Distribution multi-contenus sur Internet

Imane Mnie Filali

UNIVERSITE DE NICE-SOPHIA ANTIPOLIS

ECOLE DOCTORALE STIC

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

T H E S E

pour l'obtention du grade de

Docteur en Sciences

de l'Université de Nice-Sophia Antipolis

Mention : Informatique

présentée et soutenue par

Imane MNIE FILALI

TITRE

DISTRIBUTION MULTI-CONTENUS SUR INTERNET

Thèse dirigée par le Pr. *Guillaume URVOY-KELLER*

soutenue le *27 Septembre 2016*

Jury :

Dr.	Soufiane ROUIBIA	Examinateur (co-encadrant)
Dr.	Dino PACHECO LOPEZ	Examinateur (co-encadrant)
Pr.	Jean-Pierre GUEDON	Examinateur
Pr.	André-Luc BEYLOT	Rapporteur
Dr.(HDR)	Benoît PARREIN	Rapporteur

Abstract

In this study, we focused on peer-to-peer protocols (P2P), which represent a promising solution for data dissemination and content delivery at low-cost in the Internet. We performed, initially, a behavioral study of various P2P protocols for file sharing (content distribution without time constraint) and live streaming.Concerning file sharing, we have shown the impact of Hadopi on users' behavior and discussed the effectiveness of protocols according to content type, based on users' choice. BitTorrent appeared as the most efficient approach during our study, especially when it comes to large content. As for streaming, we studied the quality of service of Sopcast, a live distribution network that accounts for more than 60% of P2P broadcast live events. Our in-depth analysis of these two distribution modes led us to focus on the BitTorrent protocol because of its proven efficiency in file sharing and the fact that it is open source. In the second part of the thesis, we proposed and implemented a new protocol based on BitTorrent, in a controlled environment. The modifications that we proposed allow to increase the efficiency of the protocol through improved dissemination of metadata (the rarest piece), both for live and file sharing. An enhanced version is introduced with a push method, where nodes that lag behind receive an extra service so as to improve the overall performance.

Résumé

Dans cette thèse, nous nous sommes intéressés aux protocoles pair-à-pair (P2P), qui représentent une solution prometteuse pour la diffusion et le partage de données à faible coût sur Internet. Nous avons mené, dans un premier temps, une étude comportementale de différents protocoles P2P pour le partage de fichier (distribution de contenus sans contrainte de temps) puis le live. Dans la première étude centrée sur le partage de fichier, nous avons montré l'impact d'Hadopi sur le comportement des utilisateurs et discuté l'efficacité des protocoles en fonction du contenu et l'efficacité protocolaire, en se basant sur les choix des utilisateurs. BitTorrent s'est nettement démarqué au cours de cette étude, notamment pour les grands contenus. En ce qui concerne le live, nous nous sommes intéressés à la qualité de service du réseau de distribution live Sopcast, car plus de 60% des événements live diffusés en P2P le sont sur ce réseau. Notre analyse approfondie de ces deux modes de distribution nous a fait nous recentrer sur BitTorrent, qui est à la base de tous les protocoles P2P Live, et est efficace en partage de fichier et complètement open source. Dans la seconde partie de la thèse, nous avons proposé et implémenté dans un environnement contrôlé un nouveau protocole sur la base de BitTorrent avec des mécanismes protocolaires impliquant tous les pairs dans la gestion du réseau. Ces nouveaux mécanismes permettent d'augmenter l'efficacité du protocole via une meilleure diffusion, tant pour le live que le partage de fichier, de méta-données (la pièce la plus rare) et via une méthode dite de push, par laquelle un client va envoyer du contenu aux pairs les plus dans le besoin.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

L'intérêt envers les réseaux pair à pair [1] ne cesse de croître après plus de dix ans d'existence. Le partage rapide à coût faible et efficace est leur plus grand avantage. C'est une approche réseau où tous les utilisateurs se comportent en client et serveur, selon la nature de la requête qu'ils reçoivent. Ce double comportement est la clef du passage à l'échelle du service offert.

La démocratisation de l'Internet a vu le jour vers la fin des années 90 avec l'apparition du premier navigateur web Mosaic [1] développé par la NCSA[2] en 1993. L'année qui suit, cette équipe a rejoint Netscape communications corporation[3] pour introduire Netscape[4]. La même année a témoigné la création du W3C [5]. Le monde a assisté à une croissance exponentielle en terme de nombre de serveurs qui est passé de 1,3 millions en 1993 à plus de 137 millions de serveurs en 1996 et de 16 millions d'internautes au début de 1996[6] à plus de 513 millions. Cette croissance était conconmittante avec l'apparition du premier réseau P2P, Napster[7] en 1999 suivi de iMesh[8] , Freenet[9] et de Gnutella[59] en 2000 (Figure 1.1).

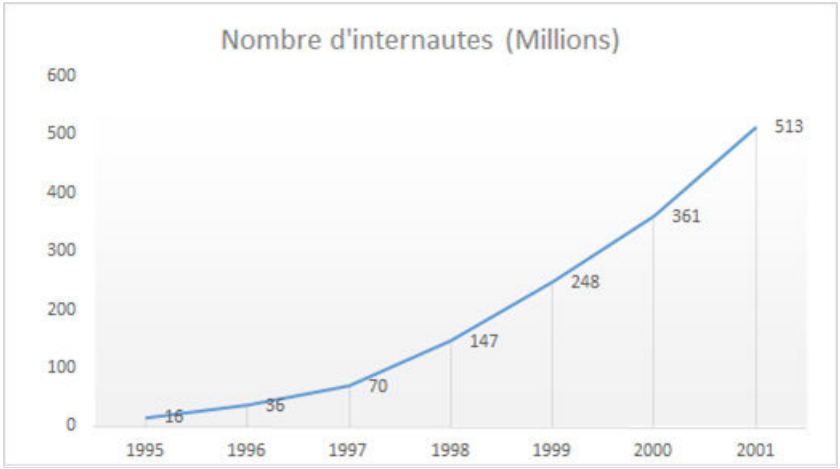

FIGURE 1.1 – Utilisateurs de l'internet entre 1996 et 2001 (source : Internet World Stats)

L'arrivée de ces réseaux P2P a contribué fortement à maintenir cette croissance durant les années 2000 : des études ont en effet montré que le trafic dominant à

1. Nous utiliserons principalement dans ce document l'abbréviation anglaise P2P, *Peer to Peer*.

cette époque sur Internet est le P2P[56] et qu'à lui seul, BitTorrent[**bittorrent**] représentait près de la moitié de ce trafic [49] .

La clé du succès des systèmes P2P est la scalabilité (passage à l'échelle). Leur architecture évolutive permet la réorganisation du réseau à l'ajout de chaque nouveau pair. Au contraire de l'architecture client/serveur les performances d'un réseau P2P doivent croître (linéairement ou non) avec l'augmentation du nombre des pairs présents sur le réseau. Ainsi la disponibilité du contenu par réplication croît avec le nombre de pairs (connectés) actifs.

Dans la suite de ce chapitre, nous allons introduire des notions de base sur l'architecture des réseaux P2P, puis faire un tour d'horizon des principaux protocols P2P, et enfin, nous ferons un état de l'art scientifique en rapport avec les contributions techniques de cette thèse.

1.1 Architectures des Réseaux P2P

Les systèmes P2P peuvent être classés selon l'aspect fonctionnel (présence de serveur) qui va définir explicitement le modèle de propagation de données. Le mode de propagation de données peut être soit non structuré, les pairs sont interconnectés de manière aléatoire, soit d'une manière structurée, basée sur une organisation sous forme d'arbre ou au travers d'une fonction de hachage. Cette dernière approche offre moins de flexibilité au réseau mais promet un routage des requêtes efficace. Nous aloons introduire dans cette section les différentes architecture des réseaux P2P.

P2P Centralisé

L'architecture centralisée rappelle celle du client/serveur (voir Figure 1.2). Les clients d'un système P2P centralisé doivent s'authentifier auprès d'un serveur qui stocke leurs informations et qui représente la seule "porte d'entrée" au réseau. Ceci ne garantit aucun anonymat puisque les clients peuvent être identifiés par leurs adresses IP et les fichiers qu'ils partagent. Bien que ce type d'architecture permette une localisation rapide des données, un serveur central représente souvent des problèmes de congestion et un point de panne unique (*single point of failure*), ce qui signifie le dysfonctionnement de tout le réseau en cas de problèmes techniques ou judiciaires. Napster est un exemple de cette architecture (victime de problèmes judiciaires plus que techniques).

Hybride P2P

Le modèle hybride (voir Figure 1.3) est un mélange entre architecture centralisée (nécessitant la présence d'un serveur) et architecture décentralisée. Le réseau est organisé d'une manière structurée, en général sous forme d'arbre et les échanges se font par propagation à partir des nœuds parents aux nœuds fils [78]. Les parents

FIGURE 1.2 – Architecture centralisée
1 : Authentification ; 2 : Initiation ; 3 : Début d'échange

envoient des données aux nœuds fils d'une manière proactive, ce qui présente des risques d'inondation : un nœud peut avoir plusieurs parents et recevra la même donnée plusieurs fois ou la donnée peut ne jamais être reçue.

Pure P2P

Le modèle *pure P2P* ou modèle décentralisé ne s'appuie pas sur la présence de serveurs. Chaque client devient automatiquement client et serveur. C'est une organisation aléatoire ou un nœud, d'une manière spontanée, envoie des requêtes aux nœuds possédant le contenu, les *seeds*, pour recevoir les données dont il a besoin (on parle de *chunks* ou morceau de contenu). Guntella est le premier système qui introduit cette architecture [80].

1.2 Les protocoles P2P

La figure 1.5 récapitule les principales familles d'applications P2P avec leurs représentants les plus populaires.

