'on les connaît aujourd'hui, BitTorrent n'a eu besoin d'aucun support financier pour devenir populaire ; ils utilisent de nombreux travaux de recherche (Google en produit même beaucoup) pour faire fonctionner leur plate-forme, BitTorrent est un protocole en grande partie totalement nouveau.

Il est incontestable que BitTorrent a d'abord laissé incrédule la communauté (par certains aspects, l'histoire de BitTorrent est proche de celle des turbo codes [49]). Avant les travaux de Yang *et al.* [89] en 2004, il était courant d'entendre des chercheurs affirmer que BitTorrent était moins efficace qu'une architecture client-serveur et que l'impression de robustesse à l'échelle n'était qu'un artefact (temporaire) de sa popularité ascendante (j'ai encore entendu de telles inepties en 2008 dans les couloirs de la conférence IPTPS).

Lorsque nous avons commencé à travailler sur BitTorrent, nous voulions également identifier des faiblesses et proposer des solutions. Seulement, tous nos travaux montraient au contraire la grande efficacité de ce protocole. Deux choix s'offraient alors à nous : s'engouffrer dans une faille identifiée à partir de scénarios artificiels et proposer une solution spécifique à ce cas particulier, ou comprendre la dynamique de BitTorrent dans la réalité et les raisons de cette efficacité. On a choisi la deuxième possibilité.

Nos travaux ont permis deux avancées importantes. Premièrement, on a montré que les politiques de sélection de pièces et de pairs étaient très efficaces en pratique, mais on a surtout montré l'importance du bon dimensionnement de la source initiale pour l'efficacité du torrent

et l'incitation au partage. Ces résultats fournissent l'explication de problèmes d'efficacité qui peuvent être perçus par les utilisateurs : ces problèmes viennent du mauvais dimensionnement de la source initiale. Deuxièmement, nos travaux ont défini un nouvel état de l'art dans l'évaluation des protocoles pair-à-pair. On a été les premiers à faire des mesures exhaustives sur de vrais torrents [64] et des expérimentations contrôlées à grande échelle [62][27].

Ces travaux ont été largement cités (au 1 août 2011, source Google Scholar) :

- 220 fois pour « *Rarest First and Choke Algorithms Are Enough* » publié à IMC'2006 [64] et 72 fois pour le rapport technique qui précéda cette publication « *Understanding BitTorrent : An Experimental Perspective* » [63] ;
- 151 fois pour « *Clustering and Sharing Incentives in BitTorrent Systems* » publié à SIGMETRICS'2007 [62] ;
- 36 fois pour « *Pushing BitTorrent Locality to the Limit* » publié dans Computer Networks en 2011 [27].

Le client BitTorrent que l'on a instrumenté [6] et qui est utilisé dans toutes les études de ce chapitre a été téléchargé 645 fois (au 1 août 2011) sur le site gforge d'Inria. Cependant, même si l'on sait (d'après des discussions durant les conférences) que beaucoup de chercheurs ont utilisé ce client pour comprendre le fonctionnement de BitTorrent, il n'y a pas à notre connaissance de publication (autre que les nôtres) utilisant ce client. Pourtant, on a mis à disposition publiquement ce client dès août 2006 et il s'agit à notre connaissance du seul client BitTorrent disponible publiquement et entièrement instrumenté.

Chapitre 3

Et la vie privée dans Internet ?

3.1 Introduction

Internet a révolutionné la manière de communiquer dans notre société sur une échelle de temps incroyablement courte. En particulier, le Web participatif (appelé parfois Web 2.0), avec ses applications phares comme Facebook ou twitter est très récent ; Facebook a été ouvert au grand public seulement fin 2006. On n'a donc qu'un recul très limité sur l'impact de ces technologies sur la vie privée de ses utilisateurs.

Il y a cependant une prise de conscience sur le danger potentiel, à court terme, de laisser un grand nombre de données personnelles publiquement accessibles. Ces dangers sont classiquement médiatisés comme étant de deux ordres. Il y a le risque de criminalité ciblée : des pédophiles traquant des enfants, des voleurs identifiant les périodes de vacances des occupants de maisons à cambrioler, etc. Il y a également le risque de préjudice : une entreprise licenciant un employé à cause de photos disponibles sur Internet, un adolescent bafoué suite à des vidéos publiées, etc.

À long terme, la production sur des années, et quotidienne pour certains, de données personnelles stockées et référencées sur des réseaux sociaux sera sans aucun doute une source majeure d'atteinte à la vie privée. En effet, dans notre société médiatique où l'exposition d'un fait à plus d'importance que son explication, il sera difficile de justifier ou de replacer dans son contexte un propos ou une image produit des années, voire des dizaines d'années, en arrière.

Ainsi, le risque d'atteinte à la vie privée est généralement vulgarisé comme étant la cause de deux problèmes distincts : une inconscience coupable des internautes qui laissent des informations personnelles et une permissivité assumée des services Web (principalement les réseaux sociaux) conduisant à la visibilité publique du plus grand nombre d'informations. L'effet pervers de cette vision manichéenne est qu'elle laisse penser qu'il suffit que les internautes soient plus prudents et que les réseaux sociaux soient plus rigoureux dans l'accès aux informations pour que le risque d'atteinte à la vie privée soit minimisé.

Dans ce chapitre, on va montrer que le simple fait d'utiliser Internet, indépendamment de l'utilisation d'un réseau social, peut considérablement exposer la vie privée des internautes. En particulier, l'utilisation d'un protocole pair-à-pair ou d'un protocole de voix sur IP peut révéler

beaucoup d'informations sensibles. Par conséquent, les risques d'atteinte à la vie privée ne sont pas limités à l'utilisation de réseaux sociaux (même s'il s'agit d'un facteur aggravant), mais ils sont liés à l'architecture actuelle d'Internet.

Pour comprendre cette notion d'atteinte à la vie privée dans le contexte d'Internet, il faut définir la notion d'identité. Il existe trois types d'identités qui peuvent être exploitées pour porter atteinte à la vie privée d'un internaute.

Notion d'identité

L'identité sociale d'une personne, c'est-à-dire l'identité utilisée dans les relations sociales, est classiquement déterminée par son nom et son prénom, et parfois, lorsqu'il y a ambiguïté, par un lieu de résidence, une date ou un lieu de naissance, un employeur, un ami, etc. Cette identité n'offre pas de garantie d'unicité (contrairement à un numéro INSEE), mais permet malgré tout d'identifier avec une grande fiabilité un individu.

L'identité applicative est liée à une application ou ensemble d'applications spécifiques (par exemple, Skype, Facebook, Amazon, les applications Google, etc.) Elle peut être temporaire, c'est le cas d'identifiants contenus dans des *cookies*, ou permanente, c'est le cas d'un nom d'utilisateur de Skype. Alors que certains identifiants ne sont qu'une chaîne aléatoire de caractères, d'autres comme un nom d'utilisateur peuvent être très explicites sur l'identité sociale de la personne. C'est typiquement le cas lorsque l'identifiant est de la forme `nom.prenom`. Il existe donc, dans certains cas, une passerelle entre identité applicative et identité sociale.

L'identité réseau est l'adresse IP utilisée par la machine de la personne. Cette identité est associée à l'ensemble de l'activité d'une personne sur le réseau. En effet, toutes les informations qui sortent ou qui entrent dans la machine sont encapsulées dans des paquets IP contenant l'adresse réseau de cette machine. Il s'agit donc d'une information très sensible du point de vue de la protection de la vie privée. Il y a cependant de nombreuses complications pour qu'un attaquant puisse exploiter ce type d'informations. L'adresse IP peut être dynamique, c'est-à-dire qu'elle change périodiquement, ou elle peut être celle d'un NAT ou d'une passerelle IPv6/IPv4 et par conséquent être partagée par plusieurs utilisateurs. De plus, il est difficile d'observer l'adresse IP d'une personne en particulier. Il n'y a en effet que deux moyens d'observer des paquets IP : soit l'attaquant capture sur le lien d'accès de la personne tous les paquets IP qui passent, ce qui est difficile à réaliser en pratique puisqu'il faut un accès physique à un endroit spécifique de l'infrastructure ; soit l'attaquant est la destination des paquets IP, ce qui limite fortement l'exploitation de cette attaque puisqu'il est difficile pour un attaquant d'obliger un internaute à lui envoyer des paquets. Cependant, les applications pair-à-pair — en particulier BitTorrent et Skype qui représentent autour de 20% du trafic d'Internet et des centaines de millions d'utilisateurs — permettent à n'importe qui d'avoir une connexion directe (c'est-à-dire pair-à-pair) avec des internautes. Nous montrons dans ce chapitre qu'il est possible d'exploiter ces applications pair-à-pair pour obtenir l'identité réseau d'un grand nombre d'internautes et

pour faire le lien entre identité sociale et identité réseau.

Attaques exploitant l'identité

Il existe deux grands types d'attaques exploitant ces notions d'identité : *le profilage* consiste à associer un ensemble d'activités à une identité ; *l'association* consiste à corréler différents profilages en se basant sur une identité commune.

On va discuter dans la suite de quelques exemples de profilages et d'associations qui sont officiellement menés par de grandes sociétés d'Internet.

Profilage

Un site Web populaire qui propose un enregistrement à ses utilisateurs (par exemple, Facebook, Google, Amazon, twitter, etc.) peut associer l'identité applicative d'une personne à l'intégralité de ses activités sur ce site : les messages envoyés ou lus, les photos postées, les applications utilisées, les documents produits, les amis sélectionnés, etc. Certains sites utilisent ce profilage pour faire des recommandations, comme Amazon, ou fournir des publicités ciblées, comme Google.

Cependant, pour la majorité des utilisateurs, ce profilage est acceptable. En effet, les utilisateurs ont confiance dans ces sites mondialement utilisés. Il est en effet raisonnable de croire qu'il est peu probable que ces sites, dont la viabilité repose sur leur popularité, prennent le risque de porter atteinte volontairement à la vie privée de leurs utilisateurs.

Mais, il faut être conscient que la notion d'atteinte à la vie privée dépend fortement de la législation du pays où sont stockées les données. Ce qui est illégal dans un pays peut être parfaitement légal dans un autre. En particulier, la législation d'un pays est démunie face à une société localisée et stockant les données de ses utilisateurs dans un autre pays, ce qui est le cas de la quasi-intégralité des sociétés non françaises (Google, Facebook, Skype, Amazon, etc.) offrant un service ou un portail en français.

Association

Une attaque par association peut exploiter la connaissance qu'une identité sociale et une identité réseau appartiennent à la même personne. Cette attaque est très dangereuse pour la protection de la vie privée puisqu'elle permet de savoir qui fait quoi sur le réseau.

En théorie, seul le FAI d'une personne connaît le lien entre l'adresse IP utilisée par cette personne et son identité sociale. Cependant, tous les sites Web populaires qui demandent un enregistrement des utilisateurs peuvent faire ce lien. Il est, en effet, probable que la personne se connecte directement au site, sans passer par un proxy ou un système garantissant l'anonymat réseau comme Tor [41]. De plus, il y a une forte incitation à utiliser son identité sociale lors de l'enregistrement auprès d'un réseau social (pour pouvoir être facilement identifié par des amis),

d'un site Web de e-commerce (pour facilement réceptionner les livraisons), ou d'un service de voix sur IP (pour être facilement contacté).

Mais si ces sites Web peuvent faire le lien entre identité sociale et identité réseau, ils n'ont pas accès au trafic réseau de leurs utilisateurs, autre que le trafic destiné à ces sites. Cependant, ils peuvent faire des associations par d'autres moyens. Par exemple, les sociétés offrant de multiples services, comme Google, peuvent associer des profilages sur chacun de leurs services. Parfois, les internautes peuvent être profilés par un service sans même en avoir conscience. C'est notamment le cas avec Google Analytics qui fournit un service de statistiques de consultation de sites Web. À chaque fois qu'un internaute consulte un site qui utilise Google Analytics, son identification (adresse IP, identifiant unique contenu dans un cookie) et la page Web consultée sont stockées par Google. De plus, rien n'indique qu'une page Web utilise Google Analytics ; pour le découvrir il faut regarder le code source de la page et identifier le code JavaScript utilisé par Google, ce qui est hors de portée d'un grand nombre d'internautes.

Quels risques supplémentaires d'atteinte à la vie privée ?

On vient de voir que d'un côté les utilisateurs, même s'ils publient un grand nombre d'informations personnelles, ne sont pas responsables des données collectées à leur insu. Et d'un autre côté, les services Web ont pour cœur de métier la collecte, l'exploitation et parfois la mise à disposition (dans le cas des réseaux sociaux) d'informations personnelles.

Par conséquent, le risque d'atteinte à la vie privée se situe dans l'exploitation que les sociétés opérant ces services Web peuvent faire des données collectées. Les conditions d'utilisations et les chartes de respect de la vie privée de ces sociétés sont souvent obscures, extrêmement permissives et parfois contradictoires. Malgré tout, les internautes continuent d'utiliser ces services parce que le service rendu est important, il n'existe pas d'alternative offrant un meilleur niveau de protection de la vie privée, et l'intérêt des sociétés n'est pas de porter préjudice à leurs utilisateurs.

Cependant, le risque d'atteinte à la vie privée ne se limite pas au scénario d'une société qui collecte des données personnelles ; il en existe d'autres dont un qui est largement sous-estimé : le scénario d'*un individu sans moyens dédiés* qui cherche à collecter des informations pour faire des profilages et des associations — nous appellerons cet individu un *attaquant* dans la suite. Ce scénario est particulièrement préoccupant puisque contrairement au cas d'une société, un attaquant peut être en dehors de tout contrôle et il peut avoir une incitation à porter préjudice.

Le but de ce chapitre est de montrer qu'un tel attaquant peut collecter sur les internautes des informations très sensibles qui étaient réputées difficiles, voire impossibles, à obtenir. En particulier, nos contributions s'articulent autour de la découverte de l'adresse IP de quatre catégories d'internautes spécifiques [28, 29, 31, 32].

– Dans la section 3.3.1, on présente la première catégorie qui est celle des internautes qui téléchargent le plus de contenus en utilisant le protocole BitTorrent. La difficulté de ce

travail est que pour identifier ces internautes, il faut avoir une vue exhaustive et donc mesurer l'intégralité des internautes utilisant BitTorrent.
- Dans la section 3.3.2, on présente la deuxième catégorie qui est celle des sources initiales de contenus utilisant BitTorrent. La difficulté ici est que, comme les sources initiales ne s'annoncent pas comme des sources, il est difficile de faire la distinction entre un pair, une source et une source initiale.
- Dans la section 3.3.3, on présente la troisième catégorie qui est celle des utilisateurs de Skype. Chaque client Skype échange des messages chiffrés avec des dizaines d'autres clients. Identifier l'adresse IP d'un client spécifique est un réel challenge.
- Dans la section 3.3.4, on présente la dernière catégorie qui est celle des utilisateurs de Tor qui est un réseau garantissant l'anonymat (au niveau IP) de ses utilisateurs. Il est par conséquent, par construction de Tor, très difficile de retrouver l'adresse IP d'un utilisateur de ce réseau.

En résumé, nos travaux montrent qu'il est possible pour un individu sans moyens dédiés de collecter des informations privées sur une grande population d'internautes, alors que l'on pensait que des collectes de ce type étaient uniquement réalisables par de grandes sociétés d'Internet ou par des gouvernements.

Dans la suite, on couvre l'état de l'art dans la section 3.2, on présente nos contributions dans la section 3.3 et on discute nos résultats dans la section 3.4.

3.2 État de l'art

On présente dans la suite trois catégories de travaux liés aux résultats de ce chapitre : les mesures à grande échelle et la caractérisation d'utilisateurs de BitTorrent, les mesures de mobilité des personnes, les attaques sur l'anonymat des utilisateurs de Tor.

3.2.1 Mesures à grande échelle sur BitTorrent

Les deux premières études largement citées présentant des mesures de BitTorrent ont été publiées en 2004 [54] et 2005 [76]. Ces études ont permis de gros progrès dans la compréhension de BitTorrent. Cependant, elles proposent soit une vue très partielle en focalisant sur un seul torrent [54], soit une exploration qui se limite à l'infrastructure mais qui ne descend pas à la granularité des pairs [76].

Les premières études sur des mesures à grande échelle de BitTorrent ont commencé à être publiées en 2009. Il existe quatre grandes catégories de mesures : les mesures collectant des adresses IP de pairs sur les trackers, les mesures collectant des adresses IP sur les DHT, les mesures collectant des statistiques agrégées sur les pairs et les mesures identifiant les sources initiales des contenus.

Collecte d'adresses IP sur les trackers

La majorité des travaux sur les mesures à grande échelle de BitTorrent concerne des collectes d'adresses IP de pairs sur les trackers. Siganos *et al.* [85] ont mesuré les 600 plus gros torrents trouvés sur *ThePirateBay* pendant 45 jours collectant ainsi 37 millions d'adresses IP ; ils ne donnent pas le nombre de pairs mesurés simultanément.