A ces trois familles emblématiques, nous ajoutons deux nouvelles applications, la messagerie instantanée et la sauvegarde de fichier.

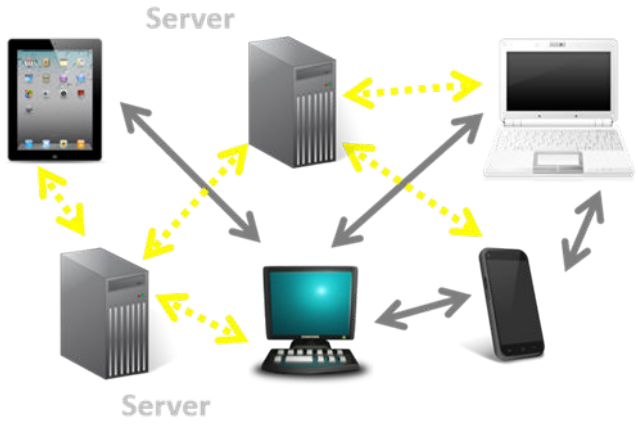

FIGURE 1.3 – Architecture Hybride

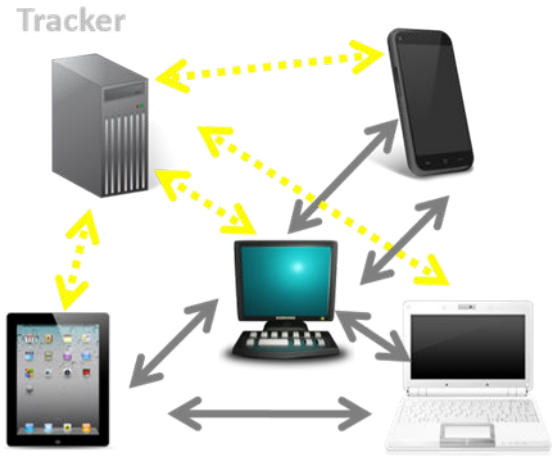

FIGURE 1.4 – Architecture pure P2P

1.2.1 Partage/Replication de fichier

La partage de fichiers est la première application populaire qui a vu le jour avec Napster puis eDonkey ou BitTotrrent. Un utilisateur qui veut partager un contenu le rend accessible en proposant un fichier torrent (pour les utilisateurs de BitTorrent par exemple) ou en se connectant sur une application (telle que eMule [10] ou Shareaza [11] pour les réseaux eDonkey ou Gnutella) qui propose un outil de recherche et rend une partie de l'espace de stockage personnel accessible aux autres utilisateurs connectés. BitTorrent se concentre en fait sur la replication et non sur la recherche du contenu. L'utilisateur de BitTorrent doit chercher un fichier torrent correspondant au contenu sur des sites web qui font objet d'annuaire.

Dans tous les cas, un contenu à échanger/répliquer est découpé en parties que nous nommons dans ce manuscrit pièces ou *chunks*.

Pour le choix du pair à servir, le protocole BitTorrent axe sa politique sur les pairs puissants en bande passante pour assurer la survie du réseau avec un brin

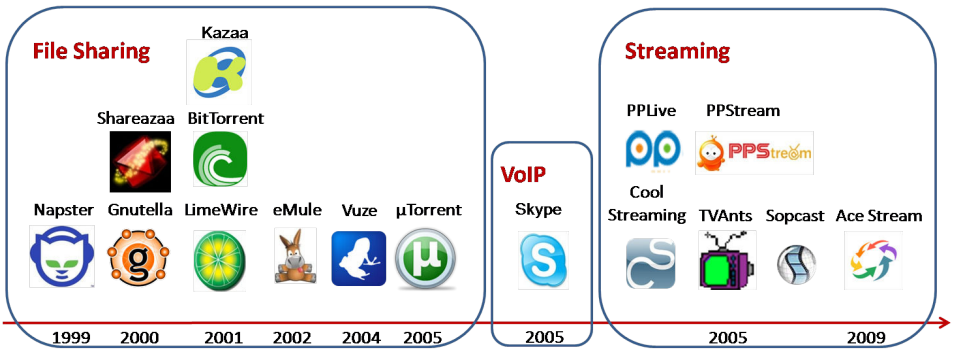

FIGURE 1.5 – Applications P2P

d'aléatoire dans le choix de la première liste envoyée par le *tracker*, le seul élément centralisé de BitTorrent qui permet aux pairs de se rencontrer. Pour les autres réseaux, c'est un fonctionnement par liste d'attente avec un coefficient d'ancienneté dans le cas du protocole eDonkey.

Des améliorations à BitTorrent sont proposées dans les chapitres 4 et 5. Nous introduisons donc ci-après des éléments du protocole que l'on utilisera dans cette thèse.

— **Le Peer exchange :** ou PeX, est l'échange entre les pairs du réseau de leurs listes de pairs dans le but de trouver le pair le mieux adapté à servir. Cette technique permet d'utiliser la bande passante des pairs qui téléchargent pour la recherche de pairs et d'information et de contenus présents dans le réseau au lieu d'utiliser la bande passante du serveur.

— **Le Have :** message qui indique que le client a fini de télécharger une pièce du contenu, l'a vérifié via son *hash* (le fichier torrent contient un code de hachage pour chaque pièce) et l'annonce à tous les autres pairs à qu'il est connecté. Ceci permet d'avoir une vision locale, limitée au voisinage, de l'état d'avancement des pairs voisins dans le téléchargement. En même temps, les pairs peuvent savoir envers qui s'orienter pour demander les pièces qui les intéressent. Cela permet également de calculer la pièce la plus rare dans le voisinage et la demander par la suite.

— **Interested :** en réponse au Have, les clients envoient un message interested pour signaler leur intérêt à son annonce.

— **Unchoke :** en réponse à Interested, si le pair accepte de partager la pièce il envoie un message unchocke. Dans le cas contraire il envoie un choke et met le pair en attente.

— **Le Bitfield :** une chaine de bits désignant la possession ou pas d'une pièce.

1.2.2 Live / Streaming

Le principe du Live est la diffusion en temps réel d'un contenu qui est généralement une vidéo partagée par une source. La vidéo qui est un stream direct ou une

diffusion d'un programme télévisé (émission, Match de foot, spectacle ...) est généralement découpée en morceaux (video chunks) qui seront par la suite transportés par le protocole TCP ou UDP. L'UDP étant préférable pour controller le débit et la gigue et TCP étant préférable pour traverser les pare-feux. Les pairs peuvent accéder aux chunks soit via la source primaire, soit aux travers d'autres pairs, créant ainsi un réseau de partage. Plus ils sont nombreux plus cela permet d'alléger la charge de la source. Cette forme de partage optimise aussi l'utilisation du tampon car cela ne nécessite moins de stockage en mémoire.

Les protocoles de streaming live les plus populaires sont Sopcast, TVants, PPlive. Une caractéristique importante est qu'ils sont propriétaires. Néanmoins, de nombreuses études (voir par exemple le cas de Sopcast section 1.4.2) ont montré que ces protocoles employaient des algorithmes similaires à BitTorrent. Cela justifie dans les chapitres 4 et 5, que nous utilisions BitTorrent pour tester des transferts en considérant des tailles de contenu faible, correpondant à une fenêtre de temps utilisées dans les protocoles de streaming.

Notre étude de Sopcast est aussi motivée par l'appétit des utilisateurs pour des (i) contenus de plus en plus grande qualité (d'encodage) ce qui (voir notre Chapitre 2) et (ii) la popularité grandissante des diffusions live sur Internet , qui met les protocoles de streaming P2P sous pression.

1.2.3 VoIP P2P

La voix sur IP ou VoIP (*Voice over IP*) est une technologie pour la communication par la voix sur des réseaux IP au lieu de passer par les réseaux téléphoniques commutés publics (RTC). Cette technique permet la convergence de la communication voix, données et vidéo sur un réseau unique. Cette mutualisation voix et données réduit les coûts d'investissement dans l'infrastructure du réseau et facilite la configuration ainsi que l'assistance. En entreprise par exemple, mettre en place une telle solution permet de souscrire chez un seul fournisseur d'accès, avec un seul contrat de maintenance, administrer un seul réseau, un câblage commun tout en gardant l'infrastructure existante. Ces réductions seront encore plus intéressantes dans le cas de plusieurs sites distants. Initialement, cette technologie reposait sur principalement deux protocoles : le protocole SIP (session initial Protocol) qui permet l'authentification et la localisation des participants à la session de communication et le protocole RTP (Real time Transport Protocol) qui permet le transport des données soumises à des contraintes temps réel et donc assure la session audio (voix) et vidéo. Contrairement aux communications téléphoniques « traditionnelles » qui ont un circuit de bout en bout dédié et établie pendant la durée de la communication, la transmission des paquets IP se fait sur un réseau partagé. Ces derniers peuvent emprunter un chemin distinct déterminé par la table de routage qui dépend du trafic réseau et ceci indépendamment de l'ordre où les paquets doivent être mis

en tampon. Il faut donc gérer ce problème au niveau applicatif ou se baser sur TCP, bien que ce dernier ne permette pas un contrôle parfait de la séquence d'envoi.

Plusieurs projets VoIP basés sur le P2P ont mis en évidence le succès d'une telle technologie : Skype [12] qui existe depuis 2003 (et que nous détaillons ci-après) et qui compte un nombre total d'utilisateurs qui s'élève à 74 millions [13] en juin 2015. Depuis 2014, un projet baptisé Tox [14] introduit par *google code of summer*[15] commence à gagner du terrain. Il a été présenté comme le remplaçant de Skype en proposant une communication chiffrée et sécurisé. Tox utilise une DHT et un routage en oignon [16] *Tor*[53] .