Piatek *et al.* [73] ont mesuré 55 523 torrents pendant un mois avec un cluster de machines collectant 12 millions d'adresses IP ; la fréquence de mesure des torrents et le nombre de pairs simultanément mesurés ne sont pas précisés. Cependant, ces auteurs ayant reporté des problèmes de bannissement en cas de mesures agressives [74], il est probable que la fréquence de mesures est faible. Le but de Piatek *et al.* est de motiver la nécessité d'avoir une nouvelle méthode d'incitation pour BitTorrent. Cependant, on remarque que comme l'ensemble des travaux dans ce domaine, nos résultats présentés en section 2.3.1 et en section 2.3.2 ne sont soit pas pris en compte soit mal cités. En effet, nos travaux montrent l'importance du dimensionnement de la source initiale dans l'efficacité et l'incitation au partage de BitTorrent. Utiliser des résultats de mesures de vrais torrents sans prendre en compte le dimensionnement de la source initiale invalide l'ensemble des conclusions sur la performance ou l'incitation au partage.

Zhang *et al.* [91] ont collecté 4,6 millions de fichiers `.torrent` pendant une période de 9 mois. Cependant, avec une infrastructure dédiée de 35 machines, ils n'ont collecté qu'une seule fois les adresses IP connectées à ces torrents sur une fenêtre de 12 heures, ce qui représente 5 millions d'adresses IP. L'objectif de cette étude est de caractériser les torrents de l'écosystème public de BitTorrent (la partie caractérisation des pairs constituant une petite partie des travaux) ; elle remplit parfaitement cet objectif et constitue la meilleure étude actuelle sur le sujet. Il faut également noter une deuxième étude des mêmes auteurs sur la caractérisation des torrents de l'écosystème privé [90].

Choffnes *et al.* ont mesuré une population de 10 000 pairs BitTorrent pendant une période de 1 mois avec un maximum de 3 000 utilisateurs simultanés ; ils n'ont cependant pas enregistré les contenus téléchargés par ces pairs. De plus, cette étude, beaucoup plus petite en terme de nombre de pairs mesurés, est très différente dans l'approche. En effet, même si la technique exacte de mesures n'est pas explicitée, il est probable que les mesures sont obtenues à partir de clients instrumentés et non à partir de mesures sur un tracker.

Les résultats présentés dans ce chapitre se différencient par l'échelle des données collectées, la performance de la collecte et l'objectif. On a collecté 148 millions d'adresses IP sur 103 jours pour 1,2 million de contenus, ce qui constitue à notre connaissance la plus large campagne de mesures de BitTorrent au moment de l'écriture de cette thèse. De plus, notre étude est la seule à présenter une méthode de mesures à grande échelle quasiment en temps réel (on peut mesurer plus de 90% des pairs dans 700 000 torrents en 30 minutes) sans infrastructure dédiée, c'est à dire d'une seule machine standard. Notre étude est la première à montrer qu'il est possible à un attaquant sans moyens spécifiques de collecter la quasi-intégralité des téléchargements des pairs

de BitTorrent en temps réel.

Collecte d'adresses IP sur les DHT

En août 2009, le tracker de *ThePirateBay* a été arrêté suite à une décision de justice. Ce tracker était le plus populaire (en nombre de pairs et de torrents), un ordre de grandeur (avec plus de 10 millions de pairs et 1 million de torrents simultanés) supérieur au deuxième tracker le plus populaire [91]. La décision prise par les administrateurs de *ThePirateBay* fut d'utiliser exclusivement les trackers distribués disponibles sur les DHT des différents clients BitTorrent.

Wolchok *et al.* [87] ont mesuré en utilisant la DHT de Vuze 1,5 million de torrents pendant 16 jours ce qui leur a permis de collecter 7,9 millions d'adresses IP.

Cette étude a été partiellement inspirée par les résultats présentés dans ce chapitre (et en particulier par l'étude [29]) ; l'objectif était de vérifier si des mesures à grande échelle étaient toujours possibles aujourd'hui sur les DHT utilisées par BitTorrent. Même si cette étude est très inférieure en terme d'échelle à nos résultats, elle montre qu'il est, en effet, toujours possible de faire de telles mesures.

Collecte de statistiques agrégées

Dan *et al.* [38] ont mesuré 2,4 millions de torrents représentant 37 millions de pairs. Cependant, ils utilisent une méthode exploitant des requêtes *scrape*, c'est-à-dire qu'ils n'obtiennent que le nombre de pairs par torrent, mais pas leur adresse IP. Par conséquent, cette étude est différente des autres études citées par les données collectées qui sont beaucoup plus faciles à obtenir. De plus, toutes les études se limitant à des requêtes *scrape* ne peuvent pas être considérées comme des preuves d'atteinte à la vie privée, puisque qu'il n'y a aucune collection d'informations liées à l'identité des utilisateurs.

Identification des sources initiales

Il n'y a à notre connaissance que deux travaux qui abordent le problème de l'identification des sources initiales : un travail de Cuevas *et al.* [36] et notre étude [28, 29]. Ces travaux ont été faits en parallèle.

Cependant, ces deux travaux sont très différents dans la technique utilisée pour identifier les sources initiales et dans les résultats. Cuevas *et al.* [36] présentent une technique d'identification des sources initiales qui consiste à se connecter le plus rapidement possible aux nouveaux torrents et à identifier la seule source du torrent comme la source initiale. Cependant, on montre que cette technique est inefficace puisque la majorité des sources initiales utilisent une optimisation appelée *super seeding* ; avec cette optimisation, une source initiale s'annonce comme un pair sans pièce, cette source est donc indétectable par la méthode utilisée par Cuevas *et al.* L'objectif du *super seeding* est d'optimiser les ressources de la source. En effet, lorsqu'un pair se connecte à la source initiale, cette dernière ne lui annonce et ne lui sert qu'une seule pièce. La source va ensuite

écouter les messages HAVE des autres pairs qui signalent la réception d'une pièce. Si la source reçoit des messages HAVE pour la pièce qu'elle a envoyée, cela signifie que le pair contribue et que donc la source peut lui envoyer d'autres pièces. Si la source ne reçoit aucun message HAVE pour la pièce envoyée, alors il est probable que le pair est un *free rider*. La source initiale ne lui enverra donc plus rien.

Notre méthode d'identification des sources initiales fonctionne dans le cas des sources utilisant le *super seeding* et également dans le cas où il y a déjà plusieurs sources dans le torrent. On montre que notre méthode permet l'identification des sources initiales dans 70% des cas.

L'objectif de Cuevas *et al.* est d'expliquer la motivation des sources initiales par un profit. Notre objectif est de montrer que l'on peut identifier avec une grande fiabilité une grosse proportion des sources initiales et en étudier la performance. En résumé, notre méthode d'identification est supérieure est nos objectifs sont orthogonaux.

3.2.2 Mesures de mobilité

Les travaux que l'on présente dans la suite sont liés à des attaques sur la mobilité des personnes ; une telle attaque est la collecte d'informations sur la mobilité d'une personne sans son consentement.

Guha *et al.* [48] montrent qu'en collectant périodiquement les adresses IP des utilisateurs d'un service de DNS dynamique, il est possible de suivre la mobilité de ces utilisateurs. Il y a cependant deux limitations à ce travail. Premièrement, cette attaque est limitée aux utilisateurs d'un service de DNS dynamique, c'est-à-dire à quelques millions d'utilisateurs. Deuxièmement, cette attaque permet de lier une mobilité à une identité sociale uniquement lorsque le choix du nom du domaine indique cette identité.

Le travail que l'on présente dans ce chapitre utilise Skype qui est le client de voix sur IP le plus populaire avec plus de 500 millions d'utilisateurs. Par conséquent, l'attaque que l'on décrit couvre une fraction importante des internautes. De plus, Skype permet de renseigner, en plus du nom d'utilisateur, un nom, un prénom, un pays de résidence, etc. On montre que 88% des utilisateurs de Skype fournissent leur nom et 82% fournissent soit leur âge, leur sexe, l'URL de leur page Web, leur pays de résidence, ou leur langue.

Le projet Carmen Sandiego [21] présente une exploitation du réseau mobile GSM pour suivre la mobilité des utilisateurs de téléphones portables. Les auteurs utilisent le service de présentation du numéro pour lier un numéro de téléphone portable à une identité sociale. Ensuite, en accédant au *Home Location Register* (HLR), les auteurs montrent qu'un attaquant peut accéder à l'identifiant du *Mobile Switching Center* (MSC) pour un numéro de téléphone donné. L'identifiant du MSC peut donner des informations sur sa localisation, donc sur la localisation de la personne passant par ce MSC. Cependant, comme il n'existe pas de convention sur la manière dont un opérateur doit nommer ses MSC, l'identifiant ne donne pas toujours une information pertinente sur sa localisation.

Même si notre objectif est similaire, notre méthode est fondamentalement différente. De plus, l'identité réseau est très différente d'un numéro de téléphone. En effet, un numéro de téléphone est essentiellement utilisé pour téléphoner. Par contre, un identifiant réseau est utilisé pour un spectre beaucoup plus large d'activités : achats sur Internet, téléchargements, consultation de sites Web, voix sur IP, activités sur des réseaux sociaux, etc. Ainsi, même s'il y a moins d'utilisateurs de Skype que d'utilisateurs de téléphones GSM, le fait que l'on puisse, à partir de l'identifiant réseau, corréler différentes activités sur Internet rend notre attaque encore plus sévère qu'une attaque de la mobilité seule

3.2.3 Attaques sur l'anonymat des utilisateurs de Tor

Onion routing [81] et son successeur Tor [41] sont des réseaux pair-à-pair garantissant un haut niveau d'anonymat sur Internet en utilisant le concept développé par Chaum [33] de routage qui ne peut pas être tracé.

Tor est le sujet d'actives recherches sur ses vulnérabilités et les moyens de les diminuer [71].

FortConsult [45] décrit deux attaques spécifiques aux applications Web utilisant de l'injection de contenus. La première attaque consiste à injecter un contenu Flash dans le trafic Web pour que la cible de l'attaque se connecte, à l'extérieur de Tor, à un serveur qui est contrôlé par l'attaquant ; le serveur connaît donc l'adresse IP de la cible. L'attaquant utilise également un cookie pour permettre au serveur de faire le lien entre l'adresse IP et la requête Web initiale. La deuxième attaque consiste à injecter du code JavaScript pour envoyer l'adresse IP de la cible à l'attaquant. Les auteurs rapportent que la première attaque est efficace, mais que la deuxième attaque rapporte souvent des adresses IP privées qui ne sont donc pas exploitables.

Abbott *et al.* [18] décrivent une attaque utilisant du JavaScript et des balises HTML *meta refresh* pour injecter au nœud de sortie (*exit node*) une signature temporelle dans les flux qui peut être détectée au nœud d'entrée (*entry node*). Pour réaliser cette attaque, il faut que l'attaquant contrôle le nœud d'entrée et le nœud de sortie par lesquels passe le flux de données de la cible. Cette attaque est donc difficile à mettre en œuvre en pratique.

McCoy *et al.* [69] étudient l'utilisation de Tor (mais pas l'identification des utilisateurs). Ils montrent que le trafic BitTorrent représente 40% du trafic de Tor alors qu'il ne représente que 3% des flux.

Notre travail est, à notre connaissance, le premier à décrire une attaque spécifique au pair-à-pair, à la valider en pratique sur une population significative de 10 000 utilisateurs et à corréler identification et usage. De plus, on montre qu'une attaque sur un flux de données pair-à-pair permet d'associer l'adresse IP de la cible à de nombreux autres flux, autres que pair-à-pair, provenant de cette cible.

3.3 Attaques sur la vie privée

On présente, dans la suite, des profilages et des attaques par association pour quelques unes des applications les plus populaires actuellement : BitTorrent, Skype et Tor.

3.3.1 Identification des utilisateurs de BitTorrent

BitTorrent est le protocole pair-à-pair le plus populaire ; il est utilisé quotidiennement par des dizaines de millions d'internautes. Cependant, le comportement de ses utilisateurs, et en particulier des plus actifs, est mal compris. La principale raison est la difficulté de mesurer BitTorrent à grande échelle.

Pour expliquer cette difficulté, on va introduire les trois étapes nécessaires pour qu'un pair se connecte à un torrent ; tous les travaux de mesures utilisent un processus similaire à celui d'un pair.

La première étape consiste à chercher s'il existe un torrent pour un contenu spécifique que le pair souhaite télécharger. Pour cela, un pair utilise en général un site Web qui indexe les torrents. Il peut cependant utiliser d'autres moyens comme des forums, des canaux IRC, des blogs, etc. Le résultat de cette recherche est un ensemble d'informations sur le torrent (date de création, utilisateur qui a déclaré le torrent sur le site Web, commentaires, etc.) et un lien qui peut être de deux types : un lien vers un fichier `.torrent` ou un lien *magnet*.

La deuxième étape consiste à utiliser le lien fourni lors de la recherche pour récupérer les informations nécessaires à la connexion au torrent. Ces informations sont les mêmes quel que soit le type du lien : un ensemble d'informations sur le contenu à télécharger (taille du contenu, taille des pièces, nombre de pièces, SHA-1 de chaque pièce) et un point de rendez-vous qui puisse fournir une liste de pairs déjà dans le torrent. Dans le cas d'un lien vers un fichier `.torrent`, le pair téléchargera un petit fichier, appelé fichier `.torrent`, contenant l'ensemble de ces informations. Cependant, comme la charge pour servir ces fichiers est importante, la notion de lien *magnet* a été introduite. Un lien *magnet* contient directement l'adresse d'un point de rendez-vous. Le pair, à partir de cette adresse, va récupérer une liste de pairs déjà présents dans le torrent (comme on le verra dans la troisième étape) et va directement télécharger le fichier `.torrent` de ces pairs. Ainsi, avec un lien *magnet*, la charge de distribution du fichier `.torrent` est répartie sur l'ensemble des pairs.

La troisième étape consiste à se connecter au point de rendez-vous pour récupérer une liste de pairs déjà dans le torrent. Il existe deux types de points de rendez-vous : les trackers et les DHT. Un tracker est un serveur qui maintient la liste des pairs dans un torrent. À chaque fois qu'un nouveau pair se connecte, le tracker lui retourne un sous-ensemble aléatoire des pairs déjà présents (typiquement entre 50 et 200). Une fois cette liste de pairs récupérée, le téléchargement du contenu peut commencer. Une DHT est une implémentation décentralisée d'un tracker. Chaque pair de la DHT va être le tracker pour un sous-ensemble de torrents. Par conséquent,

du point de vue d'un pair, demander une liste de pairs à une DHT ou à un tracker centralisé est équivalent.

Alors que dans le fonctionnement normal de BitTorrent, chaque pair connaît l'adresse IP de 50 à 100 voisins pour les torrents auxquels il participe, la communauté pensait très coûteux, voire même impossible, de connaître l'adresse IP de l'ensemble des utilisateurs de BitTorrent. En effet, les contre-mesures implémentées au niveau des sites Web et de trackers ralentissent considérablement la vitesse à laquelle un pair peut collecter des adresses IP sans être banni.

On présente une méthode qui permet de collecter quasiment en temps réel l'adresse IP de plus de 90% des utilisateurs de BitTorrent lorsqu'un tracker centralisé est utilisé. Cette méthode sans infrastructure dédiée (on n'utilise qu'un seul ordinateur standard) montre qu'il est possible pour un attaquant de collecter, sur des mois, la liste de tous les contenus téléchargés par les utilisateurs de BitTorrent. Ceci constitue un risque majeur de protection de la vie privée lors de l'utilisation de BitTorrent. Il est important de rappeler que BitTorrent est le meilleur candidat pour la distribution légale de contenus en pair-à-pair. Par conséquent, ce risque d'atteinte à la vie privée est à prendre en compte pour toutes utilisations légales de BitTorrent.

Dans la suite, nous présenterons notre méthodologie, puis un résumé de nos résultats. L'intégralité de nos résultats est disponible dans une publication LEET'10 [29] et un rapport technique [28].

Méthodologie

Du 13 mai au 24 août 2009, on a collecté 148 millions d'adresses IP pour 1,2 million de torrents uniques. La méthode que l'on a utilisée est la suivante.

Le 13 mai 2009, on a collecté sur les sites Web *mininova* et *ThePirateBay* l'ensemble des fichiers `.torrent` et des informations disponibles associées à ces torrents. Ces sites utilisent des conventions de nommage des pages Web qui permettent de lister l'ensemble des torrents disponibles. On a découvert 1 411 940 fichiers `.torrent` uniques sur *mininova* et 974 980 fichiers `.torrent` sur *ThePirateBay*. Le recouvrement entre les deux sites est de 227 620 fichiers `.torrent`. Cette collecte est une opération coûteuse en terme de bande passante à cause du grand nombre d'informations à collecter.

Toutes les 24 heures, on collecte tous les fichiers `.torrent` ajoutés à ces deux sites dans les 24 dernières heures. Cette opération est facile puisque ces sites fournissent une page qui recense les ajouts récents. De plus, étant donné le nombre limité de nouveaux fichiers `.torrent` ajoutés quotidiennement (de l'ordre de quelques milliers), cette opération est peu coûteuse et rapide à effectuer. Le but de cette opération est d'avoir des informations à jour sur les derniers torrents insérés durant toute la durée de notre expérimentation.

Les trackers supportent deux types de requêtes : *announce* et *scrape*. La requête *announce* est utilisée par un pair lorsqu'il veut joindre un torrent ; l'identifiant du torrent (appelé *infohash*) est spécifié dans la requête. Lorsqu'un tracker reçoit une telle requête, il retourne entre 50 à 200

couples (IP, port) où *IP* correspond à l'adresse IP d'un pair déjà dans le torrent et *port* au numéro du port sur lequel ce pair accepte les connexions. Le nombre de couples demandés est un paramètre optionnel de la requête. Cependant, chaque tracker est configuré avec un nombre maximal de pairs retournés par requête *announce*. Le nombre effectif de pairs retournés est donc le minimum entre ce que la requête demande et la configuration du tracker. Un pair peut envoyer au tracker une requête *announce* avec une option *stop* qui a pour effet de retirer ce pair de la liste des pairs maintenus par le tracker pour le torrent spécifié dans la requête.

La requête *scrape* est utilisée pour obtenir le nombre de sources et de *leechers* maintenus par le tracker pour le torrent spécifié dans la requête. C'est typiquement la requête utilisée par les clients BitTorrent qui affichent cette information pour chaque torrent. Lorsque la requête *scrape* ne contient pas d'identifiant de torrent, le tracker va retourner l'ensemble des identifiants des torrents déclarés avec le nombre de sources et de *leechers* actuellement dans chaque torrent. On appelle dans la suite cette variante de la requête *scrape* une requête *scrape-all*.

Toutes les 24 heures, on effectue une requête *scrape-all* sur les trackers de *ThePirateBay*. Le but de cette requête est d'obtenir, directement des trackers, les identifiants des torrents. En effet, cet identifiant (appelé *infohash*) est utilisé pour toutes les communications BitTorrent avec le tracker et les pairs. Cependant, il doit être calculé à partir du fichier `.torrent` ; c'est le résultat d'un SHA-1 calculé essentiellement sur le SHA-1 de l'ensemble des pièces du contenu, c'est par conséquent un identifiant unique du torrent. Obtenir cet identifiant directement du tracker, avec une requête *scrape-all* économise beaucoup de calculs lors des mesures. Ensuite, pour savoir à quel contenu correspond un identifiant, il suffit de le comparer à la base de fichiers `.torrent` collectés.

Chaque requête *scrape-all* permet de collecter environ 2 millions d'identifiants. On ne garde que les identifiants pour les torrents actifs, c'est-à-dire les torrents avec au moins une source et un *leecher* ; il reste entre 500 000 et 750 000 identifiants pour les torrents actifs. Le tracker de *ThePirateBay* était le plus grand tracker lors de notre travail, d'un ordre de grandeur supérieur (en nombre de pairs et de torrents) au deuxième tracker le plus populaire [91]. Pour cette raison, nous nous sommes limités à *ThePirateBay*.

Finalement, toutes les deux heures, on effectue des requêtes *announce* pour chaque identifiant de torrent collecté précédemment avec la requête *scrape-all* jusqu'à obtenir 90% des pairs présents dans le torrent. Comme chaque requête demande 200 pairs, on obtient 100% des pairs pour tous les torrents inférieurs à 200 pairs. Il faut moins d'une seconde pour obtenir 90% des pairs du plus grand torrent et 30 minutes pour obtenir les pairs pour l'ensemble des torrents. On appelle, dans la suite, cette mesure effectuée toutes les 2 heures sur l'ensemble des torrents un *instantané*.

Sur le premier instantané, on a collecté 1 870 662 identifiants de torrents dont seulement 675 161 correspondaient à des torrents actifs. Le tracker annonçait 12 124 600 pairs pour ces torrents actifs et on a réussi à collecter 11 263 752 couples (IP,port) soit 93% de l'ensemble des

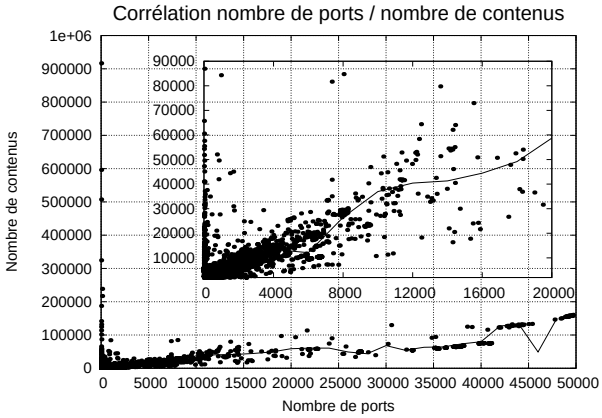

FIGURE 3.1 – Corrélation entre le nombre de ports et le nombre de torrents pour les 10 000 adresses IP les plus actives. *Chaque point représente une adresse IP. La ligne représente la moyenne sur les 148 millions d'adresses IP calculée sur un intervalle de* 2 000 *ports.*

pairs annoncés. Pour le dernier instantané, on a collecté 2 048 517 identifiants de torrents dont seulement 715 063 correspondaient à des torrents actifs. La tracker annonçait 13 845 696 pairs pour ces torrents et on a réussi à collecter 12 894 258 couples (IP,port) soit ici encore 93% de l'ensemble des pairs annoncés.

L'ensemble de ces mesures a été réalisé par une seule machine avec un processeur dual core et 32 GB de mémoire RAM.

Résultats

Sur les 103 jours de notre mesure, on a collecté 148 millions d'adresses IP uniques réparties sur 21 257 AS et téléchargeant 1 196 678 torrents. Ces téléchargements représentent 1,98 milliard de contenus et 3,6 exabytes échangés.

Ce travail montre que, sans infrastructure dédiée, un attaquant peut pendant des mois collecter la quasi-intégralité des contenus téléchargés par les utilisateurs de BitTorrent. Ceci constitue un problème majeur de protection de la vie privée.

Cependant, l'interprétation de cet énorme volume de données est difficile. En effet, la caractérisation du comportement individuel des internautes nécessite l'identification de ces derniers. Or l'adresse IP, ou le couple (IP,port), ne peut pas être utilisée pour identifier de manière fiable un pair. Pour le montrer, on va se concentrer sur les 10 000 adresses IP que l'on a vues participer au plus grand nombre de torrents sur la période de 103 jours (on les qualifie par la suite des 10 000 adresses IP les plus actives).

La figure 3.1 montre une corrélation linéaire, pour une grande partie des adresses IP les plus actives, entre le nombre de ports sur lesquels on a vu écouter une adresse IP et le nombre de torrents auxquels cette adresse a participé. En moyenne, chaque nouveau port correspond à 2 ou 3 nouveaux torrents. Par conséquent, ces adresses IP correspondent à des équipements réseaux

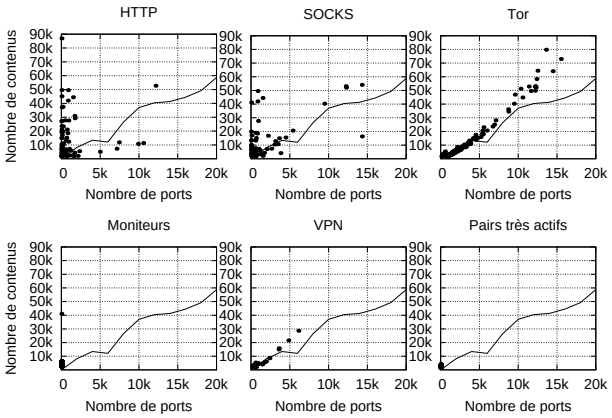

FIGURE 3.2 – Corrélation entre le nombre de ports par adresse IP et le nombre de torrents pour six catégories de pairs parmi les 10 000 adresses IP les plus actives. *Chaque point représente une adresse IP. La ligne représente la moyenne sur les 148 millions d'adresses IP calculée sur un intervalle de* 2 000 *ports.*

qui fédèrent, derrière une même adresse IP, un grand nombre d'utilisateurs ; c'est typiquement le cas des NAT et des passerelles IPv6/IPv4.

Cependant, on observe également sur la figure 3.1 qu'un certain nombre d'adresses IP dévient du comportement standard que l'on vient de décrire. Pour comprendre à quoi correspondent ces adresses IP, on identifie six catégories différentes.

Les passerelles HTTP ou SOCKS sont utilisées pour cacher son adresse IP des autres pairs d'un torrent. Il existe des listes publiques des adresses IP de ces passerelles. On a trouvé 81 passerelles HTTP et 62 SOCKS parmi les 10 000 adresses IP les plus actives.

Les Nœuds Tor (et plus particulièrement les *exit nodes*) sont également utilisés pour dissimuler son adresse IP. Pour identifier ces nœuds, on a effectué un *reverse DNS* sur les 10 000 adresses IP les plus populaires et on a filtré manuellement tous les noms contenant la chaîne de caractères `tor`. On a également utilisé des listes publiques de nœuds Tor. On a trouvé 174 nœuds Tor (*exit nodes*).

Les Moniteurs sont des infrastructures dédiées à la mesure des torrents. L'usage classique est soit académique (pour la recherche) soit dans un objectif de lutte contre le piratage. On a identifié deux AS contenant un grand nombre de pairs parmi les 10 000 plus actifs. Ces pairs avaient tous le même client BitTorrent dans la même version. On a également vérifié que ces pairs refusaient systématiquement de servir le contenu. On a trouvé 1 052 pairs dans cette catégorie appartement à seulement deux AS.

Les VPN sont des passerelles SOCKS qui ne sont pas publiques, par example *ipredator* ou *mullvad*. Pour les identifier, on a effectué un *reverse DNS* sur les 10 000 adresses IP les plus populaires et on a filtré manuellement tous les noms contenant une des chaînes de caractères : *itshidden*, *cyberghostvpn*, *peer2me*, *ipredate*, *mullvad* et *perfect-privacy*. On a trouvé 30 VPN.

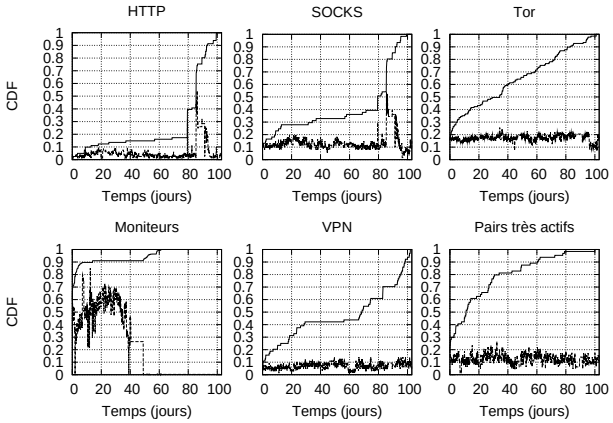

FIGURE 3.3 – Évolution dans le temps des 6 catégories de pairs parmi les 10 000 pairs les plus actifs. *Pour chaque instantané, la ligne en pointillés représente le pourcentage absolu d'adresses IP appartenant à chaque catégorie parmi les* 10 000 *adresses IP les plus populaires sur les 103 jours. La ligne pleine représente le pourcentage cumulatif normalisé par le nombre total de pairs dans la catégorie.*

Les pairs très actifs regroupent toutes les adresses IP qui ne correspondent à aucune des autres catégories. En particulier, on ne considère que les adresses IP qui ont utilisé moins de 10 ports différents. On a vérifié, de plus, que toutes ces adresses IP acceptaient de servir des pièces. On a trouvé 77 pairs très actifs.

La figure 3.2 montre le comportement des pairs de ces 6 catégories. On observe que le comportement des passerelles HTTP et SOCKS dévie significativement ; ces passerelles sont présentes dans beaucoup plus de torrents que la moyenne puisqu'elles sont utilisées par les entreprises luttant contre le piratage. En effet, on observe sur la figure 3.3 qu'au jour 50 tous les moniteurs disparaissent et qu'au jour 80 un grand nombre de passerelles HTTP et SOCKS apparaissent de manière coordonnée. La disparition des moniteurs correspond à un changement de stratégie de bannissement des trackers de *ThePirateBay*. Par conséquent, les entreprises luttant contre le piratage ont basculé leur infrastructure pour passer au travers de passerelles HTTP et SOCKS publiques.

Résumé des contributions

On a vu dans cette section qu'il était possible — sans aucune infrastructure dédiée — de profiler le comportement de la quasi-intégralité des utilisateurs de BitTorrent sur des mois, c'est-à-dire de faire le lien entre identité réseau et contenus téléchargés. Il s'agit d'un problème majeur de protection de la vie privée des utilisateurs de BitTorrent. Depuis la publication de nos résultats, les trackers ont changé leur comportement pour rendre plus difficile le profilage massif des pairs. Il est difficile de dire si nos travaux sont à l'origine de ce changement, mais étant donné la couverture médiatique internationale de ces travaux [7, 10, 11, 12, 9], c'est probable.

Actuellement, il y a également une utilisation croissante des DHT. Cependant, on estime que

le problème du profilage de l'ensemble des pairs utilisant BitTorrent est toujours aussi critique. En effet, en combinant les possibilités offertes par les trackers, les DHT et *peer exchange* (une fonctionnalité de découverte de pairs par *gossiping*) on est convaincu qu'il est toujours possible de profiler la quasi-intégralité des utilisateurs de BitTorrent en temps réel.

3.3.2 Identification des sources de contenus dans BitTorrent

À l'origine de chaque torrent, il y a une *source initiale*. Cette source est le premier pair à mettre à disposition un contenu dans un torrent ; c'est donc le premier pair à joindre le torrent. Ainsi, il semble facile d'identifier ces sources : il suffit de joindre un torrent rapidement après sa création et d'identifier l'unique source comme la source initiale. Il y a cependant deux complications. Premièrement, il peut être difficile de joindre rapidement un torrent. En effet, certains torrents (en général les torrents les plus intéressants, c'est-à-dire les torrents regroupant un grand nombre de pairs) démarrent par une phase privée durant laquelle seule une communauté restreinte d'utilisateurs peut joindre le torrent. Lorsque ces torrents sont déclarés sur les sites publics (comme *ThePirateBay*), ils peuvent déjà regrouper des centaines, voire des milliers de pairs. Deuxièmement, les clients BitTorrent modernes utilisent tous une optimisation appelée *super seeding* qui a pour conséquence que les sources initiales s'annoncent comme des pairs sans aucune pièce. En effet, cette optimisation, utilisée uniquement pour les sources initiales, a pour objectif de ne servir des pièces qu'aux *leechers* qui contribuent. Ainsi, lorsqu'un *leecher* contacte une source initiale, elle va annoncer et servir une seule pièce. Si la source observe que cette pièce est répliquée par le *leecher* (il suffit pour cela d'observer des messages HAVE envoyés par d'autres pairs pour cette pièce) elle lui donne une autre pièce et recommence la procédure d'observation. Si elle n'observe pas de réplication de la pièce, elle considère le *leecher* comme un *free rider* et elle ne lui donne plus rien.

On voit donc qu'en pratique l'identification des sources initiales est compliquée. Il est à noter que, à notre connaissance, notre travail, publié en 2010, est le premier à identifier avec succès les sources initiales dans BitTorrent. Cuevas *et al.* [36] ont proposé une méthode d'identification des sources initiales à la même période, mais beaucoup moins efficace que la notre (voir la section 3.2.1).

On présente, dans la suite, notre méthode pour identifier les sources initiales puis un résumé de nos résultats. L'intégralité de nos résultats est disponible dans une publication LEET'10 [29] et un rapport technique [28].

Méthodologie

Le site Web de *ThePirateBay* maintient une page qui recense les nouveaux torrents déclarés ; chaque torrent est déclaré par un utilisateur repéré par un identifiant unique. Du 8 juillet au 24 août 2009, on a interrogé cette page chaque minute et on s'est immédiatement connecté aux nouveaux torrents déclarés. Pour chacun de ces nouveaux torrents, on a enregistré l'identifiant

|Triviale|	|Identifiants|	|Triviale ∩ Identifiants|	Fiabilité
21 544	15 308	9 243	99,99%

TABLE 3.1 – Corrélation des résultats (en nombre de torrents pour lesquels une source initiale a été identifiée) obtenue par chacune des méthodes. *La première colonne donne le nombre de torrents pour lesquels une source initiale a été identifiée avec la méthode triviale et la deuxième colonne le donne pour la méthode des identifiants. La troisième colonne donne le nombre de torrents pour lesquels les deux méthodes ont identifié une source initiale. La dernière colonne donne le pourcentage de torrents (parmi ceux de la deuxième colonne) pour lesquels les deux méthodes ont identifié la même source initiale pour le même torrent.*

de l'utilisateur l'ayant déclaré et on est resté connecté pendant 24 heures collectant l'adresse IP des voisins et tous les messages échangés avec ces derniers.

Ainsi, sur 48 jours, on s'est connecté à 39 298 torrents déclarés par 6 210 utilisateurs et on a collecté l'adresse IP de 9 102 817 pairs.