Skype : Skype est un client VoIP développé initialement par KaZaa en 2003. Il opère sur un modèle P2P ce qui permet une scalabilité sans avoir à investir dans de l'infrastructure centralisée. Skype permet de passer des appels audio et vidéo depuis 2006 ainsi que des services comme le partage de fichiers, la messagerie instantanée et le partage d'écran. Selon leurs ressources (CPU, mémoire, bande passante, ...) les utilisateurs peuvent profiter des services offerts seulement ou y contribuer en devenant un super nœud ou un point de terminaison dans le réseau Skype. Les utilisateurs s'y connectent avec un login et un mot de passe qui seront enregistrés sur un serveur qui représente le seul point centralisé du réseau. Après cette étape, toutes les informations utilisateurs et données sont transmises d'une manière décentralisée. Les nœuds sont organisés par la suite en maille autour des super nœuds [43] et un protocole de découverte derrière les retours NAT est mis en place pour pouvoir échanger en UDP [81]. La popularité de Skype vient, au delà de sa facilité d'utilisation, de son algorithme de cryptage robuste qui permet la protection de la confidentialité des appels sur VoIP, le codec utilisé et la sélection du chemin de routage dynamique [85].

1.2.4 Messagerie instantanée

L'idée d'une messagerie instantanée sur un réseau P2P vient du besoin de protéger son identité. Plusieurs projets ont vu le jour dans ce sens permettant d'envoyer des messages sans qu'ils soient stockés au niveau d'une troisième entité autre que le destinataire et le récepteur. Bleep est l'un de ces projets, proposé par BitTorrent en 2015. Cette application est disponible pour les différents systèmes d'exploitation, mac et Windows ainsi que iOS et Android. Les messages sont cryptés et stockés en local sans passer par des serveurs. L'authentification se fait par pseudonyme sans fournir une adresse mail ou un numéro de téléphone. L'utilisateur doit fournir sa clé privée à la personne avec qui il veut échanger ou il a la possibilité de faire scanner son propre code QR si cette dernière se trouve à proximité. Le mode whisper est proposé permettant la disparition du message après lecture. L'application dans sa nouvelle version propose les appels directs chiffrés de bout en bout ainsi que des options comme le partage des photos et l'envoi de messages vocaux. Dans le même sens, une messagerie a été proposée par PeerSafe [17] baptisée ShadowTalk [18]

, Serval Chat [19] par Serval Project[20] (pour iOS seulement), FireChat [21] par OpenGarden [22] et Tuse [23] par Tuse [24].

1.2.5 Système de sauvegarde distribuée

Les systèmes de sauvegarde distribuées en P2P sont une alternative aux systèmes centralisés tels que Dropbox ou Google Drive. Ces systèmes représentent un point de défaillance unique, inconvénient de tout système centralisé et les utilisateurs en acceptant les termes d'utilisation autorisent inconsciemment que leurs données ne soient plus confidentielles. Les systèmes de sauvegarde distribuée reposent sur la coopération des utilisateurs qui mettent à disposition une partie de leur espace de stockage. Une approche proposée en 2007 est apparue sous le nom de MaidSafe [25, 60]. MaidSafe vise à maintenir la confidentialité et à assurer la protection d'identité ainsi que les données personnelles des utilisateurs sur Internet. L'idée est d'héberger chez chaque utilisateur des morceaux de données qui seuls, n'ont pas de sens, mais rassemblés reprennent leur sens. En échange, les utilisateurs auront des Safecoin[26] la version MaidSafe du Bitcoin[73] et leur permettra de maximiser les ressources (espace disque, mémoire, bande passante. . .) et les partager entre eux. Une protection des données est assurée par cryptage et dispersion, la disponibilité est assurée par réplication.

Dans le même ordre d'idées, d'autres projets existent comme PeerStore[63] ou Pastiche[48]. Des études[72] ont montré que pour maintenir le principe d'égalité entre pairs les deux systèmes essayent d'équilibrer entre ce que consomment les pairs en espace disque et ce qu'ils proposent ; cependant le problème des *free-riders* (consommateurs ne partageant pas leurs ressources) persistent.

Synchronisation de fichier

Un concept relativement similaire au précédent est la synchronisation des fichiers. L'objectif est de pouvoir synchroniser les travaux ou les opérations effectuées sur un ou plusieurs fichiers par une même personne ou un groupe d'utilisateurs indépendamment de leur localisation ou le terminal sur lequel ils travaillent. Ces solutions permettent de collaborer sur un même projet sans restrictions (impératifs) géographiques ni temporelles. Nous pouvons imaginer des projets qui nécessitent l'expertise de plusieurs personnes : élaborer des bases de données par des communautés scientifiques, établir des diagnostiques par plusieurs médecins qui travaillent sur un même sujet, développer une application par plusieurs développeurs, rédactions d'articles scientifiques par plusieurs chercheurs . . . Il existe plusieurs solutions client/ serveur dans ce sens, comme Dropbox, cité dans la section précédente, qui offrent aux utilisateurs le partage de fichiers ainsi que la synchronisation avec des accès en lecture et écriture selon les droits attribués à chacun.
TeamDrive[27] propose aussi une solution de synchronisation de fichier orienté travail collaboratif en entreprise. L'équipe a accentué l'aspect sécurité en utilisant

une clé 2048-bit RSA générée par la librairie cryptographique OpenSSL. TeamDrive propose aussi un service de sauvegarde et de partage de données. Plusieurs autres solutions orientées entreprise existent aussi comme Tresorit[28]qui utilise un chiffrement de bout à bout (données chiffrées avant d'arriver sur le serveur), Mediafire[29] ,Engnyte[30], SugarSync[31]. Le problème commun de ces systèmes réside dans la présence d'un serveur central. De plus ils ne sont en général pas open-source. Si les utilisateurs ne prévoient pas une couche sécurité, les fournisseurs des services de synchronisation ont pleinement accès aux données clients.

D'autres services plus commerciaux/ populaires comme MicroSoft One Drive [32], Google Drive [33] ou encore iCloud d'Apple [34] sont dédiés à la synchronisation de fichier mais ce processus entre les machines des utilisateurs crée une copie sur le cloud avant d'arriver à destination. La sécurité des données autour de ses services n'est pas complétement transparente dans l'absence de garantie de confidentialité des données. Le Google cloud par exemple, fournit un chiffrement coté serveur en utilisant une clé *128-bit Advanced Encryption Standard*. Lors de la synchronisation, Google se charge de crypter et controle parallèlement les fichiers, ce qui signifie que lors de ce processus les données sont toujours lisibles si besoin. Du coté de One Drive, la charte d'utilisation certifie que seul l'utilisateur est propriètaire de son contenu. Néanmoins, Microsoft s'autorise dans ses conditions générales d'utilisation[35] à collecter, utiliser, modifier, adapter, reproduire et divulguer les contenus et les données de ses utilisateurs concernant tous ses services. Par ailleurs, Google, Microsoft et Dropbox sont des sociétés américaines soumises aux législations en vigueur aux états-unis ce qui permet à des agences comme la NSA, CIA et FBI de demander l'exploitation de l'ensemble des contenus partagés en s'appuyant sur le Patriot Act[61] ou l'Electronic Communications Privacy Act [70] par exemple.
Des alternatives existent en pair-à-pair pour contourner les solutions basées sur un serveur tiers et ou un cloud, notamment BitTorrent Sync.

BitTorrent Sync

BitTorrent propose une alternative aux solutions C/S et Cloud nommée BTSync ou simplement Sync [36]. Le service est basé sur le protocole de partage BitTorrent permettant de contourner la centralisation des données. Sync propose en outre, une option DHT pour éviter de s'authentifier au niveau du tracker. Contrairement aux offres payantes présentées précédemment, Sync ne limite pas ses utilisateurs à un certain volume de stockage mais au propre volume de stockage des pairs synchronisés et leur bande passante. Une option chiffrement RSA est proposée dans le mode de transmission entre réseaux locaux. Cependant le protocole n'est pas open-source.

1.3 Plan de la thèse

Après ce rapide tour d'horizon et avant de donner des éléments d'état de l'art technique, nous allons dresser le plan global de la thèse en terme de contributions. Notre objectif global est de proposer de nouvelles méthodes de distributions de contenus sur Internet, principalement au travers de l'améliorations de protocoles existants.

— Nos contributions concernent les mesures de réseaux P2P existant ou la modification et l'évaluation dans un environnement réalistes de protocoles existants.

— Les chapitres 2 et 3 appartiennent à la première catégorie. Partant de la maitrise de la mesure des réseaux P2P chez TMG, nous nous sommes tout d'abord attachés à comprendre, au travers de données historiques, quelles sont les protocoles P2P plébiscités par les utilisateurs. Cette étude, qui forme le chapitre 2, montre d'une part l'impact de loi Hadopi sur les utilisateurs, mais également leur appétit pour des contenus de plus en plus haute qualité (en terme d'encodage). Ce dernier est à mettre en lien avec le protocole BitTorrent qui semble le plus efficace lorsque le contenu à transmettre devient grand, face à son concurrent principal, eDonkey/eMule/aMule.

— Toujours dans un contexte de mesures, nous nous sommes attachés à l'étude de Sopcast (chapitre 3), un protocole de streaming P2P comme expliqué précédemment. Ce protocole est devenu populaire (à nouveau, nous le savons au travers des mesures effectuées chez TMG) notamment dans la diffusion d'évènements sportifs. Notre objectif était de comprendre si cette popularité était due à un plus grande efficacité dans l'algorithme de choix des pairs, qui venait de changer concomitamment au début de la thèse. Cette étude, qui est présentée chapitre 3 montre la difficulté d'une telle évaluation lorsque le protocole est propriétaire et que les mesures doivent être faites in vivo sur Internet avec une reproducibilité limitée. Nous avons donc modifier notre angle d'attaque pour la fin de la thèse en repartant d'un protocole populaire open source : BitTorrent.

— Dans les chapitres 4 et 5, nous nous attachons à construire un environnement de mesure de performance adéquat et à proposer des modifications de certains protocoles de BiTorrent afin d'obtenir des améliorations de performance dans un cas live ou enregistré. En particulier, nous étudions une technique pour mieux disséminer dans le réseau l'information sur la pièce la plus rare chapitre 4 et nous étudions une approche où un pair va pousser du contenu dans le réseau dès qu'il juge qu'un pair peu en bénéficier chapitre 5. Dans les 2 cas, nous montrons que nous pouvons améliorer les performances du protocole.

1.4 Etat de l'art scientifique

Dans cette section nous présentons les études en lien avec nos différentes contributions techniques, à raison d'une sous section distincte par partie.

1.4.1 Travaux connexes mesure

Plusieurs études ont examiné l'effet des lois anti-piratage sur le comportement des pirates.[50] a mis l'accent sur l'aspect commercial et a démontré que la vente des offres légales françaises (sur iTunes, pour le cas de 4 grands labels) sont plus importantes que dans d'autres pays européens après l'adoption de la loi Hadopi. En 2010, un an après l'adoption de Hadopi, les auteurs de [51] ont présenté une analyse statistique basée sur un échantillon de 2000 utilisateurs français et ont suggéré que si certains pirates ne réduisent pas leurs activités illégales, d'autres sont passés à d'autres formes de techniques de piratage (streaming ou téléchargement direct). Par conséquent, dans les deux cas, ils continuent de télécharger du contenu illégal. En 2014, les auteurs de [52] ont constaté que la crainte de l'Hadopi joue un rôle significatif dans la structuration de l'offline swapping (réseau non surveillé) par exemple, en utilisant une clé USB. Ils ont également suggéré que seuls les "amateurs de risque" continuent à télécharger illégalement à travers des réseaux P2P. En 2013, les auteurs de [76] mettent l'accent sur l'impact de la fermeture de Megaupload sur un point de vue spécifique : l'activité des utilisateurs qui partagent des contenus sur BitTorrent. Ils démontrent que cet événement des lois anti-piratage dans le monde entier a déclenché une chute immédiate de l'activité des utilisateurs qui partagent sur BitTorrent, spécialement ceux qui ne partagent qu'occasionnellement contrairement aux « habitués ». Dans [42], les auteurs ont analysé l'impact de la loi Hadopi sur les individus utilisant un modèle économétrique théorique. Leur modèle suggère que la loi Hadopi n'a aucun effet dissuasif.

1.4.2 Travaux Sopcast

Dans [69], Marfia déchiffre les principaux paramètres de conception de Sopcast, PPLive, CoolStreaming et Anysee, en se basant sur des mesures de trafic à partir de ces applications. Les paramètres étudiés comprenaient la formation de la topologie de l'overlay, la taille du buffer et le délai de lecture. Ils proposent de séparer entre les paquets de contrôle et de données en fonction de la taille du paquet. Une approche similaire est utilisée dans toutes les études ultérieures, y compris la nôtre. Pour le cas de Sopcast, ils ont observé que le nombre de pairs à partir duquel un nœud peut télécharger des paquets varie généralement entre 2 à 5 pairs, tandis que le nombre de voisins peut être beaucoup plus élevé.

Dans [58], les auteurs ont analysé PPLive, TVants et Sopcast et ont constaté que ce dernier ne favorise pas les pairs avec plus de bande passante dans le processus de téléchargement. En outre, ils ont observé que TVants favorise les pairs proches pour

former le réseau, contrairement aux deux autres protocoles qui semblent choisir les voisins au hasard. Dans la phase de transfert de données, les deux TVants et PPLive ont tendance à favoriser les échanges de données entre les pairs proches (même AS) contrairement à Sopcast.

Dans [71], les auteurs ont effectué en 2009 des mesures par le biais d'un seul hôte connecté par une ligne ADSL avec respectivement 12 Mb/s et 512 kb/s en débit descendant et débit montant. Ils se sont concentrés sur trois protocoles de streaming P2P : TVAnts, Sopcast et TVU Player2. Dans leur expérimentation, leur pair est resté connecté pendant plusieurs heures sur quelques chaînes. Ils soutiennent que Sopcast ne télécharge pas les chunks seulement des pairs ayant une bonne bande passante puisqu'ils téléchargent plus de 10GB avec leur modeste 512 k /s en capacité de téléchargement. Cependant, Sopcast a enregistré la plus haute performance dans le sens ou l'évolution du taux de téléchargement marque la plus basse courbe en pente, ce qui signifie moins d'images figées. Globalement, ils concluent que ces trois applications n'exploitent aucune localisation pour réduire la charge du trafic, ce qui contredit les résultats fournis par [58] en 2009 (un an avant).

Dans [44] les auteurs se sont focalisés seulement sur les performances de Sopcast. Contrairement aux études précédentes, où quelques clients P2PTV étaient sous le contrôle des expérimentateurs et reliés à des chaînes bien choisies, ils ont observé Sopcast en utilisant une approche passive en s'appuyant sur des collectes d'un ISP européen. Ils ont aussi détecté que Sopcast était le plus populaire parmi toutes les applications, avec PPLive et TVants. Ils ont observé que les pairs avec les débits montants les plus élevés présentaient un taux de découverte de pairs beaucoup plus élevé, ils ont conclu que le taux de découverte dépend du débit montant des pairs. Ils ont affirmé que le mécanisme de découverte de pairs suit un processus aléatoire dans lequel la probabilité de contacter

1.4.3 Travaux BitTorrent

La réplication de contenu dans une optique temps-réel (live) ou non a suscité de nombreux travaux. Si on prend le cas de BitTorrent [46], de nombreuses études ont questionné tel ou tel choix du protocole [83]. On peut citer par exemple le choix du nombre de pairs vers lesquels téléchargés ou la répartition du débit montant entre clients [64, 45]. L'impact de la taille des pièces a été analysé dans [**legout08**], où il est montré, dans le cas de grands contenus (cas file sharing de nos expérimentations), l'utilisation de pièces trop petites pouvait avoir un effet délétère sur les performances.

Chapitre 2

Utilisation des protocoles P2P en France

Dans ce chapitre nous allons nous concentrer sur les protocoles P2P les plus utilisés (voir figure 2.1) en s'appuyant sur des données collectées par la plateforme TMG (voir section 2.4).

Notre objectif est de montrer l'impact d'Hadopi [1] sur le comportement des utilisateurs mais également de comprendre quels sont les protocoles efficaces (temps de téléchargement voire d'autres métriques comme la discrétion), qui devraient être ceux que les utilisateurs plébiscitent.

La figure 2.1 est générée sur la base d'IPs localisées en France et les contenus sont protégés par des droits d'auteur. Nous sommes sur du téléchargement illégal. A partir de la figure 2.1, nous pouvons constater que la répartition des clients P2P utilisés est fortement biaisée par deux protocoles dominants à savoir eDonkey/ Kad et BitTorrent (en termes de nombre de clients actifs détectés), suivis par des protocoles moins populaires avec au moins un ordre de grandeur moins d'IPs par rapport aux IP clients observées sur les réseaux précédents.

Nous pouvons distinguer deux principales familles de protocoles P2P : les protocoles de réplication et les protocoles de partage de fichiers. Nous trouvons ainsi d'une part, les protocoles de réplication de fichier représentés par un seul membre sur notre plateforme, BitTorrent (mais plusieurs clients comme uTorrent [2], BitTorrent [3], Vuze (autrefois Azureus) [4], Transmission [5], BitComet [6] et Libtorrent [7]) où les clients coopèrent temporairement avec le seul objectif de reproduire au plus vite un contenu spécifique. La recherche du contenu se fait habituellement sur des sites web qui proposent des fichiers de type torrent comme The pirate bay (`http://thepiratebay.se/`), kickasstorrents (`https://kat.cr/`), cpasbien (`http://www.cpasbien.cm/`) et T411 (`http://www.t411.ch/`).

1. https ://www.hadopi.fr/la-haute-autorite/lhadopi-en-bref
2. http ://www.utorrent.com/
3. http ://www.bittorrent.com
4. http ://www.vuze.com/
5. https ://www.transmissionbt.com/
6. http ://www.bitcomet.com/
7. http ://www.libtorrent.org