On a défini deux méthodes d'identification des sources initiales. La première méthode, que l'on appelle la *méthode triviale*, consiste à identifier, pour un torrent donné, le seul pair présent dans ce torrent au moment où l'on se connecte comme étant la source initiale. S'il y a plus d'un pair lorsque l'on se connecte, la méthode échoue.

La deuxième méthode, que l'on appelle la *méthode des identifiants*, exploite les identifiants des utilisateurs ayant déclaré les contenus. Cette méthode part de l'hypothèse qu'il y a une seule personne, ou un petit groupe de personnes, qui joue le rôle de source initiale pour un utilisateur donné. Ainsi, pour tous les torrents déclarés par un même utilisateur, on dit que l'on a identifié la source initiale si les deux conditions suivantes sont vérifiées :

- il existe une adresse IP, ou un petit nombre d'adresses IP, qui apparaît pour tous les torrents déclarés par cet utilisateur ;
- cette adresse IP, ou ce petit nombre d'adresses IP, n'apparaît dans aucun autre torrent qui n'a pas été déclaré par cet utilisateur.

Parmi les 2 206 sources initiales identifiées par la méthode des identifiants, seulement 77 sources initiales participent à des torrents de plus d'un utilisateur et seulement 8 à plus de 3 utilisateurs. La méthode des identifiants est donc très fiable.

Résultats

On a présenté en section 3.3.2 deux méthodes indépendantes d'identification des sources initiales. La table 3.1 résume le nombre de torrents pour lesquels chacune de ces méthodes a identifié une source initiale. Il est intéressant de noter que les deux méthodes ont identifié une source initiale pour le même contenu pour seulement 9 243 contenus. Ces deux méthodes sont donc complémentaires. De plus, pour ces 9 243 contenus, les deux méthodes ont identifié la même source initiale dans 99,99% des cas. Comme on a pu identifier les mêmes sources initiales avec deux méthodes indépendantes, on peut avoir une grande confiance dans les résultats obtenus

|Triviale|	|Identifiants|	|Identifiants \ Triviale|
9 184	2 206	432

TABLE 3.2 – Nombre de sources initiales identifiées par chacune des méthodes. *La première colonne donne le nombre de sources initiales identifiées par la méthode triviale et la deuxième colonne par la méthode des identifiants. La troisième colonne donne le nombre de sources initiales identifiées par la méthode des identifiants qui n'ont pas été identifiées par la méthode triviale.*

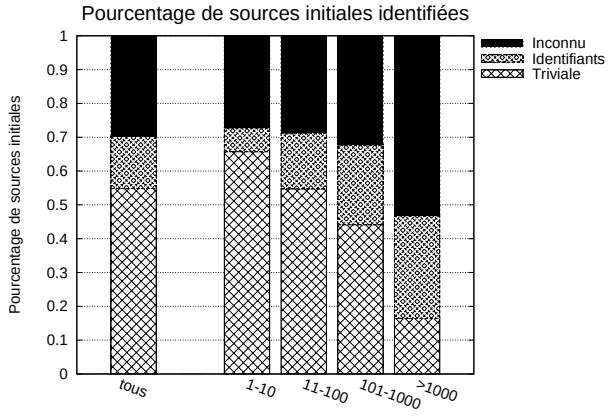

FIGURE 3.4 – Pourcentage de sources initiales identifiées. *Sur l'axe des x,* **tous** *est pour tous les contenus,* **a-b** *est pour les contenus avec entre a et b pairs après 24 heures,* **>1000** *est pour les contenus avec plus de 1000 pairs après 24 heures.* **Inconnu** *correspond au pourcentage de sources initiales non identifiées.*

par chacune de ces méthodes.

On voit dans la table 3.2 que la méthode triviale identifie beaucoup plus de sources initiales que la méthode des identifiants. De plus, la méthode des identifiants n'a trouvé que 432 sources initiales non identifiées par la méthode triviale, soit moins de 5%. On peut donc se poser la question de l'efficacité de la méthode des identifiants qui est plus complexe et plus difficile à mettre en œuvre que la méthode triviale.

La figure 3.4 donne une motivation forte pour la méthode des identifiants. En effet, on voit que cette méthode est plus efficace que la méthode triviale pour les grands torrents de plus de 1 000 pairs. Comme ces torrents sont ceux qui génèrent le plus de trafic (comme discuté en section 2.3.3 pour la figure 2.17), la méthode des identifiants est fondamentale pour capturer un comportement représentatif des sources initiales.

En résumé, nos deux méthodes d'identification des sources initiales permettent d'en identifier autour de 70%, voir figure 3.4. On va, dans la suite, caractériser ces sources initiales (de nombreux résultats supplémentaires sont disponibles dans le rapport technique [28]).

La figure 3.5 représente la popularité des noms des torrents servis par les sources initiales. On remarque que la popularité suit de près l'actualité. En effet, notre étude a été faite sur une période couvrant le décès le Michael Jackson et la sortie au cinéma d'un épisode de Harry Potter.

FIGURE 3.5 – Nuage de noms des contenus insérés par les sources initiales identifiées (figure générée par www.wordle.net sous licence CC 3.0). *On extrait les deux mots les plus significatifs des noms de contenus. La taille de la police est linéairement proportionnelle à la fréquence d'apparition d'un mot.*

On note également que d'après les noms de contenus, la majorité des contenus insérés semble piratée. Étant donné que l'on n'a pas fait une inspection par contenu, il faut rester prudent. En effet, il existe de nombreux contenus comme des bandes annonces de films ou de concerts, des demos de jeux, etc. qui peuvent être distribués légalement.

Si l'on regarde l'activité des utilisateurs les plus actifs, la table 3.3 nous montre que certains utilisateurs peuvent insérer jusqu'à 6 contenus par jour. C'est le cas de *eztv*, le premier dans notre liste et le fournisseur de séries TV le plus actif. On note également que sur les 20 utilisateurs les plus actifs, deux hébergeurs européens (français et allemand) se partagent les dix premiers. Cependant, il est faux de conclure que ces utilisateurs sont français ou allemands. En effet, en recherchant dans les noms de contenus des chaînes de caractères indiquant la langue utilisée pour ces contenus, on a trouvé un faible pourcentage de contenus en français ou allemand, mais plus d'un tiers de contenus en espagnol. Par conséquent, on conclut que les infrastructures des hébergeurs sont utilisées en fonction de leur faible coût et de leur qualité, mais pas en fonction d'une quelconque proximité géographique.

On s'étonne de constater que peu de sources initiales insèrent la majorité des contenus distribués avec BitTorrent. En effet, alors que l'on a identifié 148 millions d'utilisateurs de BitTorrent, la figure 3.6 montre que les 100 sources initiales les plus actives insèrent 30% des contenus et les 1000 sources initiales les plus actives en insèrent 60%.

Rang	Nombre de contenus	Taille (GB)	Code pays	Nom de l'AS
1	313	136	NZ	Vodafone
2	304	79	FR	OVH
3	266	152	DE	Keyweb
4	246	34	FR	OVH
5	219	186	FR	OVH
6	212	247	DE	Keyweb
7	201	535	FR	OVH
8	181	73	US	HV
9	181	17	CA	Wightman
10	180	7	SK	Energotel
11	172	161	FR	OVH
12	167	23	RU	Corgina
13	145	197	DE	Keyweb
14	140	11	FR	OVH
15	138	109	US	Aaron
16	132	12	US	Charter
17	117	119	FR	OVH
18	116	109	FR	OVH
19	114	79	NL	Telfort
20	107	225	RU	Matrix

TABLE 3.3 – Statistiques pour les 20 sources initiales les plus actives (ordonnées par nombre de contenus).

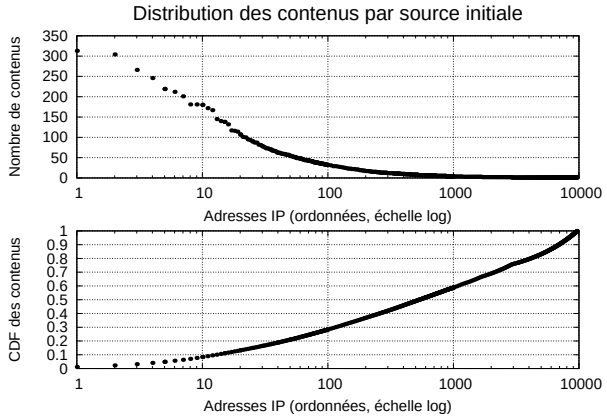

FIGURE 3.6 – Distribution du nombre de contenus insérés par chaque source initiale.

Résumé des contributions

On a montré dans cette section qu'il était possible de profiler 70% des sources initiales de BitTorrent en continu sur 48 jours. Outre le problème de protection de la vie privée, il est surprenant de constater que les entreprises luttant contre le piratage n'attaquent pas directement les sources initiales. On a en effet montré que seul un petit nombre de sources initiales était responsable de 60% de tous les contenus distribués avec BitTorrent. On a ici deux hypothèses. La première hypothèse est que la difficulté technique est trop importante. Cependant, même si

la méthode des identifiants est difficile à concevoir et à mettre en œuvre, la méthode triviale est extrêmement simple et fonctionne dans de nombreux cas. De plus, des mesures non techniques, comme l'infiltration de groupes privés, permettraient d'améliorer la performance de la méthode triviale. La deuxième hypothèse est que les entreprises luttant contre le piratage n'ont pas intérêt à attaquer les sources initiales. Il est difficile de connaître les raisons de ce manque d'intérêt. Mais, il faut noter l'intérêt contradictoire des ayants droit qui veulent lutter contre le piratage et des entreprises qui vivent du piratage en traquant les pirates. On pourrait également spéculer que la volonté des ayants droit n'est pas d'arrêter le piratage mais de le limiter, car celui ci, lorsqu'il est contrôlé, peut être considéré comme de la publicité gratuite.

3.3.3 Identification et activité des utilisateurs de Skype

On a montré qu'il était possible de profiler des utilisateurs de BitTorrent, c'est-à-dire de lier l'identité réseau d'un utilisateur (son adresse IP) avec son activité (les torrents auxquels il participe).

Cependant, même s'il s'agit déjà d'un problème majeur de protection de la vie privée, sa sévérité est mitigée par deux difficultés dans l'application pratique de ce type d'attaques. Premièrement, l'identification sociale d'un utilisateur par son adresse IP est difficilement réalisable en pratique. En effet, à part le FAI, personne — en particulier, un attaquant sans moyens dédiés — ne peut faire le lien de manière systématique entre l'identité réseau et l'identité sociale. Par conséquent, sans identité sociale, l'impact réel d'un profilage sur la vie privée d'un internaute est faible. Il peut y avoir des astuces pour faire ce lien. Par exemple, en faisant un *reverse DNS* sur l'adresse IP, le nom DNS associé peut révéler l'identité sociale de l'utilisateur : `arnaudlegout.com`. Il est également possible de faire une requête HTTP sur le port 80 et de tomber sur une page personnelle donnant une identité. Cependant, ces astuces sont limitées par leur faible taux de succès et ne constituent donc, en aucun cas, un problème majeur de protection de la vie privée. On peut également remarquer qu'une personne utilisant un service DNS dynamique ou faisant tourner son propre site Web est consciente — et même désireuse — que son identité réseau soit facilement associée à son identité sociale.

Deuxièmement, dans de nombreux cas, l'adresse IP n'est pas un identifiant unique, mais partagé par de nombreux utilisateurs. En effet, le manque d'adresses IP publiques dans des pays connaissant un fort développement d'Internet conduit à l'utilisation soit de NAT à l'échelle des FAI, soit de réseaux nativement IPv6 mais devant communiquer avec le reste du monde au travers d'une passerelle IPv6/IPv4. Dans les deux cas, des milliers voire des dizaines de milliers d'utilisateurs peuvent être multiplexés sur une même adresse IP.

Il est en théorie possible de pallier ces deux difficultés.

Pour pallier la première difficulté, il faut trouver un moyen, pour un attaquant sans moyens dédiés, de lier identité réseau et identité sociale pour un grand nombre d'internautes. Un

tel moyen est d'utiliser un système de voix sur IP populaire qui offre à la fois un annuaire intégré et des communications pair-à-pair entre les utilisateurs. L'annuaire contient, par définition, une correspondance entre une identité sociale (nom, prénom, lieu de résidence, etc.) et une identité applicative (un identifiant unique dans le système). Les communications pair-à-pair permettent, en appelant un utilisateur ciblé, de lier identité applicative (et donc sociale) avec l'identité réseau en capturant les champs source ou destination des entêtes IP provenant de cet utilisateur.

Skype est un système de voix sur IP regroupant 500 millions d'utilisateurs enregistrés et plusieurs dizaines de millions d'utilisateurs simultanés. En plus d'être extrêmement populaire, il offre un annuaire et des communications pair-à-pair entre les utilisateurs lorsqu'au moins un des deux utilisateurs accepte les connexions entrantes. Cependant, pour passer de la théorie à la pratique, il y a plusieurs difficultés à surmonter.

- Est-ce possible de déterminer avec certitude l'identifiant applicatif Skype (que l'on appelle *Skype ID* dans la suite) d'un utilisateur ciblé ? Un utilisateur peut ne fournir, par exemple, qu'un nom sans prénom ni lieu de résidence lors de son enregistrement dans l'annuaire de Skype. Ainsi, lors d'une recherche dans l'annuaire, il peut y avoir des milliers de réponses.
- Est-ce possible de déterminer quels paquets IP proviennent de l'utilisateur ciblé ? Lors d'une session Skype, un utilisateur va échanger des paquets avec des dizaines d'autres utilisateurs, même s'il n'a pas de communication applicative (voix, texte ou vidéo) avec eux. Trouver, parmi ces utilisateurs, qui est la destination d'une communication applicative est difficile puisque l'ensemble des communications est chiffré.
- Est-ce possible de forcer une communication avec un utilisateur ciblé sans être détecté ? Si chaque communication se traduit par une notification de la cible (typiquement *l'utilisateur X essaie de vous joindre*), l'attaque pourra difficilement être exécutée sur un grand nombre d'utilisateurs ciblés et répétée dans le temps.

Pour pallier la deuxième difficulté, il faut trouver un moyen d'identifier une signature de la machine d'un utilisateur, même si cette machine est derrière un NAT. Il existe quelques techniques documentées sur l'identification de la signature d'une machine [60, 22]. Cependant, à notre connaissance, ces techniques n'ont pas été utilisées pour faire des attaques par association de machines derrières des NAT. Il y a plusieurs difficultés spécifiques à ce problème.

- On désire montrer la faisabilité d'une attaque par association. Notre hypothèse est que, si deux applications tournent au même moment sur la même machine, la probabilité que ces applications soient lancées par la même personne est élevée. Cependant, est-ce qu'il est fréquent, en pratique, d'avoir deux applications (Skype et BitTorrent) actives en même temps qui permettent de faire une attaque par association ?
- L'identification de la signature d'une machine fonctionne dans des scénarios contrôlés [60, 22]. Est-ce toujours possible de trouver la signature d'une machine inconnue dans un environnement réel ?

On va montrer, dans la suite, que l'on peut non seulement lier systématiquement identité réseau et identité sociale en utilisant Skype, mais que l'on peut également faire des attaques par association entre Skype et BitTorrent même sur des machines qui sont derrière des NAT ou des passerelles IPv6/IPv4. Une description exhaustive de nos résultats, une discussion des aspects éthiques et les mesures que l'on a prises pour préserver la vie privée des personnes impliquées dans nos expérimentations sont disponibles dans une publication IMC'2011 [32].

Méthodologie

On va présenter dans la suite les deux méthodes que l'on a conçues pour résoudre deux problèmes : comment lier une identité sociale à une identité réseau ? comment associer deux applications à la même machine lorsque cette dernière est derrière un NAT ?

Lier une identité sociale à une identité réseau se déroule en deux étapes. La première étape consiste à trouver l'identifiant Skype d'un utilisateur ciblé. En cherchant un nom et un prénom dans l'annuaire Skype, on a souvent un grand nombre de résultats. Cependant, en utilisant des informations annexes disponibles dans l'annuaire (lorsqu'elles sont renseignées par les utilisateurs) comme le pays ou la ville de résidence, l'URL d'une page personnelle, etc. il est souvent possible de lever toutes ambiguïtés. Lorsqu'il y a toujours plusieurs Skype ID candidats pour un utilisateur ciblé, il est possible (en utilisant la méthode décrite dans la deuxième étape ci-dessous) de trouver l'adresse IP correspondant à chaque Skype ID, de localiser ces adresses IP et de ne garder que le Skype ID qui correspond à l'utilisateur ciblé. Par exemple, la localisation d'une adresse IP peut donner une zone géographique ou un préfixe réseau correspondant à un employeur.

La deuxième étape, la plus difficile, est une contribution majeure de ce travail. L'attaquant qui cherche à obtenir l'adresse IP correspondant à un Skype ID doit accepter les connexions entrantes. Cet attaquant va appeler (on le nomme *appelant* dans la suite) un utilisateur ciblé (que l'on nomme *appelé*). L'objectif est de forcer l'envoi de paquets IP entre l'appelé et l'appelant. Il suffit alors d'extraire le champ source ou destination dans l'entête IP pour avoir l'adresse IP publique de l'appelé.