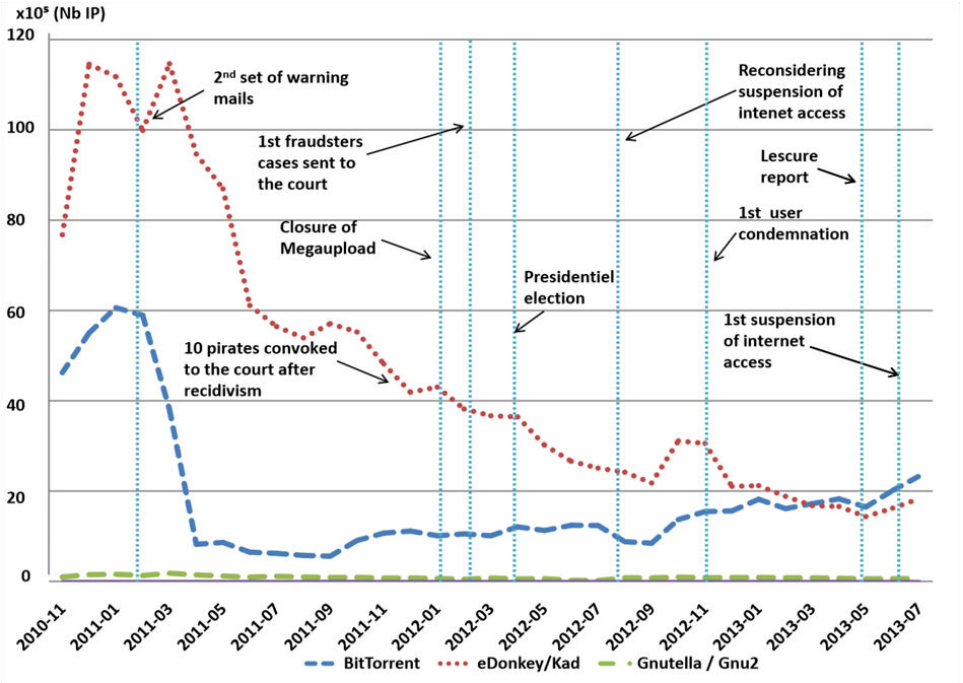

FIGURE 2.1 – Impact de la loi Hadopi sur l'usage du P2P : dates importantes

D'autre part, nous observons du eDonkey/ Kad (eMule [8], aMule [9]) et Gnutella / Gnutella2 (Wireshare vesrion pirate de LimeWire [10], Shareaza [11]) et Ares [12] qui appartiennent à la famille des applications de partage de fichier où l'accent est mis sur l'efficacité de la recherche d'un ensemble de serveurs, pour tout contenu disponible sur le réseau. Les applications de partage de fichier diffèrent dans la façon dont leur réseau est organisé. Gnutella fait usage d'une approche dépassée basée sur l'inondation pour rechercher un contenu. eDonkey, Gnutella2 et Ares sont également des réseaux non structurés (absence de DHT - Distributed Hash Table) qui utilisent une organisation hiérarchique avec la notion de super-pairs / hubs auxquels les clients se connectent. En revanche, Kad utilise le protocole Kadmelia où une DHT est utilisée pour trouver les clients ainsi que les contenus. Une telle approche permet de contourner le problème de la congestion rencontrée avec les techniques de recherche classiques dans d'autres protocoles. Les réseaux eDonkey et Kad sont étroitement liés étant donné que les clients utilisent généralement le réseau eDonkey pour démarrer (initier) leur connexion au réseau Kad. Le seul orphelin dans les clients observés est Shareaza qui opère sur les réseaux eDonkey et BitTorrent simultanément. Notez que pendant les deux ans et demi d'observation, seulement 3,1% des clients P2P n'étaient pas correctement identifiés par notre plateforme.

8. http ://emule-project.net
9. http ://www.amule.org
10. http ://sourceforge.net/projects/wireshare
11. http ://shareaza.sourceforge.net/
12. http ://aresgalaxy.sourceforge.net/

L'adoption d'un réseau P2P spécifique par l'utilisateur est liée à plusieurs facteurs. Parmi ceux-ci, la disponibilité du contenu qu'il recherche et l'efficacité du protocole lors de la phase de téléchargement. En effet ces deux facteurs sont liés, si le téléchargement d'un contenu est trop lent avec un client spécifique, cela va limiter le réseau P2P au partage de contenus de petite taille. Comme nous le verrons dans ce qui suit, un tel facteur explique pourquoi eDonkey / Kad est principalement spécialisé dans le contenu de type musique pendant que BitTorrent est spécialisée dans la distribution des contenus de grande taille comme la vidéo. Un dernier facteur à prendre en compte, du point de vue utilisateur, est le niveau d'exposition : sur les réseaux de partage, le client doit partager avec les autres utilisateurs le contenu d'un dossier spécifique. En revanche, avec la réplication de fichier, le client a besoin de partager seulement le contenu qui est en cours de téléchargement. Dans le contexte de la loi contre le piratage, cette différence peut être très importante et décisive du point de vue de l'utilisateur.

Ce chapitre sera organisé comme suit. Nous donnerons tout d'abord un aperçu du paysage législatif en terme de lois anti-piratage en France et dans le monde en section 2.1. La plateforme utilisée est détaillée section 2.4. Une analyse détaillée des résultats est fournie section 2.5.

2.1 Le Partage de fichiers et la protection des droits d'auteurs

La distribution de contenu audio et vidéo sur Internet se fait, en plus des réseaux pair-à-pairs, au travers de sites web complétés par des CDN (content distribution network). Ces contenus-là peuvent être protégés par des droits d'auteur. Or, la protection de la propriété intellectuelle sur Internet est moins fréquente que dans le monde "physique". En effet, aujourd'hui plusieurs pays ne disposent d'aucune loi interdisant la distribution illégale de données protégées. Seuls quelques pays, comme l'Angleterre, la Nouvelle-Zélande, la France ou les Etats-Unis ont des lois pour protéger la propriété intellectuelle sur Internet. Les réseaux P2P sont fréquemment utilisés pour partager du contenu illégalement. De ce fait nous présentons dans cette section une analyse faite sur l'évolution du trafic P2P illégal sur deux ans et demi, et comment un tel trafic a été touché par l'adoption de lois de protection des droits d'auteur sur Internet. Nous nous concentrons sur le cas français. Nous verrons ainsi comment les protocoles de distribution P2P dominants ont été affectés de manière différente, selon le type de contenu qu'ils proposent et la furtivité offerte à l'utilisateur.

2.2 Les lois anti-piraterie autour du monde

À la fin des années 2000, certains pays, comme la France (nous détaillons le cas français dans la section 2.3), la Nouvelle-Zélande et le Royaume-Uni ont adopté des

lois basées sur la méthode de " la réponse graduée" (three strikes anti-piracy). Ces lois stipulent que les contrevenants doivent être d'abord mis en garde à plusieurs reprises s'ils téléchargent du contenu illégalement, et en cas de récidive, ils devraient être temporairement déconnectés de l'Internet ou recevoir des amendes. En vertu de la Loi néo-zélandaise [37] (Amendement Act 2011) , les utilisateurs qui partagent des fichiers reçoivent des avertissements au cours d'une "période de quarantaine". Ils sont ajoutés à une liste à laquelle les ayants droits peuvent accéder et prendre des mesures judiciaires. Le Royaume-Uni a adopté une loi similaire en 2010, le "UK Digital Economy Act [38] ", qui a été suspendu jusqu'à 2015 . Fin 2011, la "Loi Sinde" a été approuvée en Espagne. Elle donne le droit à un organisme gouvernemental de bloquer les sites Web impliqués dans le piratage [39]. Une commission de la propriété intellectuelle décide si elle veut prendre des mesures contre un site enfreignant la loi ou les fournisseurs d'accès Internet qui l'hébergent. Un juge se prononcera par la suite sur la question savoir si le site doit être fermé. En Italie, une seule accusation est tout ce qu'il faut pour appliquer des sanctions juridiques. Contrairement à la réponse graduée que les autres pays ont adoptée, la législation italienne du droit d'auteur permet de déconnecter les pirates, avec la coopération des FAI, après un seul avis. Les ayants droit pourraient déposer plainte dans ce cas et pour éviter une récidive les FAI maintiennent une liste des fraudeurs. En 2013, les fournisseurs d'Internet américains ont signé un accord volontaire avec les ayants droit pour sévir contre les contrevenants au droit d'auteur. La coordination est faite par le Centre d'information sur le droit d'auteur CCI (Center for copyright Information) [40], qui met fortement l'accent sur l'éducation avec un programme de six étapes que les fraudeurs reçoivent comme avertissement [41]. L'objectif de ces mesures est à la fois de lutter contre le piratage et de promouvoir l'utilisation des solutions numériques pour la distribution de contenu légalement.

2.3 La loi anti-piraterie française : Hadopi

L'acronyme Hadopi se réfère à la loi Hadopi (Haute Autorité pour la diffusion des Œuvres et la Protection des Droits sur Internet) qui est une loi française présentée en Avril 2008 et adoptée en 2009 pour protéger les œuvres (la propriété intellectuelle) sur Internet. Une fois adoptée, un comité, dont le nom est également Hadopi, a été créé pour protéger les contenus sous droit d'auteur et promouvoir des moyens légaux pour distribuer les contenus numériques sur Internet. Pour lutter contre le piratage, la version initiale de la loi définie ladite procédure de "réponse graduée". Lorsque l'utilisateur est détecté pour la première fois pour un contenu illégal, il recevra un avertissement par courriel. La seconde fois, une lettre recommandée sera envoyée. Lors de la troisième infraction, l'accès à Internet sera suspendu. La version initiale de la loi a conduit à un grand nombre de débats liés à l'utilisation de l'adresse IP pour identifier la personne qui effectue un téléchargement illégal

(le propriétaire de l'abonnement Internet et la personne qui effectue le téléchargement pourrait être différentes) ou la contradiction entre la loi Hadopi et le « pack Télécom» proposé au niveau européen qui stipule que la suspension de l'accès à Internet ne peut être décidée que par une autorité judiciaire et non administrative. Par conséquent, une version révisée de la loi, connue sous le nom Hadopi 2, a été adopté en Septembre 2009.

2.3.1 Aperçu de l'impact de la loi Hadopi sur l'utilisation du P2P

Nous présentons sur la figure 2.1, une vue globale de la quantité de pairs qui téléchargent du contenu illégal, comme observé par notre plateforme expérimentale. Notez qu'un échantillon correspond ici à une adresse IP localisée en France, et le contenu téléchargé est illégal. Nous pouvons directement remarquer que deux protocoles dominent, à savoir eDonkey / Kad et BitTorrent. Une activité négligeable est observée sur les autres réseaux P2P surveillés par notre plateforme (plus de détails dans la section plateforme expérimentale). Nous rapportons sur la figure 2.1 des événements pertinents qui pourraient avoir un impact sur l'utilisation des réseaux P2P pour télécharger des contenus illégaux. En Octobre 2010, les premiers e-mails d'alerte ont été envoyés par Hadopi. Nous pouvons observer sur la figure 6 un léger effet sur les utilisateurs français du P2P.

En revanche, la deuxième série de courriels d'avertissement envoyée en février 2011 a mené à une diminution remarquable du nombre d'infractions un mois plus tard. En Octobre 2011, dix personnes qui avaient reçu des avertissements par emails dans une première étape, puis des lettres recommandées ont été convoqués à se justifier devant des magistrats de Hadopi. En février 2012, une première affaire a été envoyé devant la cour, accusée d'avoir effectué illégalement des téléchargements . En Avril 2012, l'élection du président français a eu lieu. Ce fut l'occasion d'un grand nombre de débats autour de l'avenir de la loi Hadopi. En Août 2012, le ministre français de la culture a déclaré que la suspension de l'accès à Internet n'était pas une méthode appropriée. En Septembre 2012, un premier pirate a été condamné par le tribunal à une amende de 150 euros après avoir avoué qu'il n'a utilisé aucune sécurité pour empêcher l'accès à sa connexion Internet. Plus tard, en mai 2013, une évaluation de la loi Hadopi a été publié par le gouvernement français avec des recommandations visant à renforcer les lois anti-piratage.

En Juin 2013, un nouveau contrevenant a été condamné à une suspension de son accès à Internet pendant 15 jours et une amende de 600 euros.

Dans l'ensemble, en faisant une liaison entre les événements décrits ci-dessus et à ce qui a été observé via notre plateforme de test, nous pouvons remarquer que eDonkey / Kad et BitTorrent ont connu une baisse significative de leur activité après la deuxième série de courriels d'avertissement émise et les événements ultérieurs cités n'ont pas modifié cette tendance. La fermeture de MegaUpload [13] en Janvier

13. http ://www.bbc.com/news/technology-16642369

2012 n'a pas donné lieu non plus à un regain d'activité des principaux réseaux P2P. Nous pouvons néanmoins observé que eDonkey / Kad et BitTorrent connaissent un sort différent après Septembre 2011, les deux premiers continuent leur diminution quant à BitTorrent, il reprend régulièrement son activité. Nous reviendrons sur ce point plus tard dans ce chapitre.

2.4 Plateforme expérimentale

La plateforme de R&D utilisée dans cette étude se compose d'un environnement client-serveur conçu pour analyser les réseaux P2P et les sites Web pour recueillir et analyser des preuves. Toutes les deux semaines une liste des titres les plus téléchargés (films et musique) est mise à jour. La liste contient 5000 titres audio et 100 films. Pour identifier le contenu piraté, une recherche de texte initiale est effectuée pour identifier potentiellement les fichiers par balayage complet sur plusieurs sites. Chaque nouvelle instance d'œuvre identifiée est entièrement téléchargée et examinée manuellement ou avec la technique du fingerprinting (empreinte digitale) automatisée pour vérifier qu'il s'agit en effet d'une copie illégale réelle du contenu. Les résultats sont ajoutés à une base de données qui stocke les métadonnées pertinentes sur chaque fichier, y compris son nom, les valeurs du hash, la taille et les informations sur le réseau P2P où le fichier a été trouvé (Kad / eDonkey, Bittorrent, etc.). Cette étape de vérification est cruciale pour assurer que la plateforme R&D ne surveille pas des œuvres non protégées et c'est aussi un élément clé pour identifier correctement les infractions P2P des droits d'auteurs. Pour chaque réseau P2P, des nœuds de collecte sont déployés sur la base des contenus piratés identifiés.

Les nœuds de collecte exécutent un logiciel (un code) développé au sein de TMG, et sont déployés sur des machines virtuelles dans 70 serveurs physiques avec plus de 500 adresses IP statiques et dynamiques. Les nœuds de collecte sont tous conçus pour intégrer et fonctionner sur le réseau P2P en tant que pairs standards ainsi que communiquer et télécharger des données à partir d'autres pairs, tout comme un client P2P standard. Les nœuds de collecte cherchent à télécharger des échantillons et à créer des paquets de données probantes, y compris (entre autres données) l'adresse IP, le port, heure / date, la taille, le PeerID et la valeur du hach.

Le téléchargement d'un échantillon, généralement autour de 10 kbytes, est réalisé après avoir vérifié qu'il s'agit d'une IP française. Un échantillon se compose de ce qu'on appelle des morceaux du contenu. Ces pièces individuelles sont vérifiées par la fonction de hachage cryptographique SHA1 pour vérifier s'il s'agit bien du contenu d'origine ciblé. Après confirmation, le téléchargement est arrêté et l'adresse IP est rapportée comme étant impliquée dans une action illégale. Le téléchargement est limité à un morceau du fichier pour permettre aux nœuds de collecte de minimiser l'espace de stockage requis et cibler autant de pairs partageant un contenu donné que possible. En raison de contraintes techniques, toutes les adresses IP détectées chaque jour par la plateforme ne peut être vérifiées pour

réellement détenir un échantillon du contenu. La plateforme est actuellement en mesure de vérifier 70 000 adresses IP par jour. Les résultats de la figure 2.1 correspondent à toutes les adresses IP observées (sans vérification). Pour les résultats présentés dans la section suivante, nous allons nous concentrer sur les adresses IP pour lesquelles une vérification a été réalisée.

2.5 Résultats

Nous avons observé dans la section **??** que bien que le volume de l'activité illégale du P2P a diminué significativement après la mise en place de la loi Hadopi, les tendances de BitTorrent et eDonkey/Kad sont clairement différentes. Nous visons à étudier plus cette question dans cette section, selon deux dimensions : la taille du contenu que l'utilisateur souhaite télécharger et l'efficacité des différents protocoles par rapport à la taille du contenu.

2.5.1 Analyse par rapport à la taille du contenu

Les réseaux P2P sont principalement utilisés pour fournir des fichiers audio et vidéo. Les fichiers audio peuvent avoir des tailles différentes selon la nature du contenu, comme nous pouvons le voir sur le Tableau 2.1

Idem pour les fichiers vidéo, voir le tableau 2.2. La qualité d'encodage ici est le facteur principal qui influence la taille du contenu, et dans cette catégorie nous pouvons aussi trouver des compilations de vidéos et des clips. Plus précisément, un contenu vidéo fait généralement partie de l'une des catégories suivantes :

— CAM, TS : Enregistré dans une salle de cinéma en utilisant un caméscope, un smartphone ...
— VODRip : Créé par l'enregistrement ou la capture d'une vidéo / film à partir d'un service à la demande tel que le câble ou service TV satellite.
— WEBRip, WEB-DL : Créé par une capture vidéo à partir d'un écran, en utilisant des services comme CanalPlay, Netflix, Amazon ...
— DVDRip : Version commerciale finale d'un film, divulguée (dévoilée) avant qu'il ne soit disponible en dehors de sa région. (la date de sortie d'un DVD change d'un pays à un autre).
— HDTV, HDTVRip : source capturée à partir d'une carte de capture analogique. La qualité de ces captures peut surpasser les DVDs.

Pour le cas audio et vidéo, nous utilisons une classification par taille introduite dans les tableaux 2.1 et 2.2 pour observer l'évolution de popularité de chaque catégorie de contenu.
Les figures 2.2 et 2.3 montrent les évolutions du nombre d'adresses IP françaises

TABLE 2.1 – Nature du fichier selon sa taille (contenu audio).

Moins de 10 Mbytes	De 10 à 100 MB	Plus de 100 MB
Taille d'une chanson (un single)	Taille d'un album d'un artiste, album des Hits du moment...	Compilation d'un artiste, par type de musique ..., Hits d'une décennie ...

TABLE 2.2 – Nature du fichier selon sa taille (contenu video).

Moins de 700 MB	De 700 et 1GB	Plus d'un GB
Film de qualité médiocre (CAM, Screener, WebTV), épisode de série d'une durée relativement courte...	Vidéo de bonne qualité (DVDrip, BRrip...) HDTV	Vidéo HD, BluRay, 3D...

observées sur les réseaux P2P, impliquées dans le téléchargement de contenu protégé entre Septembre 2010 et Avril 2013. La baisse observée au cours des deux derniers mois dans toutes les figures de cette section est un artefact de mesure dû à l'arrêt de la collecte des adresses IP vérifiées au milieu du dernier mois contrairement à l'ensemble des adresses IP utilisées dans la figure 2.1. Notez que le téléchargement ici est défini dans le sens expliqué dans la section (plateforme expérimentale) : l'IP est reconnue dès lors qu'elle détient au moins un morceau valide du contenu protégé. Ceci confirme l'intérêt de l'utilisateur derrière cette IP pour ce contenu, mais ne garantit pas que ce dernier a effectué le téléchargement du contenu en entier.