Cependant, l'intégralité du trafic Skype est chiffrée et chaque utilisateur ouvre de multiples connexions avec d'autres utilisateurs ; dans nos expérimentations, on trouve de l'ordre d'une centaine de connexions. Il est donc difficile de savoir si les paquets IP envoyés ou reçus proviennent de l'appelé ou d'un autre pair.

Pour résoudre ce problème, on a créé une méthode exploitant trois schémas de communications caractéristiques d'une communication entre un appelant et un appelé. Ces schémas sont présentés sur la figure 3.7. Ils correspondent aux trois états possibles d'un appelé.

1. L'appelé est connecté (client Skype démarré et connecté à l'infrastructure Skype) et public (acceptant les connexions entrantes). Dans ce cas, l'infrastructure va demander à l'appelant

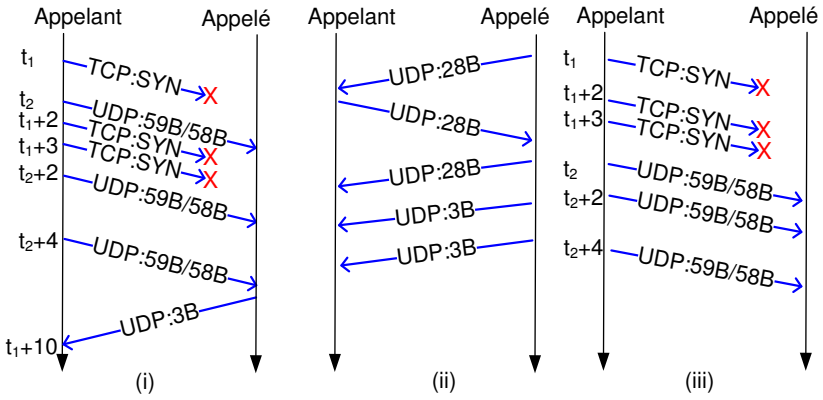

FIGURE 3.7 – Schémas de communications. (i) L'appelé est connecté et public ; (ii) l'appelé est connecté, mais derrière un NAT ; (iii) l'appelé est déconnecté. Les croix correspondent aux paquets SYN qui sont jetés pour appeler furtivement.

de se connecter directement à l'appelé. L'infrastructure Skype doit donc donner l'adresse IP de l'appelé à l'appelant, mais cette communication étant chiffrée, il n'est pas possible ici de récupérer cette adresse IP. Par contre, lorsque l'appelant va se connecter à l'appelé, l'adresse IP de l'appelé sera contenue dans le champ destination des paquets IP sortants identifiés, grâce au schéma i), comme envoyés à l'appelé.

2. L'appelé est connecté mais derrière un NAT ou tout équipement réseau empêchant les connexions entrantes. Dans ce cas, l'infrastructure Skype va demander à l'appelé de se connecter à l'appelant. L'adresse IP de l'appelé sera contenue dans le champ source des entêtes des paquets IP entrants identifiés, grâce au schéma ii), comme provenant de l'appelé.
3. L'appelé est déconnecté mais a été connecté dans les 72 dernières heures (si un client Skype est déconnecté pendant plus de 72 heures, Skype ne retourne plus l'adresse IP du client appelé à l'appelant). Dans ce cas, l'infrastructure va demander à l'appelant de se connecter directement à l'appelé (comme pour l'état 1). Lorsque l'appelant va se connecter à l'appelé, l'adresse IP de l'appelé sera contenue dans le champ destination des paquets IP sortants identifiés, grâce au schéma iii), comme envoyés à l'appelé.

À chaque fois que l'attaquant appelle l'utilisateur ciblé, cela déclenche une notification de demande de connexion. Notre objectif est de montrer que cette attaque peut être furtive, c'est-à-dire sans aucune notification au niveau de l'utilisateur ciblé. On a constaté, en expérimentant, que si les connexions TCP sont bloquées au niveau de l'attaquant (c'est-à-dire tous les messages TCP SYN sont jetés une fois que le client est connecté à l'infrastructure Skype), alors on observe toujours les schémas de la figure 3.7, mais plus aucune notification n'apparaît au niveau du client Skype de l'utilisateur ciblé. Il est probable que le client Skype attende qu'une connexion TCP soit établie entre l'appelant et l'appelé avant de notifier l'appelé de la demande d'appel.

Skype a deux paramètres de protection de la vie privée qui peuvent être configurés dans

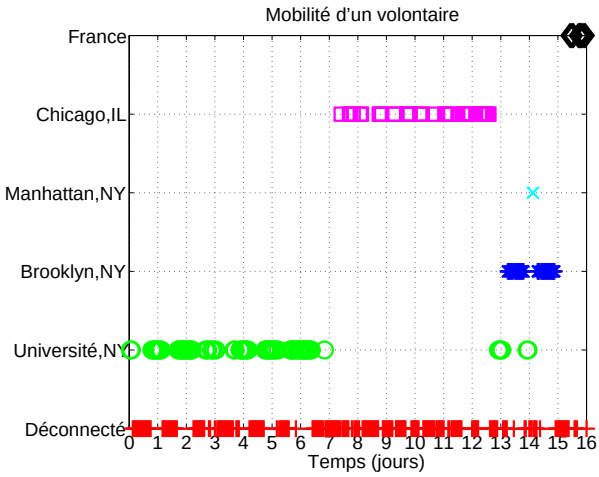

FIGURE 3.8 – Exemple de mobilité d'un volontaire.

le client. Le premier consiste à accepter uniquement les appels des utilisateurs dans une liste de contacts. Le deuxième consiste à bloquer explicitement un autre utilisateur de Skype pour empêcher toutes connexions. Nous avons testé ces deux paramètres avec notre attaque et nous avons constaté qu'aucun de ces paramètres ne permettait d'empêcher l'attaque.

Associer deux applications à la même machine lorsque cette dernière est derrière un NAT nécessite l'identification d'une signature de la machine. On utilise ici une technique introduite par Bellovin [22] exploitant la possibilité de prédire le champ identification (appelé IP-ID dans la suite) des entêtes IP. Ce champ identification (de 16 bits) est utilisé pour la reconstruction de fragments de paquets IP à la destination. On a vérifié que sur Windows XP, Vista et Seven (qui sont utilisés par 90% des ordinateurs) le IP-ID était généré de manière séquentielle. De plus, Bellovin [22] a observé que les NAT ne modifient pas le IP-ID.

Comme le IP-ID est déterminé par la pile réseau du système d'exploitation, la génération séquentielle du IP-ID de chaque paquet IP est faite dans l'ordre d'émission des paquets, indépendamment de l'application qui génère ces paquets. Il est ainsi, en théorie, possible d'identifier deux applications émettant simultanément des données depuis une même machine, grâce à la présence de IP-ID très proches dans les paquets IP reçus. Nous sommes, à notre connaissance, les premiers à montrer que l'on peut utiliser cette technique pour l'association de deux applications à la même machine lorsqu'elle est derrière un NAT.

Résultats

Lorsque l'on peut lier identité sociale et identité réseau, la première attaque qui est réalisable est de suivre la mobilité d'un utilisateur ciblé. En effet, l'adresse IP obtenue lors de l'attaque peut être localisée géographiquement en utilisant un service comme MaxMind [3] qui donne pour chaque adresse IP l'AS, la ville et le pays. Cependant, dans certains cas on peut obtenir encore

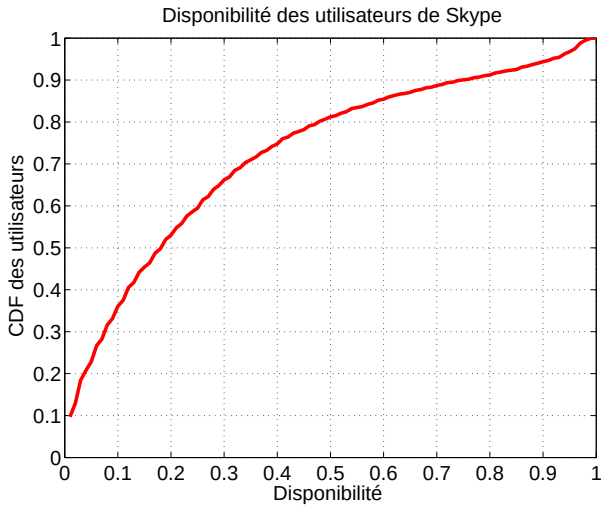

FIGURE 3.9 – CDF de la disponibilité des utilisateurs de Skype.

plus d'informations sur la localisation de l'utilisateur ciblé. Par exemple, dans une ville bien connectée ou utilisant un réseau Wi-Fi public, la résolution DNS de l'adresse IP peut donner une information précise à quelques dizaines de mètres de la localisation d'un utilisateur.

La figure 3.8 présente un exemple de trace de mobilité d'un utilisateur ciblé volontaire pour cette expérimentation. Tous les déplacements identifiés sur la figure ont été confirmés par cette personne On observe les déplacements professionnels de cette personne, mais également les connexions depuis les logements utilisés lors des déplacements et une connexion depuis un café Internet à Manhattan. Notons qu'en suivant la mobilité des amis Facebook de cette personne, on aurait pu savoir qui cette personne rencontrait. On voit donc ici que l'information de mobilité est une donnée privée et très sensible en particulier lorsqu'elle est corrélée avec la mobilité d'autres personnes

Cependant, même si cet exemple illustre parfaitement le risque d'atteinte à la vie privée d'un utilisateur fréquent de Skype, on peut légitimement se demander s'il est possible d'observer une réelle mobilité pour un utilisateur typique de Skype. Pour répondre à cette question, on a optimisé notre attaque (les détails sont donnés dans notre publication IMC'2011 [32]) pour pouvoir la lancer simultanément sur 10 000 utilisateurs de Skype choisis au hasard (mais ayant utilisé Skype dans les 72 dernières heures précédant notre expérimentation) toutes les heures pendant deux semaines.

Le coût de cette attaque est faible. En effet, en la lançant depuis la plate-forme EC2 [1] d'Amazon, on a eu besoin de 30 machines (1 client Skype par machine) pour un coût d'environ 500$ par semaine. En utilisant des machines virtuelles, on aurait pu réduire le nombre de machines, et donc le coût, d'un ordre de grandeur.

Sur les 10 000 utilisateurs suivis sur 2 semaines, on a pu obtenir au moins une adresse IP pour 9 500 utilisateurs (les autres n'ayant probablement pas utilisé Skype pendant les deux semaines).

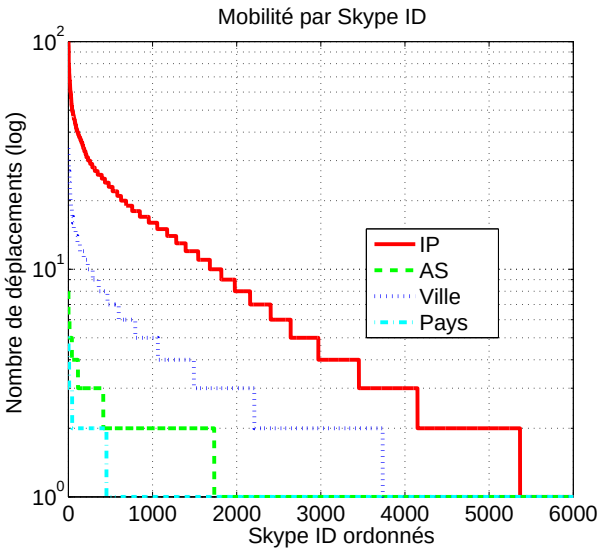

FIGURE 3.10 – Nombre de déplacements d'un utilisateur de Skype sur une période de deux semaines.

De plus, on a trouvé, en permanence, entre 2 000 et 3 000 utilisateurs connectés simultanément à chaque instant.

Les utilisateurs de Skype restent connectés longtemps. En effet, la figure 3.9 présente la disponibilité des 9 500 utilisateurs que l'on a pu attaquer. On définit la disponibilité d'un utilisateur comme étant le rapport du nombre de fois que l'on a vu cet utilisateur connecté par le nombre total de mesures faites pour cet utilisateur sur les deux semaines (soit une mesure par heure). On observe que 20% des utilisateurs ont une disponibilité de 0.5 ce qui est très élevé. Cette haute disponibilité est due au fait que Skype étant un système de voix sur IP, il y a une forte incitation pour les utilisateurs de démarrer Skype avec l'ordinateur pour toujours pouvoir être joint.

On observe que les utilisateurs de Skype se connectent de plusieurs endroits. On voit sur la figure 3.10 que 40% des utilisateurs se connectent d'au moins deux villes différentes, 19% d'AS différents et 4% de pays différents sur une période de 2 semaines.

En résumé, le cas de l'utilisateur volontaire présenté sur la figure 3.8 est représentatif du comportement d'un utilisateur typique de Skype.

Maintenant que l'on a montré que l'attaque permettant de lier identité sociale et identité réseau concerne une population très importante de dizaines de millions d'internautes, on va montrer qu'il est possible de faire des attaques par association entre Skype et BitTorrent même pour des utilisateurs qui sont derrière des NAT. Notons que l'attaque par association que l'on fait avec BitTorrent aurait pu être faite avec d'autres protocoles pair-à-pair comme eMule ou Xunlei.

Pour faire cette attaque par association, on a choisi 100 000 utilisateurs de Skype au hasard

pour lesquels on a tenté de récupérer l'adresse IP toutes les 24 heures. Parallèlement, on a récupéré l'adresse IP des pairs participants aux 50 000 plus gros torrents. Cette liste de torrents est mise à jour toutes les 24 heures d'après les informations de popularité contenues dans la réponse à la requête *scrape-all* (voir section 3.3.1) sur les trackers PublicBitTorrent qui étaient les trackers les plus populaires au moment de cette étude. Pour récupérer, toutes les heures, la liste des adresses IP des pairs participant aux 50 000 torrents, on exploite la DHT *mainline*.

Sur 100 000 utilisateurs de Skype, on en a trouvé 15 000 dont l'adresse IP a été simultanément identifiée sur Skype et BitTorrent. Cependant, sur ces 15 000 utilisateurs, la moitié (soit 7 500) ont une adresse IP utilisée par plus d'un client BitTorrent à la fois. On détecte la présence de n clients BitTorrent utilisant une même adresse IP par la présence de n ports distincts pour cette adresse. En effet, tous les clients BitTorrent modernes multiplexent tous leurs torrents sur un seul port. Par conséquent, chaque port (identifié comme étant un port utilisé par BitTorrent) sur une adresse IP correspond à un client BitTorrent différent.

Il est très probable que ces 7 500 utilisateurs soient derrière des NAT (ou des passerelles IPv6/IPv4) et qu'il y ait un utilisateur différent pour chaque client BitTorrent utilisant une même adresse IP. Mais la question est : est-ce que l'utilisateur de Skype utilise également un client BitTorrent ? Il y a deux réponses possibles : un même utilisateur utilise à la fois un client BitTorrent et Skype, ou l'utilisateur de Skype n'utilise aucun client BitTorrent. Notons également que si l'on a la preuve que ces 7 500 utilisateurs sont derrière des NAT, on n'a aucune information spécifique sur les 7 500 autres utilisateurs ; ils pourraient être derrière un NAT ou seuls à utiliser leur adresse IP.

En résumé, pour une attaque par association, on vient de montrer qu'utiliser l'adresse IP comme identifiant unique d'un utilisateur n'est pas suffisant. En effet, l'adresse IP ne permet pas de distinguer deux utilisateurs qui sont derrière le même NAT ou la même passerelle.

C'est pour cette raison que l'on utilise une procédure de vérification permettant de contrôler qu'un même utilisateur utilise à la fois Skype et BitTorrent. À chaque fois que l'on trouve une adresse IP utilisant en même temps Skype et BitTorrent, on déclenche simultanément un appel Skype et un *HANDSHAKE* BitTorrent vers cette adresse IP. Si la distance (sur un anneau de 2^{16} éléments) entre les IP-ID des paquets IP de chaque application est inférieure à 1 000, alors on conclut que les applications sont sur la même machine.

Notre procédure de vérification a besoin que l'utilisateur vérifié accepte les connexions BitTorrent entrantes. Cependant, c'est rarement le cas ; l'utilisateur peut être derrière un NAT ou une passerelle qui ne relaie pas les connexions entrantes, ou le client BitTorrent peut refuser les connexions entrantes parce que son voisinage est déjà au maximum. Notre objectif est de montrer la possibilité de réaliser une procédure de vérification dans la réalité, mais pas de faire une vérification exhaustive. Un véritable attaquant pourrait, pour faire une vérification exhaustive, s'enregistrer sur la DHT pour tous les torrents et déclencher la procédure de vérification (c'est-à-dire l'appel Skype dans ce cas) dès qu'il reçoit un *HANDSHAKE* BitTorrent.

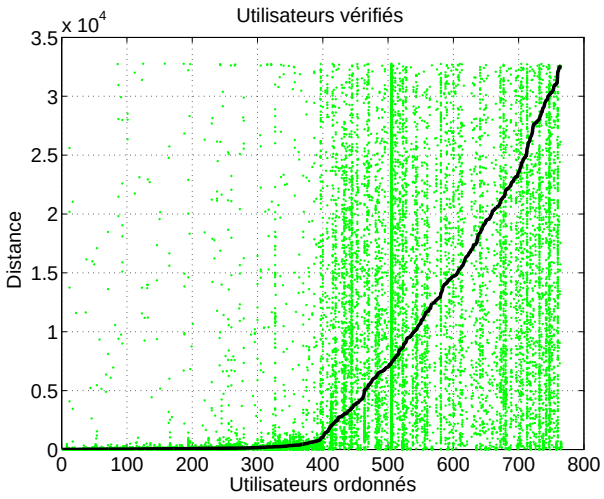

FIGURE 3.11 – Après deux semaines, on affiche le 9e décile (courbe noire) de la distance la plus courte entre les IP-ID du premier paquet Skype et du premier paquet BitTorrent reçus de l'utilisateur en cours de vérification. *Il y a un point par vérification.*

La figure 3.11 montre le résultat de notre procédure de vérification. On a pu réaliser cette procédure sur 765 utilisateurs uniques. On voit que pour 400 de ces utilisateurs le 9e décile est inférieur à 1 000. On peut donc conclure qu'ils utilisent simultanément Skype et BitTorrent. Par conséquent, l'attaque par association est possible pour eux. On ne peut pas avoir de certitude pour les 365 autres utilisateurs, parce qu'il peut y avoir des faux négatifs dus à des IP-ID non séquentiels. Cependant, étant donné que 90% des ordinateurs utilisent une version de Windows qui génère des IP-ID séquentiels, on pense qu'une grande partie des 365 utilisateurs n'utilisait pas BitTorrent au moment de notre procédure de vérification.

On voit donc qu'une attaque par association est complexe à mettre en œuvre, mais est réalisable à grande échelle.

Résumé des contributions

On a vu dans les section précédentes (section 3.3.1 et section 3.3.2) qu'il était possible pour un attaquant sans moyens dédiés de profiler des internautes en les identifiant par leur identité réseau. Cependant, lorsqu'il est possible de lier l'identité réseau à l'identité sociale, ce profilage devient un problème majeur de protection de la vie privée.

On a montré qu'il était possible de lier identité réseau et identité sociale en exploitant Skype, le système de voix sur IP le plus populaire. On a également montré qu'il était possible d'utiliser ce lien pour tracker la mobilité d'un internaute ciblé. Les techniques que l'on a mises en œuvre sont applicables à des dizaines de milliers d'internautes pour un coût modeste. Finalement, on a montré que l'on pouvait réaliser une attaque par association entre Skype et BitTorrent.

Il n'est pas surprenant que les applications pair-à-pair (comme Skype ou BitTorrent) exposent l'adresse IP des pairs qui communiquent entre eux. Par contre, il est surprenant de constater que,

sans moyens dédiés, un attaquant peut lier identité réseau et identité sociale, et faire des attaques par association à grande échelle. Les informations exposées par des attaques de ce type sont très sensibles : déplacements, personnes rencontrées, contenus téléchargés, etc. Si l'on considère que ces informations peuvent être collectées à l'insu des internautes pendant des mois ou des années, on est face à un problème majeur qui concerne des dizaines de millions d'internautes.

Nous avons informé Skype en novembre 2010 de ce problème et nous avons proposé [32] des solutions permettant de réduire de manière très importante l'impact de ces attaques. À ce jour (juillet 2011) l'attaque fonctionne toujours comme décrite.

3.3.4 Identification et activité des utilisateurs de Tor

Il existe deux grandes catégories d'internautes : ceux qui savent qu'ils ne comprennent pas le fonctionnement d'Internet et qui se trouvent totalement démunis face au type d'attaques sophistiquées décrites dans ce chapitre ; et ceux qui pensent comprendre son fonctionnement (les *geeks*) et imaginent être suffisamment compétents pour être à l'abri de ces attaques. Ces geeks imaginent que des mesures consistant à cacher leur adresse IP sont suffisantes. De telles mesures sont, par exemple, l'utilisation d'un proxy ou, encore mieux, l'utilisation d'un réseau garantissant l'anonymat comme *Tor* [41].

Tor est une implémentation des réseaux de Chaum [33] dont la spécificité est de minimiser la latence dans les communications entre utilisateurs. Un réseau de Chaum est un ensemble de machines appelées *mix*. Le but de chaque mix est de rendre difficile le lien entre messages entrant et messages sortant. En chaînant les mix, il devient difficile pour un attaquant de faire le lien entre source et récepteur d'un message M envoyé sur le réseau de Chaum. Plus précisément, les réseaux de Chaum utilisent une cryptographie à clefs publiques et privées. Chaque mix va choisir une paire de clefs et rendre publiquement accessible sa clef publique. Lorsqu'une source S veut envoyer un message M vers un récepteur R (avec une clef publique K_R), elle va choisir au hasard plusieurs mix dans le réseau de Chaum, par exemple les mix m1 et m2 avec les clefs publiques K_{m1} et K_{m2}. En supposant que la source décide d'envoyer le message M suivant le chemin (m1,m2,R), la source va créer les messages suivants : $M' = (R, K_R(M))$, $M'' = (m2, K_{m2}(M'))$, $M''' = (m1, K_{m1}(M''))$. Chaque nouveau message est un couple (A,B) où A est une adresse et B un message pouvant être déchiffré par la machine d'adresse A. La source va envoyer M''' à $m1$ qui va déchiffrer le message, obtenir M'' et l'envoyer à $m2$. $m2$ va déchiffrer le message, obtenir M' et l'envoyer à R. R va déchiffrer le message et trouver le message M.

On note que sur le chemin aucun mix ne peut savoir qui est la source et la destination, ni le contenu du message. Cependant, en contrôlant l'ensemble des mix sur le chemin (et en supposant que le nombre de mix utilisés est connu), un attaquant pourrait connaître le chemin suivi par un message et donc la source et la destination (mais pas le contenu du message). Plus le nombre de mix sur le chemin est grand et plus la traçabilité est difficile. Cependant, le délai pour acheminer un message augmente avec le nombre de mix. Par conséquent, il existe un compromis

entre traçabilité et latence.

Tor est une implémentation des réseaux de Chaum qui est utilisée en partie pour accéder anonymement à des serveurs publics. Il introduit à cet effet la notion de *exit node* qui est le dernier mix sur le chemin vers le serveur. Cependant, à la différence des réseaux de Chaum classiques, cet *exit node* connaît le récepteur (c'est-à-dire le serveur dans ce cas) et a accès au contenu non chiffré lorsque le serveur ne supporte pas le chiffrement des réseaux de Chaum (ce qui est le cas le plus fréquent), ni le chiffrement de bout en bout. Comme n'importe qui peut déclarer un *exit node*, il est possible pour un attaquant d'observer le trafic qui circule sur Tor. Cependant, il n'est pas possible de connaître la source du trafic en observant les entêtes de paquets IP, mais si cette source envoie un identifiant dans un flux de données non chiffré, l'attaquant pourra récupérer cette information. Pour cette raison, Tor ne garantit pas l'anonymat d'un utilisateur d'une application non sécurisée (c'est-à-dire qui permet de retrouver l'identité de l'utilisateur) et recommande l'utilisation d'outils effaçant les identifiants comme privoxy [78] ; mais la détection et la suppression de ces identifiants par un tel outil est une tache très difficile parce qu'ils peuvent être transmis de façon subtile, indirecte ou involontaire.

Une autre spécificité de Tor est la notion de *circuits* qui a été introduite d'après les auteurs pour améliorer la sécurité. Le principe est de multiplexer plusieurs flux de données provenant de la même source pendant un laps de temps, tous ces flux empruntant le même chemin.

On va montrer, dans la suite, qu'il est possible de trouver l'adresse IP d'un utilisateur de BitTorrent sur Tor. Tor ne garantit aucun anonymat pour les utilisateurs de BitTorrent car BitTorrent n'est pas considéré comme une application sécurisée. Cependant, en exploitant la notion de circuit, on montre qu'il est possible, à partir de la première attaque, de lier d'autres flux non BitTorrent et sécurisés à cet utilisateur ; on appelle cela une attaque par contamination. On va commencer par présenter les attaques puis on discutera du résumé de nos résultats. L'intégralité de nos contributions est disponible dans une publication LEET'11 [31].

Méthodologie

Pour mettre en œuvre les trois attaques décrites dans la suite, on a instrumenté et capturé le trafic sur 6 *exit nodes* répartis à travers le monde (deux en Asie, deux en Europe et deux aux États-Unis) pendant 23 jours du 15 janvier au 7 février 2010.

La première attaque est basée sur un détournement des requêtes au tracker. Lorsqu'un pair BitTorrent veut obtenir une liste de pairs, il doit contacter un tracker. Si cette demande est faite sur Tor et qu'elle passe par un des *exit nodes* instrumentés, on peut intercepter la réponse du tracker et remplacer l'un des pairs retourné par un pair que l'on contrôle. Ainsi, le pair contrôlé recevra une requête de connexion du pair attaqué. Cependant, le pair attaqué pourrait également se connecter au pair contrôlé à travers Tor ; dans ce cas, l'adresse IP observée sera celle d'un *exit node* et non l'adresse IP du pair. Comme la liste de tous les *exit nodes* est publique, on peut facilement comparer l'adresse IP observée avec celle de tous les *exit nodes*.

On a observé sur nos mesures que 72% des pairs BitTorrent utilisent Tor uniquement pour les connexions au tracker, mais pas pour les connexions entre pairs. Par conséquent, cette attaque permet de trouver l'adresse IP des pairs BitTorrent utilisant Tor dans 72% des cas. Les pairs utilisent rarement Tor pour les communications pair-à-pair principalement pour des raisons de performance. En effet, on a montré qu'il y avait une importante baisse de performance lorsque Tor est utilisé pour les communications pair-à-pair [28].

La deuxième attaque consiste à exploiter les informations disponibles sur la DHT. En effet, la DHT de BitTorrent utilise le protocole UDP qui n'est pas supporté par Tor. Par conséquent, lorsqu'un pair s'enregistre sur la DHT il le fait avec son adresse IP publique : typiquement, un pair fournit lors de son enregistrement son adresse IP, le numéro de port sur lequel il écoute et l'identifiant du torrent auquel il participe. Pour corréler l'adresse IP publique du pair contenu dans la DHT avec le trafic BitTorrent que l'on observe sur un *exit node*, on utilise le port et l'identifiant de contenu qui est dans la requête au tracker ou dans un HANDSHAKE entre deux pairs. Comme le port sur lequel un pair écoute est choisi au hasard uniformément sur l'ensemble des ports disponibles et que les torrents observés sont généralement petits, la probabilité de collision (c'est-à-dire de deux pairs dans le même torrent et écoutant sur le même port) est faible. En cas de collision lors de nos expérimentations, on considère que l'attaque a échoué.

La troisième et dernière attaque est une attaque par contamination. Un circuit contient plusieurs flux multiplexés provenant de la même source. Par conséquent, il suffit de trouver l'adresse IP de la source d'un seul flux d'un circuit pour associer cette adresse IP à tous les autres flux du même circuit. Cette attaque par contamination intracircuit est triviale puisqu'elle découle de la notion même de circuit. Cependant, cette attaque permet de révéler l'adresse IP pour des flux provenant d'applications autres que BitTorrent et pouvant être sécurisées. Il est également possible de faire une attaque par contamination intercircuit. Cette attaque, plus sophistiquée, nécessite d'avoir trouvé l'adresse IP source pour un flux d'une application qui envoie un identifiant unique. Par la suite, il est possible d'utiliser cet identifiant unique pour associer l'adresse IP de la source à d'autres circuits sur lesquels passent le flux contenant l'identifiant unique. On a montré qu'une attaque intercircuit était réalisable en pratique en utilisant, par exemple, l'identifiant unique du pair envoyé lors du HANDSHAKE de BitTorrent.

Résultats

On a trouvé l'adresse IP source pour 9% de l'ensemble des flux sortant par les *exit nodes* que l'on a mesurés. De plus, les flux pour lesquels on a trouvé l'adresse IP ne se limitent pas à des flux BitTorrent. En effet, en exploitant l'attaque par contamination, on a pu associer l'adresse IP d'une source à 193% de flux en plus des flux BitTorrent. Parmi ces flux non BitTorrent, il y avait 27% de flux HTTP. En tout, on a retrouvé l'adresse IP d'environ 10 000 sources utilisant Tor pour rester anonyme.

Pour comprendre la motivation des utilisateurs de BitTorrent à rester anonyme en utilisant

Rang	#	%	Surreprésentation	Pays
1	958	14	0.9	États-Unis
2	937	13	5.6	Japon
3	887	13	2.8	Allemagne
4	369	5	1.3	France
5	354	5	1.8	Pologne
6	236	3	0.9	Italie
7	232	3	0.6	Royaume-Uni
8	231	3	-	Chine
9	203	3	0.7	Canada
10	200	2	1.4	Russie

TABLE 3.4 – Popularité et surreprésentation par pays des utilisateurs de BitTorrent sur Tor.

FIGURE 3.12 – Nuage de noms des contenus téléchargés par des utilisateurs de BitTorrent aux États-Unis (gauche), au Japon (milieu) et en Allemagne (droite) (figure générée par www.wordle.net sous licence CC 3.0). *On extrait les catégories des contenus téléchargés et on varie linéairement la taille des catégories en fonction de la fréquence des contenus dans chaque catégorie.*

Tor, on a classé les utilisateurs de BitTorrent sur Tor par pays. Ensuite, pour observer s'il y avait une surreprésentation d'utilisateurs de BitTorrent sur Tor dans certains pays par rapport aux utilisateurs de BitTorrent qui n'utilisent pas Tor, on a comparé la répartition par pays d'un instantané (tel que décrit dans la section 3.3.1) collecté le 22 août 2009 et regroupant environ 10 000 000 de pairs. Par exemple, une surreprésentation de 2 signifie que le pourcentage, pour un pays donné, des utilisateurs de BitTorrent sur Tor est deux fois plus grand que le pourcentage de ceux n'utilisant pas Tor.

On remarque sur la table 3.4 qu'il y a une surreprésentation importante pour le Japon et l'Allemagne. Lorsque l'on compare les catégories des contenus téléchargés sur la figure 3.12, on constate que les pairs sur Tor des deux pays en surreprésentation, le Japon et l'Allemagne, téléchargent beaucoup plus de contenus à caractère pornographique que les pairs des État-Unis qui ont une surreprésentation proche de 1. On en conclut qu'une motivation majeure pour rester anonyme avec BitTorrent est le téléchargement de contenus pornographiques.

On va maintenant regarder les catégories des sites Web visités par les utilisateurs de Tor. La figure 3.13 montre que les catégories visitées par les utilisateurs de Tor en général (première colonne) sont différentes de celles visitées par les utilisateurs de BitTorrent sur Tor (deuxième colonne). Il est probable que cette différence s'explique par le fait que les utilisateurs de BitTorrent ne visitent pas certains sites Web sur lesquels on peut obtenir les mêmes contenus qu'avec

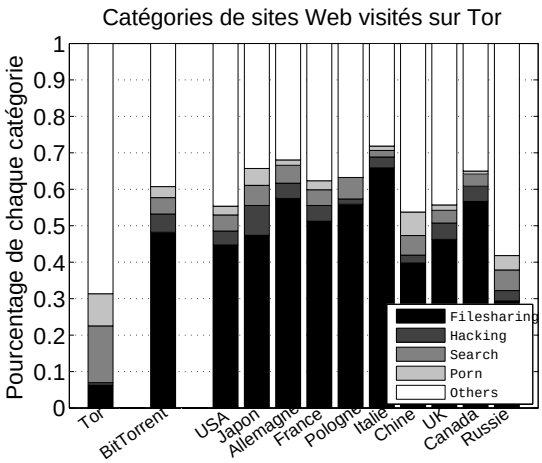

FIGURE 3.13 – Catégorie des sites Web visités par pays d'origine. *Tor représente les catégories des sites Web visités pour l'ensemble du trafic Web sur les exit nodes instrumentés. BitTorrent représente les catégories des sites Web visités pour les flux pour lesquels on a trouvé l'adresse IP.*

BitTorrent. Il est également possible que les utilisateurs de Tor qui n'utilisent pas BitTorrent utilisent Tor pour autre chose que le partage de fichiers sous copyright, c'est en particulier ce que les 70% de la catégorie *Others* semble suggérer.

Résumé des contributions

On a vu dans ce chapitre qu'il existait un vrai risque d'atteinte à la vie privée des internautes. Une mesure naturelle pour se prémunir de ces risques est d'utiliser un système garantissant l'anonymat comme Tor. Cependant, on vient de montrer qu'il était possible d'obtenir l'adresse IP d'utilisateurs de BitTorrent sur Tor. De plus, en utilisant une attaque par contamination, il est également possible d'associer du trafic autre que BitTorrent — notamment Web — à ces utilisateurs.