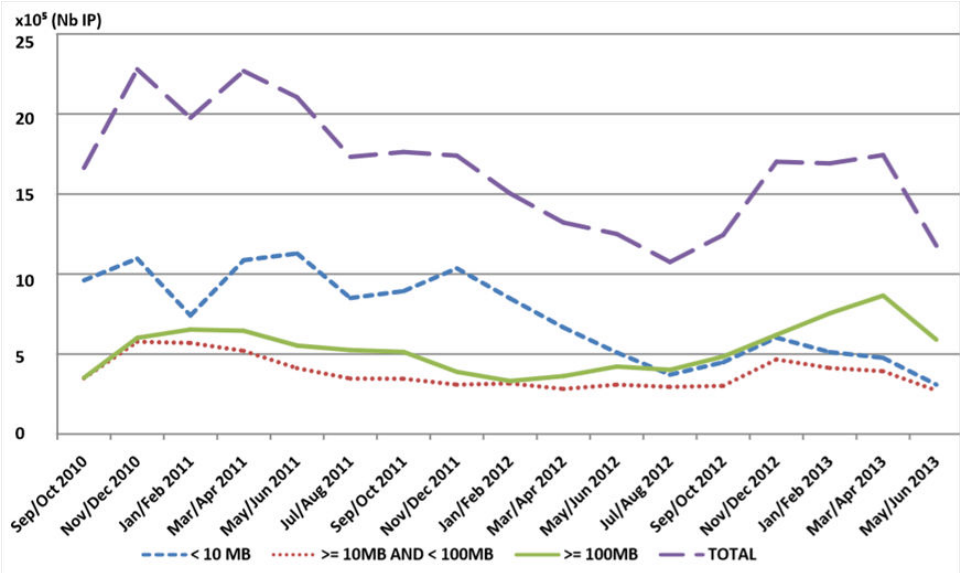

FIGURE 2.2 – Téléchargement des fichiers selon la taille du contenu (Audio)

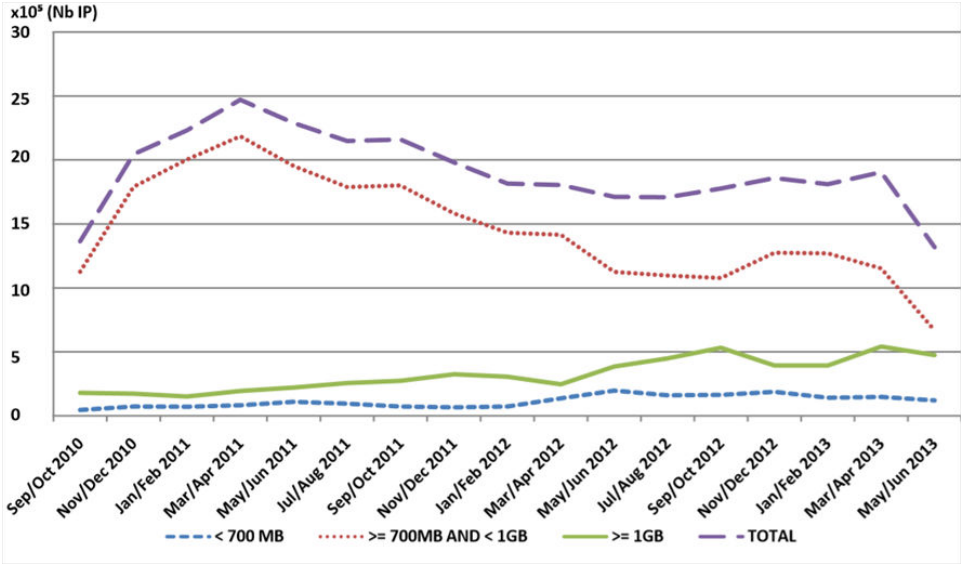

FIGURE 2.3 – Téléchargement des fichiers selon la taille du contenu (Video)

Pour les deux contenus audio et vidéo, nous pouvons observer clairement une nette tendance entre fin 2010 et mi-2013 : les utilisateurs sont attirés par des contenus de taille croissante. Cela signifie qu'ils préfèrent des albums ou des compilations complètes plutôt que les morceaux de musique quand il s'agit de contenu audio. De même pour le cas de la vidéo, les utilisateurs préfèrent les contenus encodés avec une qualité supérieure. Cette tendance est conforme à ce qui est observé sur le marché de la télévision en France, où les gens sont prêts à acquérir de plus grands écrans qui offrent une haute définition et supportent la 3D. Cette remarque signifie que, bien que le nombre de téléchargements observés sur les réseaux P2P peut diminuer, la quantité de volume correspondante (en octets) pourrait suivre une tendance différente. Nous rapportons sur la figure 2.4 et 2.5 les volumes par type de contenu, ainsi que les volumes globaux par contenus audio et vidéo. Les courbes peuvent être biaisées par les seuils que nous avons définis lors de la capture et, qui limitent au nombre maximum d'IP à détecter par jour. Nous pouvons néanmoins observer que les courbes du volume total montrent une claire tendance à la hausse et sont orientées par le désir des utilisateurs finaux de contenus audio et vidéo plus volumineux.

2.5.2 Type de contenu vs. Protocoles

Dans la section précédente, nous avons observé que l'intérêt des utilisateurs a évolué vers des contenus audio et vidéo plus volumineux entre 2010 et 2013. Nous avons ensuite cherché à savoir si cette préférence pour les contenus volumineux peut expliquer l'écart observé entre BitTorrent et eDonkey / Kad sur la figure 2.1, où après la chute initiale des deux protocoles en Février 2011, eDonkey / Kad a

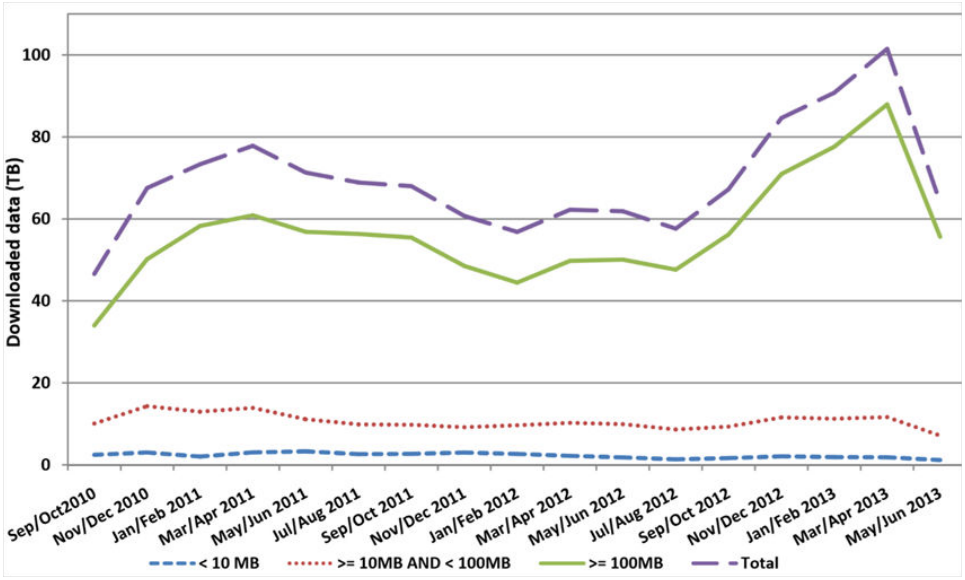

FIGURE 2.4 – Volume de donnée téléchargée selon les catégories de taille définies (Audio)

TABLE 2.3 – Taille de contenu et durée de la session de téléchargement par BitTorrent et Kad / eDonkey

Title of content	**Size**	**BitTorrent**	**Kad / eDonkey**
Game of Thrones S03Ep10 Final	551 MB	06mn10s	1h00mn02s
Breaking bad S01Ep07 Final	353.5 MB	08mn41s	38mn21s
Veronica Mars the movie	1.36 GB	07mn59s	18h31mn42s
The wolf of wall street	1.38GB	06mn42s	4j18h46mn18s
Will power Album – Will i.am	185 MB	02mn39s	14h36mn48s
The 20/20 experience – Justin Timberlake	142 MB	02mn11s	8h03s
Frank Sinatra- My way	4.24 MB	04mn31s	05mn06s
Cris Cab – Liar Liar	8.16 MB	02mn39s	10mn35s

continué à diminuer tandis que BitTorrent a recommencé à augmenter après Novembre 2010. Nous pensons que la raison principale pour laquelle BitTorrent est préféré à eDonkey / Kad est que la réplication du fichier devrait se traduire par un temps de téléchargement réduit par rapport à celui du partage de fichier en général. Comme exemple illustratif, nous avons examiné différents contenus populaires disponibles sur eDonkey / Kad et BitTorrent et ensuite nous les avons téléchargés en utilisant l'un des clients les plus populaires du moment. L'ensemble du contenu va d'un single musical d'environ 4.24 MB à des films de haute qualité à plus de 1.3GB. Par la suite, nous les avons téléchargés à partir d'un serveur dix fois pour obtenir les résultats du tableau 2.3 (moyenne des sessions de téléchargement).

Il est clair que BitTorrent surpasse eDonkey / Kad par d'un ordre de grandeur à chaque fois. Nous sommes conscients que l'utilisation d'une machine avec un haut débit peut biaiser les mesures de BitTorrent , mais nous croyons que ceci