Les deux failles que l'on exploite sont difficiles à corriger. La première faille concerne l'utilisation des applications non sécurisées. Tor ne donne aucune garantie pour ce type d'applications, mais ne fournit pas non plus un moyen de valider qu'une application est sécurisée. Dans certains cas, il est évident qu'une application n'est pas sécurisée, par exemple, lorsque l'application envoie l'adresse IP de la source dans le flux de données qui passe par un *exit node*. Dans d'autres cas, la fuite de l'identité de la source est beaucoup plus subtile ; dans le cas de l'exploitation de la DHT de BitTorrent, à aucun moment l'identité de la source est dans le flux de données traversant un *exit node* et ce n'est seulement qu'après avoir corrélé un numéro de port et un identifiant de contenus avec les informations de la DHT que l'on peut trouver l'identité de la source.

La deuxième faille concerne l'exploitation de la notion de circuits. Les circuits ont été introduits pour résoudre un problème de sécurité, mais on a montré qu'ils en introduisaient un autre.

Il n'existe pas de solution actuellement pour bloquer l'attaque par contamination que l'on a introduite ; ce problème a été discuté et reconnu par Roger Dingledine [40] l'un des concepteurs de Tor.

Les réseaux de Chaum représentent une avancée importante dans le domaine de la protection de la vie privée. Cependant, ils souffrent de nombreuses limitations qui rendent leur utilisation difficile pour le plus grand nombre. L'implémentation faite de ces réseaux par Tor améliore de manière importante la facilité d'utilisation, mais cette facilité est au détriment de la protection offerte.

Notre position sur le compromis entre facilité d'utilisation et protection de la vie privée est que si un utilisateur peut accepter une réduction de la protection de sa vie privée (au profit d'une meilleur facilité d'utilisation), c'est que son cas ne justifie probablement aucune mesure de protection. En effet, étant donné le faible nombre de personnes cherchant à protéger leur anonymat, il serait facile pour un attaquant de surveiller l'intégralité de leurs communications si cet attaquant savait où regarder. En utilisant un réseau comme Tor pour accéder à des ressources non chiffrées de bout en bout, c'est justement cette information que fournit l'utilisateur : « je cherche à me cacher et vous pouvez trouver tout le trafic que je cherche à cacher sur un *exit node* de Tor ». Étant donné qu'il est très difficile de garantir qu'aucune information personnelle ne pourra être interceptée sur un *exit node*, nous considérons que l'utilisation de Tor pour préserver son anonymat (lors de l'accès à des ressources non chiffrées de bout en bout) expose à plus de risques qu'elle n'en prémunit.

3.4 Discussion

« Pourquoi y a-t-il des problèmes de protection de la vie privée sur Internet ? » Pour répondre à cette question, il faut remonter aux années 1960 lorsque les bases d'Internet ont été posées. À cette époque, le microprocesseur qui est la base de toute l'électronique et de l'informatique actuelle n'existait pas. Les ordinateurs étaient des machines rares et chères, complexes à utiliser et dédiées à des taches très spécifiques. Comme il n'existait aucun moyen d'utiliser ces machines à distance, l'idée de les connecter en réseau est apparue. Le seul réseau de communication existant à l'époque était le réseau téléphonique. Cependant, ce réseau dédié à la voix n'est pas adapté aux communications entre ordinateurs qui avaient pour principale caractéristique d'être en rafales. Il fallut donc repartir de zéro et concevoir une architecture entièrement nouvelle. La motivation initiale d'Internet était donc d'accéder à distance à des ressources rares et chères ; les personnes qui avaient besoin de ces accès étaient des chercheurs ou des ingénieurs dans un cadre strictement professionnel. À la fin des années 1980 — juste avant l'invention du Web par Tim Berners Lee, les fondements architecturaux et protocolaires d'Internet étaient établis tels qu'on les connaît aujourd'hui. Personne n'avait prévu à cette époque l'explosion de la popularité d'Internet parce qu'il n'existait aucune application accessible au grand public.

Deux choix architecturaux fondateurs d'Internet sont ceux de *best effort* et de *end-to-end argument* ; ce sont ces choix qui permettent à Internet de fonctionner aujourd'hui avec des applications pour lesquelles il n'a pas été conçu. Une architecture *best effort* n'offre aucune garantie ; alors que ce type d'architecture peut sembler surprenant, il s'agit en fait de la clef de l'incroyable souplesse d'Internet. Comme il n'est pas possible de faire un réseau qui soit optimisé pour toutes les applications, le principe d'une architecture *best effort* est de garder le réseau le plus simple possible ; si une optimisation est nécessaire (par exemple, pour rendre une communication fiable), cela doit être fait de bout en bout, proche de l'application — c'est le principe du *end-to-end argument*. Cependant, le *end-to-end argument* stipule que si une optimisation est d'intérêt commun (par exemple, le routage) ou qu'elle ne peut pas être faite de bout en bout (par exemple, la qualité de service) alors elle doit être faite dans le réseau.

Ces deux choix architecturaux ont parfaitement fonctionné jusqu'au milieu des années 1990. Ensuite, un basculement d'un Internet académique (c'est-à-dire par des scientifiques pour des scientifiques) vers un Internet commercial a commencé, ce qui a eu pour conséquence une ossification du réseau. Il est ainsi devenu impossible d'apporter des modifications au réseau [1] et toutes les améliorations qui auraient nécessité une telle modification (parce que ne pouvant être faites de bout en bout) n'ont pu être réalisées ; ainsi, deux des choix architecturaux fondateurs d'Internet ne s'appliquaient plus.

Parmi les améliorations qui n'ont pas pu être apportées avant le milieu des années 1990 (parce que le besoin n'existait pas encore) il y a la protection de la vie privée des internautes. Ce besoin n'est apparu qu'à la fin des années 1990, avec l'explosion de la popularité et de l'utilisation du Web par le grand public (avant cette date, des chercheurs avaient contribué à ce domaine [33], mais pour résoudre un problème théorique qui ne se posait pas de manière pratique à l'époque). Le besoin initial était d'empêcher la censure de sites Web et de garantir que les personnes accédant aux sites Web ne pouvaient pas être identifiées par une interception sur le chemin, l'illustration caricaturale étant celle d'un dissident chinois. Cependant, en quelques années, les services offerts sur Internet se sont extrêmement complexifiés et les risques d'atteinte à la vie privée multipliés : architecture pair-à-pair, réseaux sociaux, site Web utilisant des codes complexes Ajax exécutés par le navigateur de l'internaute, etc.

Ainsi, au début des années 2000 l'architecture du cœur d'Internet était figée alors que la complexité des applications développées augmentait rapidement. Cependant, le risque réel d'atteinte à la vie privée des internautes était mal compris. C'est dans ce contexte, que l'on a étudié ce risque. Nos travaux ont permis d'améliorer la compréhension de ce risque sur trois points :

– il est possible de surveiller en temps réel l'intégralité des internautes sur une application

1. C'est cette constatation qui a poussé une partie de la communauté à abandonner des propositions de modifications du réseau (notamment les travaux sur le multipoint) et à travailler sur les *overlays* dont le déploiement peut être fait uniquement de bout en bout. Cependant, après une dizaine d'années de recherche dans cette direction, il est apparu que certains problèmes ne pouvaient être résolus correctement qu'en modifiant le réseau. C'est ce qui a donné lieu à la mouvance *clean slate*.

populaire [28, 29] ;
- il est possible de corréler différentes applications pour lier identité sociale, identité réseau, identité applicative et activité sur Internet [32] ;
- il est possible de retrouver l'identité réseau d'un internaute cherchant à se cacher en utilisant un service garantissant l'anonymat [31].

La contribution majeure de ces travaux est de montrer que toutes ces attaques sont possibles sans infrastructure dédiée et à grande échelle. Cela veut dire — et c'est cela le vrai danger — que n'importe qui à travers le monde peut collecter un grand nombre d'informations personnelles sur des internautes et les utiliser à leur détriment sans que ces internautes n'aient aucun moyen de détecter ni d'empêcher (sauf à ne pas utiliser Internet) ces attaques. Les applications les plus populaires ne prennent aucune mesure pour protéger leurs utilisateurs de ce type d'attaques. Notre objectif était également d'attirer l'attention sur ces problèmes qui sont largement sous-estimés.

En cela, notre objectif a été pleinement atteint. Nos travaux ont été l'objet de nombreux articles dans la presse française et internationale, et ils figurent parmi les articles les plus téléchargés sur l'archive ouverte HAL. On donne dans la suite les statistiques de téléchargement au 1er août 2011 et une liste non exhaustive des articles.

- La publication LEET'10 [29] « *Spying the World from your Laptop – Identifying and Profiling Content Providers and Big Downloaders in BitTorrent* » a été téléchargée 4789 fois et le rapport technique associé « *Angling for Big Fish in BitTorrent* » 1412 fois. Ces travaux ont été couverts par Slashdot [7], TorrentFreak [10], The Register [11], Slyck [12], Le Monde [9], etc.
- La version initiale [30] de la publication LEET'11 [31] « *One Bad Apple Spoils the Bunch : Exploiting P2P Applications to Trace and Profile Tor Users* » a été téléchargée 1473 fois. Ces travaux ont été couverts par Slashdot [13, 14], The Register [11], Slyck [12], Le Monde [9], le blog de Bruce Schneier [16], The Kaspersky Lab Security News Service [15], NewScientist [17], etc. Les problèmes que l'on a identifiés sur Tor ont été reconnus par le projet Tor comme importants [40].

Ces articles ont cependant été peu cités. On attribue cela à deux raisons. Premièrement, il s'agit d'articles récents, voire très récents pour les papiers LEET'11 et IMC'11, qui n'ont pas encore eu le temps d'être cités. Deuxièmement, le potentiel de citations est inférieur pour ce type de travaux comparé aux travaux présentés dans le chapitre 2. Cependant, cela ne préjuge en rien de la pertinence de mener ces travaux qui ont, a notre sens, un impact important.

Chapitre 4

Conclusion

Dans cette thèse, j'ai couvert les deux principaux axes de recherche que j'ai menés entre mon arrivée à Inria, fin 2004, et août 2011.

Le premier axe, sur l'étude de l'efficacité de BitTorrent, montre expérimentalement la grande efficacité de BitTorrent en pratique, mais également l'importance du dimensionnement de la source initiale dans l'efficacité et l'incitation au partage. En particulier, on montre que les problèmes de performance qui peuvent être observés sur de vrais torrents sont dus au dimensionnement de cette source. On montre également qu'il est possible de localiser le trafic BitTorrent sans pénalité significative pour les utilisateurs et avec une réduction importante du trafic interFAI.

Le deuxième axe aborde l'étude des risques d'atteinte à la vie privée sur Internet et plus particulièrement avec l'utilisation de protocoles pair-à-pair. On montre que sans infrastructure dédiée, il est possible de suivre l'intégralité des utilisateurs et des sources de contenus de BitTorrent. On montre également qu'en exploitant les communications pair-à-pair de Skype, il est possible de lier une identité sociale à une activité réseau, en particulier la liste des contenus téléchargés par des utilisateurs de BitTorrent. Finalement, on montre que même l'utilisation d'un réseau garantissant l'anonymat, comme Tor, ne permet pas de préserver la vie privée sur Internet.

En conclusion, cette thèse a contribué de manière significative à la compréhension de BitTorrent et des problèmes de protection de la vie privée des internautes.

Il reste beaucoup de problèmes fondamentaux à étudier sur BitTorrent et plus généralement dans le domaine du transfert de données en pair-à-pair. Cependant, la difficulté pour publier des résultats sur ce sujet dans les meilleures conférences et le manque d'intérêt d'une partie de la communauté ont conduit à une forte réduction du nombre de chercheurs travaillant dans ce domaine. Je donne ici deux exemples de problèmes fondamentaux qu'il reste à explorer.

1. Une clef de l'efficacité de BitTorrent est sa robustesse à la géométrie du graphe d'interconnexion des pairs ; c'est notamment cette robustesse qui explique la possibilité de localiser fortement le trafic BitTorrent sans perte de performance pour les pairs. Cependant, les limites de cette robustesse sont mal connues. On a étudié l'impact du graphe d'intercon-

nexion des pairs sur l'efficacité avec différentes méthodes de construction : tracker, DHT, *peer exchange*. Nos résultats expérimentaux (partiellement publiés [52]) montrent une robustesse très supérieure à ce que l'on avait imaginé. Mon hypothèse est que lorsque les pairs forment des clusters, il suffit que chaque cluster reçoive de nouvelles pièces à la vitesse de la source initiale pour qu'il n'y ait aucune perte de performance. J'appelle cette arrivée de nouvelles pièces dans un cluster une *source secondaire* ; cette source peut être matérialisée par une ou plusieurs connexions avec le cluster. Il y a plusieurs questions sur l'impact de ces sources sur l'efficacité : quel est le lien entre sources secondaires et efficacité globale ? quel est l'impact de la géométrie physique (c'est-à-dire du réseau) et de la géométrie logique (c'est-à-dire de l'interconnexion des pairs) sur la possibilité d'avoir de telles sources ? est-ce que le principe des sources secondaires fonctionne quelle que soit la taille des clusters ?

2. Est-ce que la distribution de contenus diffusés en direct est possible en pair-à-pair à grande échelle ? Les grandes difficultés de la distribution de tels contenus sont : la contrainte de synchronisation de l'ensemble des pairs avec un décalage maximum de l'ordre de la dizaine de secondes, la connexion rapide à un nouveau flux, et la robustesse à l'arrivée et au départ des pairs. Ces contraintes impliquent de multiples connexions entre les pairs et la possibilité pour chaque pair d'envoyer des données sur l'ensemble de ses connexions à au moins la même vitesse que celle de la source initiale. Je pense que la faisabilité de la distribution de contenus diffusés en direct est contrainte par la capacité d'envoi des pairs. Les questions sont donc : est-ce qu'il y a un lien entre la vitesse d'envoi des pairs et le nombre de pairs qui peuvent recevoir le contenu diffusé en direct ? à partir de quelle vitesse d'envoi des pairs une diffusion est possible ?

L'exploration des risques d'atteinte à la vie privée est, par contre, un domaine nouveau et qui suscite un fort intérêt de la communauté. On a montré que sans infrastructure dédiée on pouvait lier identité sociale et activité réseau. La question qui se pose maintenant est : est-ce possible d'améliorer la protection de la vie privée sur Internet sans entièrement changer son architecture ? La tendance actuelle est soit de proposer une solution *ad hoc* (c'est, par exemple, le cas de OneSwarm [53]), soit de proposer une architecture entièrement nouvelle pour Internet (c'est le cas de CCN [56]). L'inconvénient de la solution *ad hoc* est qu'elle ne s'adresse qu'à un problème spécifique ; elle ne peut donc pas résoudre d'une manière générale le problème de la protection de la vie privée sur Internet. L'inconvénient d'une nouvelle architecture est qu'elle risque d'introduire de nouveaux problèmes de sécurité qui n'existaient pas avant. Je pense, au contraire, que la protection de la vie privée peut être significativement améliorée par quelques mesures simples adressant les deux problèmes majeurs que l'on exploite dans nos attaques (je ne propose pas ici de solutions, mais uniquement des pistes de réflexion).

1. La possibilité de faire des requêtes sur l'ensemble des utilisateurs d'un système est le problème que l'on exploite dans nos attaques sur BitTorrent. En effet, on peut facile-

ment faire des requêtes sur des millions d'utilisateurs en quelques dizaines de minutes d'une seule machine. Il est important de limiter la fréquence de ces requêtes sans pour autant pénaliser le fonctionnement du protocole. Cependant, appliquer cette limitation peut-être difficile en pratique. En effet, l'utilisation de NAT ou de passerelles IPv6/IPv4 rend difficile l'utilisation de l'adresse IP comme identifiant unique d'un utilisateur. Il faudrait donc avoir un service tiers d'attribution d'identifiants uniques certifiés qui garantisse la difficulté d'obtenir plusieurs identifiants sur la même période de temps. La migration d'IPv4 vers IPv6 peut également offrir une solution en permettant de nouveau l'utilisation d'une adresse IP comme identifiant unique. Cependant, l'attribution prévisible d'un grand nombre d'adresses IP par utilisateur pourrait permettre de contourner une mesure visant à limiter le nombre de requêtes.

2. L'impossibilité pour un utilisateur d'empêcher une communication pair-à-pair est un problème que l'on exploite dans nos attaques sur Skype. En effet, il est impossible pour un utilisateur de bloquer l'établissement d'une connexion TCP avant de connaître l'identité applicative de l'utilisateur établissant la connexion. Le problème est que durant l'établissement de la connexion l'adresse IP de la destination est utilisée (donc facilement identifiable dans le champ source ou destination de l'entête IP). Il faudrait un mécanisme qui permette d'établir une connexion indirecte avec un pair. Cette connexion pourrait être limitée à la transmission de l'identifiant applicatif (qui représente seulement quelques bytes). Une fois cet identifiant transmis, le pair peut décider d'accepter ou de refuser une connexion pair-à-pair provenant de cette adresse IP. Si l'établissement de connexions indirectes se limite à la transmission d'un identifiant, la charge est faible et pourrait être supportée par une infrastructure légère.

Ces quelques problèmes ne sont que des exemples de questions à élucider dans des domaines à fort impact. Rappelons que BitTorrent représente toujours en moyenne autour de 30% de l'ensemble du trafic d'Internet et que la protection de la vie privée des internautes est un enjeu sociétal majeur.

Bibliographie

[1] Amazon EC2. http://aws.amazon.com/.

[2] Grid'5000. https://www.grid5000.fr.

[3] MaxMind. http://www.maxmind.com/.

[4] PlanetLab platform. http://www.planet-lab.org.

[5] Bittorrent protocol specification v1.0. http://wiki.theory.org/BitTorrentSpecification, June 2005.

[6] Instrumented bittorrent client, v2, 2009. https://gforge.inria.fr/projects/bt-instru/.

[7] Anyone can play big brother with bittorrent. Slashdot, April 2010.

[8] Bluebear : Exploring privacy threats in the internet. http://planete.inria.fr/bluebear, January 2010.

[9] L'anonymat du réseau bittorrent mis en cause. Le Monde.fr, May 2010.

[10] Research exposes risks for bittorrent seeders. TorrentFreak, April 2010.

[11] Researchers spy on bittorrent users in real-time. The Register, April 2010.

[12] Surprise ! you're not anonymous on bittorrent. Slyck, May 2010.

[13] Why tor users should be cautious about p2p privacy. Slashdot, April 2010.

[14] Attacking and defending the tor network. Slashdot, March 2011.

[15] Attacking and defending the tor network. The Kaspersky Lab Security News Service, March 2011.

[16] Identifying tor users through insecure applications. Bruce Schneier blog, March 2011.

[17] Want to stay anonymous online ? don't share files. NewScientist, April 2011.

[18] T. Abbott, K. Lai, M. Lieberman, and E. Price. Browser-based Attacks on Tor. In *Proc. of PETS'07*, Ottawa, Canada, 2007.

[19] V. Aggarwal, A. Feldmann, and C. Scheideler. Can isps and p2p users cooperate for improved performance ? *Proc. of CCR*, July 2007.

[20] N. Andrade, M. Mowbray, A. Lima, G. Wagner, and M. Ripeanu. Influences on Cooperation in BitTorrent Communities. In *Proc. of the Workshop on Economics of Peer-to-Peer Systems (P2PEcon'05)*, Philadelphia, PA, August 2005.

[21] D. Bailey and N. DePetrillo. The Carmen Sandiego Project. In *Proc. of BlackHat*, Las Vegas, NV, USA, 2010.

[22] S. M. Bellovin. A Technique for Counting NATed Hosts. In *Proc. of IMW*, Marseille, FR, 2002.

[23] A. R. Bharambe, C. Herley, and V. N. Padmanabhan. Analysing and improving bittorrent performance. In *Proc. IEEE Infocom'2006*, Barcelona, Spain, April 2006.

[24] A. R. Bharambe, C. Herley, and V. N. Padmanabhan. Analyzing and Improving a BitTorrent Network's Performance Mechanisms. In *Proc. of Infocom'06*, Barcelona, Spain, April 2006.

[25] E. W. Biersack, P. Rodriguez, and P. Felber. Performance analysis of peer-to-peer networks for file distribution. In *Proc. Fifth International Workshop on Quality of Future Internet Services (QofIS'04)*, Barcelona, Spain, September 29–October 1 2004.

[26] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang. Improving traffic locality in bittorrent via biased neighbor selection. In *Proc. of ICDCS'06*, Lisboa, Portugal, July 2006.

[27] S. L. Blond, A. Legout, and W. Dabbous. Pushing bittorrent locality to the limit. *Computer Networks*, 55(3) :541 – 557, 2011.

[28] S. L. Blond, A. Legout, F. Lefessant, and W. Dabbous. Angling for big fish in bittorrent. Technical Report inria-00451282, version 1 - 28 Jan 2010, INRIA, Sophia Antipolis, France, January 2010.

[29] S. L. Blond, A. Legout, F. Lefessant, W. Dabbous, and M. A. Kaafar. Spying the world from your laptop - identifying and profiling content providers and big downloaders in bittorrent. In *Proc. of LEET'10*, San Jose, CA, USA, April 27 2010.

[30] S. L. Blond, P. Manil, A. Chaabane, M. A. Kaafar, C. Castelluccia, A. Legout, and W. Dabbous. De-anonimizing Bittorrent Users on Tor. In *Poster Session of the 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI'10)*, San Jose, CA, United States, 2010. Usenix.

[31] S. L. Blond, P. Manils, A. Chaabane, M. A. Kaafar, C. Castelluccia, A. Legout, and W. Dabbous. One bad apple spoils the bunch : Exploiting p2p applications to trace and profile tor users. In *Proc. of LEET'11*, March 29 2011.

[32] S. L. Blond, C. Zhang, A. Legout, K. Ross, and W. Dabbous. Exploring the privacy limits of real-time communication applications. In *In Proc. of IMC'2011*, Berlin, Germany, November 2011.

[33] D. Chaum, C. O. T. Acm, R. Rivest, and D. L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24 :84–88, 1981.

[34] D. R. Choffnes and F. E. Bustamante. Taming the torrent : A practical approach to reducing cross-isp traffic in p2p systems. In *Proc. of ACM SIGCOMM*, Seattle, WA, USA, August 2008.

[35] B. Cohen. Incentives build robustness in bittorrent. In *Proc. First Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, USA, June 2003.

[36] R. Cuevas, M. Kryczka, A. Cuevas, S. Kaune, C. Guerrero, and R. Rejaie. Is content publishing in bittorrent altruistic or profit-driven ? In *Proceedings of the 6th International COnference*, Co-NEXT '10, pages 11 :1–11 :12, New York, NY, USA, 2010. ACM.

[37] R. Cuevas, N. Laoutaris, X. Yang, G. Siganos, and P. Rodriguez. Deep diving into bittorrent locality. *CoRR*, abs/0907.3874, 2009.

[38] G. Dán and G. Carlsson. Dynamic swarm management for improved bittorrent performance. In *IPTPS'09*, Boston, MA, USA, 2009.

[39] S. E. Deering. Multicast routing in internetworks and extended lans. In *Proc. ACM SIGCOMM '88*, pages 55–64, Stanford, CA, August 1988.

[40] R. Dingledine. Bittorrent Over Tor isn't a Good Idea. `https://blog.torproject.org/blog/bittorrent-over-tor-isnt-good-idea`, 2010.

[41] R. Dingledine, N. Mathewson, and P. Syverson. Tor : The second-generation onion router. In *In Proc. of the 13th USENIX Security Symposium*, pages 303–320, 2004.

[42] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment issues for the ip multicast service and architecture. *IEEE Network magazine special issue on Multicasting*, 14(1) :78–88, January/February 2000.

[43] B. Fan, D.-M. Chiu, and J. C. Lui. The Delicate Tradeoffs in BitTorrent-like File Sharing Protocol Design. In *Proc. of ICNP'06*, Santa Barbara, CA, November 2006.

[44] P. A. Felber and E. W. Biersack. Self-scaling Networks for Content Distribution. In *Proc. of the International Workshop on Self-* Properties in Complex Information Systems (Self-*'04)*, Bertinoro, Italy, May 31–June 2, 2004.

[45] FortConsult. Practical Onion Hacking : Find the Real Address of Tor Clients. `http://www.fortconsult.net/images/pdf/Practical_Onion_Hacking.pdf`, 2006.

[46] P. Ganesan and M. Seshadri. On cooperative content distribution and the price of barter. In *IEEE ICDCS'05*, Columbus, Ohio, USA, June 2005.

[47] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *Proc. IEEE Infocom'2005*, Miami, USA, March 2005.

[48] S. Guha and P. Francis. Identity Trail : Covert Surveillance Using DNS. In *Proc. of PETS*, Ottawa, Canada, 2007.

[49] E. Guizzo. Closing in on the perfect code. *IEEE Spectrum*, 41 :36–42, March 2004.

[50] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang. Measurements, analysis, and modeling of bittorrent-like systems. In *Proc. ACM IMC'2005*, Berkeley, CA, USA, October 2005.

[51] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang. Measurements, Analysis, and Modeling of BitTorrent-like Systems. In *Proc. of IMC'05*, Berkeley, CA, October 2005.

[52] A. A. Hamra, N. Liogkas, A. Legout, and C. Barakat. Swarming overlay construction strategies. In *In Proc. of ICCCN'2009*, pages 2–6, San Francisco, CA, USA, August 2009.

[53] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson. Privacy-Preserving P2P Data Sharing With OneSwarm. In *Proc. of SIGCOMM*, Bangalore, India, 2010.

[54] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. Felber, A. A. Hamra, and L. Garcés-Erice. Dissecting BitTorrent : Five Months in a Torrent's Lifetime. In *PAM'04*.

[55] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. Felber, A. A. Hamra, and L. Garcés-Erice. Dissecting BitTorrent : Five Months in a Torrent's Lifetime. In *Proc. of PAM'04*, Antibes Juan-les-Pins, France, April 2004.

[56] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. Briggs, and R. Braynard. Networking named content. In *Proc. of CoNEXT*, Rome, Italy, December 2009. ACM.

[57] S. Jun and M. Ahamad. Incentives in bittorrent induce free riding. In *Proc. SIGCOMM'05 Workshops*, Philadelphia, PA, USA, August 2005.

[58] S. Jun and M. Ahamad. Incentives in BitTorrent Induce Free Riding. In *Proc. of the Workshop on Economics of Peer-to-Peer Systems (P2PEcon'05)*, Philadelphia, PA, August 2005.

[59] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should internet service providers fear peer-assisted content distribution ? In *Proc. of IMC'05*, Berkeley, CA, USA, October 2005.

[60] T. Kohno, A. Broido, and K. C. Claffy. Remote Physical Device Fingerprinting. In *Proc. of Security & Privacy*, Oakland, CA, 2005.

[61] D. Kostić, R. Braud, C. Killian, E. Vandekieft, J. W. Anderson, A. C. Snoeren, and A. Vahdat. Maintaining high bandwidth under dynamic network conditions. In *Proc. USENIX'05*, Anaheim, CA, USA, April 2005.

[62] A. Legout, N. Liogkas, E. Kohler, and L. Zhang. Clustering and sharing incentives in bittorrent systems. In *Proc. of ACM SIGMETRICS'07*, San Diego, CA, USA, June 2007.

[63] A. Legout, G. Urvoy-Keller, and P. Michiardi. Understanding bittorrent : An experimental perspective. Technical Report inria-00000156, version 3 - 9 November 2005, INRIA, Sophia Antipolis, France, November 2005.

[64] A. Legout, G. Urvoy-Keller, and P. Michiardi. Rarest First and Choke Algorithms Are Enough. In *Proc. of IMC'06*, Rio de Janeiro, Brazil, October 2006.

[65] M. Lin, J. C. S. Lui, and D.-M. Chiu. An isp-friendly file distribution protocol : Analysis, design and implementation. *IEEE TPDS*, 2009.

[66] N. Liogkas, R. Nelson, E. Kohler, and L. Zhang. Exploring the Robustness of BitTorrent Peer-to-Peer Systems. *Concurrency and Computation : Practice and Experience*, 2007. DOI : 10.1002/cpe.1187.

[67] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer. Free Riding in BitTorrent is Cheap. In *Proc. of HotNets-V*, Irvine, CA, November 2006.

[68] L. Massoulie and M. Vojnovic. Coupon Replication Systems. In *Proc. of SIGMETRICS'05*, Banff, Canada, June 2005.

[69] D. Mccoy, T. Kohno, and D. Sicker. Shining Light in Dark Places : Understanding the Tor Network. In *Proc. of PETS'08*, Leuven, Belgium, 2008.

[70] A. Odlyzko. Internet pricing and the history of communications. *Computer Networks*, 36 :493–517, 2000.

[71] M. Perry. Securing the Tor Network. In *Proc. of Black Hat*, Las Vegas, NV, 2007.

[72] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do incentives build robustness in BitTorrent ? In *Proc. of NSDI'07*, Cambridge, MA, April 2007.

[73] M. Piatek, T. Isdal, A. Krishnamurthy, and T. Anderson. One hop reputations for peer to peer file sharing workloads. In *Proc. of NSDI'08*, San Franciso, CA, USA, 2008.

[74] M. Piatek, T. Kohno, and A. Krishnamurthy. Challenges and directions for monitoring p2p file sharing networks or why my printer received a dmca takedown notice. In *HotSec'08*, San Jose, CA, USA, July 2008.

[75] M. Piatek, H. V. Madhyastha, J. P. John, A. Krishnamurthy, and T. Anderson. Pitfalls for isp-friendly p2p design. In *Hotnets-VIII*, New York City, NY, October 2009.

[76] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The bittorrent p2p file-sharing system : Measurements and analysis. In *Proc. of IPTPS*, Ithaca, NY, USA, February 2005.

[77] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The BitTorrent P2P file-sharing system : Measurements and Analysis. In *Proc. of IPTPS'05*, Ithaca, NY, February 2005.

[78] T. T. project FAQ. Why do we need polipo or privoxy with tor ? `https://wiki.torproject.org/noreply/TheOnionRouter/TorFAQ#WhydoweneedPolipoorPrivoxywithTor.3FWhichisbetter.3F`, accessed Apr, 2010.

[79] D. Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *Proc. ACM SIGCOMM'04*, Portland, Oregon, USA, Aug. 30–Sept. 3 2004.

[80] A. Rao, A. Legout, and W. Dabbous. Can realistic bittorrent experiments be performed on clusters ? In *Proc. of P2P'10*, Delft, Netherlands, August 25–27 2010.

[81] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous Connections and Onion Routing. *IEEE Journal on Selected Areas in Communications*, 16 :482–494, 1998.

[82] B. Ribeiro, W. Chen, G. Miklau, and D. Towsley. Analyzing privacy in enterprise packet trace anonymization. In *In Proceedings of the 15 th Network and Distributed Systems Security Symposium*, 2008.

[83] P. Rodriguez and E. W. Biersack. Dynamic parallel-access to replicated content in the internet. *IEEE/ACM Transactions on Networking*, 10(4), August 2002.

[84] J. Shneidman, D. Parkes, and L. Massoulie. Faithfulness in Internet Algorithms. In *Proc. of the Workshop on Practice and Theory of Incentives and Game Theory in Networked Systems (PINS'04)*, Portland, OR, September 2004.

[85] G. Siganos, J. Pujol, and P. Rodriguez. Monitoring the bittorrent monitors : A bird's eye view. In *Proc. of PAM'09*, Seoul, South Korea, April 2009.

[86] M. Sirivianos, J. H. Park, R. Chen, and X. Yang. Free-riding in BitTorrent Networks with the Large View Exploit. In *Proc. of IPTPS'07*, Bellevue, WA, February 2007.

[87] S. Wolchok and J. A. Halderman. Crawling BitTorrent DHTs for Fun and Profit. In *Proc. of WOOT*, Washington, DC, USA, 2010.

[88] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz. P4p : Provider portal for applications. In *Proc. of ACM SIGCOMM*, Seattle, WA, USA, August 2008.

[89] X. Yang and G. de Veciana. Service capacity in peer-to-peer networks. In *Proc. IEEE Infocom'04*, pages 1–11, Hong Kong, China, March 2004.

[90] C. Zhang, P. Dhungel, Z. Liu, and K. W. Ross. BitTorrent Darknets. In *Proc. of INFOCOM*, San Jose, CA, USA, 2010.

[91] C. Zhang, P. Dhungel, D. Wu, and K. W. Ross. Unraveling the bittorrent ecosystem. *Parallel and Distributed Systems, IEEE Transactions on*, 22(7) :1164 –1177, july 2011.

[92] H. Zhang and J. Bolot. Anonymization of location data does not work : A large-scale measurement study. In *MobiCom'11*, Las Vegas, Nevada, September 2011.

Résumé

Cette thèse présente deux axes de recherche que j'ai menés entre 2004 et 2011. Le premier axe est sur la compréhension sur protocole BitTorrent. Il recouvre des travaux expérimentaux et de mesures montrant l'efficacité de BitTorrent et expliquant les variations de performances observées dans la littérature et interprétées à tort comme un problème de BitTorrent. Une caractéristique de ces travaux est l'utilisation d'expérimentations à grande échelle (par exemple 10 000 pairs BitTorrent que l'on expérimente dans un environnement contrôlé) et de mesures à grande échelle (par exemple, 148 millions d'adresses IP mesurées sur 3 mois).

Le deuxième axe est sur l'étude de la protection de la vie privée sur Internet. Ces travaux montrent que l'on peut surveiller sans infrastructure dédiée (c'est-à-dire d'une seule machine) la quasi-intégralité des utilisateurs de BitTorrent. On montre également qu'en exploitant les communications pair-à-pair dans Skype on peut lier une identité sociale (nom, prénom, adresse email, etc.) à une activité sur Internet (par exemple, une liste de contenus téléchargés avec BitTorrent). On peut également suivre la mobilité des utilisateurs de Skype et on montre que cette mobilité est réelle pour un utilisateur standard. Pour finir, on montre qu'un réseau garantissant l'anonymat, comme Tor, n'offre pas une protection satisfaisante.