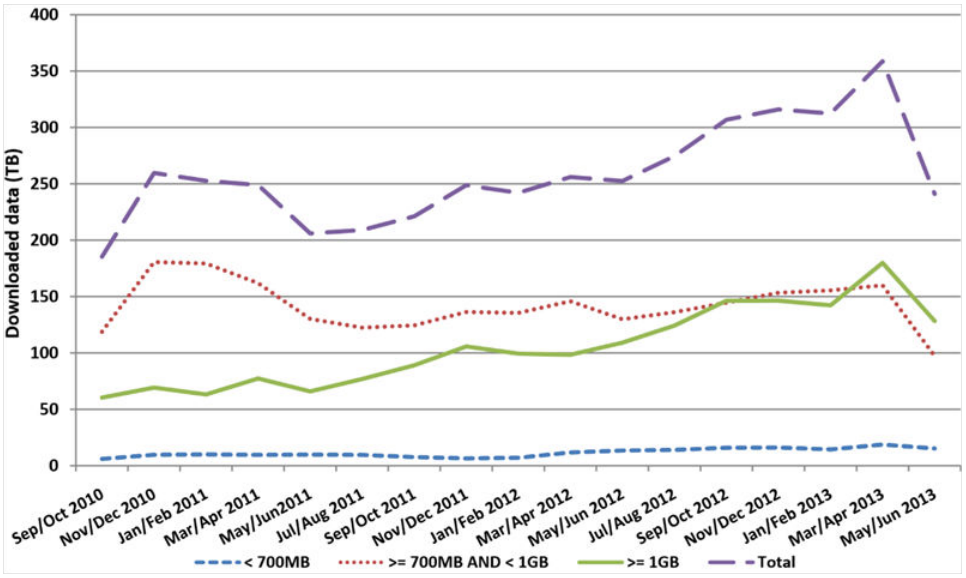

FIGURE 2.5 – Volume de donnée téléchargée selon les catégories de taille définies (Video)

est suffisant pour soutenir notre hypothèse, que BitTorrent surpasse généralement eDonkey / Kad en ce qui concerne le temps de téléchargement. Ceci est confirmé par les utilisateurs eux-mêmes, en examinant les contenus de plus que 1GB : près de 100% d'entre eux ont été desservis en utilisant BitTorrent. Une raison alternative pour laquelle les utilisateurs préfèrent BitTorrent à eDonkey / Kad pourrait être que le premier expose moins l'utilisateur quand il est engagé dans un téléchargement illégal. En effet, lors de l'utilisation de BitTorrent, un utilisateur ne révèle que son intention de télécharger le contenu correspondant à un torrent donné. En revanche, en utilisant eDonkey / Kad, l'utilisateur expose un répertoire partagé en entier. Bien que nous ayons aucune preuve directe du facteur qui motive les utilisateurs a préféré BitTorrent à eDonkey / Kad, nous croyons que le temps de téléchargement réduit est pour la plupart des gens, la motivation dominante.

Nous avons généré à partir des données de la plateforme TMG l'évolution de l'utilisation des réseaux P2P les plus populaires chez les utilisateurs. Nous rapportons sur la figure 2.6 et 2.7 les données de l'année 2011, 2012 et 2013. Bien que l'utilisation de ces réseaux pour le téléchargement illégal des contenus audio connait une évolution modérée (2.7), il est clair que pour les contenus de type vidéo, et donc de grande taille, les préférences ont bien changé (2.6). En accord avec nos conclusions précédentes, l'orientation vers un réseau qui expose moins ses utilisateurs et réduit le temps de téléchargement a fait que BitTorrent a marqué un gain de 49,47% de 2011 à 2013.

Sur la figure 2.4, nous voyons le protocole Soulseek que nous n'avons pas évoqué auparavant vu sa faible utilisation par rapport aux autres protocoles. Soulseek est un réseau P2P qui est apparu en 1999 par un ancien programmeur de chez Napster. Le réseau est spécialisé dans les types de musique moins commerciale comme

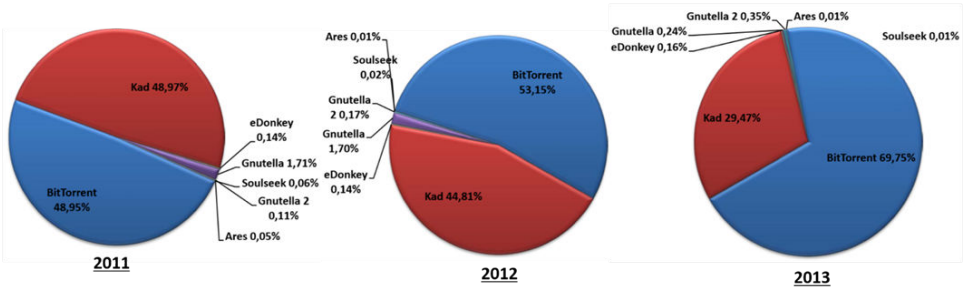

FIGURE 2.6 – Utilisation des réseaux P2P populaires chez les français (Audio)

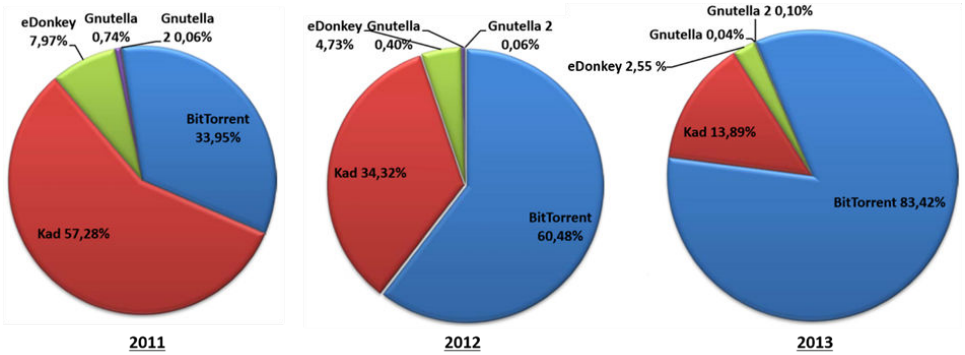

FIGURE 2.7 – Utilisation des réseaux P2P populaires chez les français (Video)

l'underground et la musique alternative.

2.6 Positionnement par rapport à l'état de l'art

Contrairement aux travaux présentés dans la section 1.4.1, notre étude repose sur des mesures effectuées sur une période de deux ans et demi et couvre tous les réseaux P2P majeurs simultanément. Cette vue unique du trafic P2P nous permet d'observer avec précision l'effet de la loi Hadopi, ainsi que divers autres événements connexes, sur la quantité de trafic P2P illégal et ses caractéristiques (type, taille). Sans donner de conclusion quant à la quantité totale de contenus illégaux traversant l'internet (qui peuvent être partagées en utilisant différentes techniques qu'un réseau P2P), notre étude met en évidence la diminution significative du trafic P2P sur les principaux réseaux P2P. Plus important encore, nous démontrons que la demande pour des contenus plus volumineux (contenus plus grands qu'un Gigaoctet) est en constante augmentation depuis mi-2011, et ainsi nous montrons la raison pour laquelle seul BitTorrent en profite. A partir des résultats obtenus nous avons discuté l'évolution du trafic P2P en fonction de l'évolution de la qualité des contenus ainsi que le comportement utilisateur.

2.7 Conclusion

Dans ce chapitre, nous avons profité des données collectées par une infrastructure expérimentale qui nous permet d'observer le trafic P2P illégal sur Internet en France. Cela nous a permis d'étudier l'impact de la loi anti-piratage adoptée dans ce pays depuis l'année 2009. Nous avons confirmé, mais à plus grande échelle, ce qui a déjà été observé par d'autres études : le trafic P2P a diminué significativement après la première campagne massive de courriers électroniques aux utilisateurs engagés dans des téléchargements illégaux. En outre, la fermeture de Megaupload n'a pas conduit à des changements significatifs sur l'utilisation des protocoles P2P pour distribuer du contenu illégal. Ceci pourrait signifier (comme suggéré par d'autres études) que de nouveaux moyens plus furtifs sont utilisés maintenant pour distribuer des copies illégales de contenu protégé par des droits d'auteur.

Notre étude a également révélé, particulièrement pour la vidéo, qu'actuellement les personnes recherchent des contenus de haute qualité, ce qui se traduit par une augmentation significative de la taille de ce type de fichiers. La taille croissante de ces fichiers explique à son tour la domination de BitTorrent sur eDonkey / Kad, les deux étant les protocoles P2P les plus populaires en France. En effet, BitTorrent appartient à la famille des protocoles de réplication de fichier qui met l'accent sur la réplication rapide des données plutôt que de la capacité de localiser les serveurs pour un contenu donné, comme dans le cas des applications de partage de fichier comme eDonkey / Kad.

Chapitre 3

Etude des protocoles Live à partir de données réelles

Nous nous intéresserons dans ce chapitre aux protocoles P2P utilisés pour faire du streaming live, c'est-à-dire de l'envoi de contenu vidéo qui sont consommés directement et non stockés par le client. Il peut s'agir par exemple d'une retransmission d'un match de football. Nous nous concentrons dans ce chapitre sur un protocole particulier, le protocole Sopcast qui est apparu, sur les sondes de mesures de TMG, comme un protocole très populaire pour la diffusion de contenu live. Ce protocole est fermé, c'est-à-dire que le code n'est pas public. Nous découvrons la nouvelle stratégie (nouvelle au moment où nous avons mené cette étude en 2012) appliquée par Sopcast pour sélectionner le voisinage initial d'un nouveau pair arrivant. Nous démontrons par la suite, que l'efficacité de l'algorithme P2P dépend de la taille de la population (à savoir, la popularité de la chaine). Nous démontrons également qu'un autre paramètre clé à prendre en compte est la relation géographique entre le contenu et le nouveau pair.

3.1 Le Live et Sopcast

En raison de leur scalabilité, chaque nœud du réseau étant à la fois client et serveur, les applications P2P présentent une infrastructure propice pour la distribution live sur Internet. Les nœuds du réseau transfèrent et distribuent les vidéos chunks d'une manière coopérative contribuant ainsi avec leurs ressources à réduire le besoin en bande passante. La puissance du réseau sera d'autant plus accentuée avec le nombre de pairs présents sur celui-ci augmente. Le réseau dans ce cas n'a pas besoin d'une entité tiers pour la gestion ni de serveur centrale pour la distribution. Les applications P2P cependant nécessitent une faible latence, une faible gigue et le moins de perte de paquet possible, ce qui est peut-être complexe dont les noeuds (les pairs) sont indépendants.

3.2 Présentation de sopcast

Le P2PTV ou peer-to-peer television désigne les applications P2P qui permettent de diffuser du contenu multimédia sur Internet. La popularité de ces applications n'a cessé de croitre en raison de leur capacité à distribuer du contenu avec ou sans protection des droits d'auteur à un large public. Le contenu est en général un stream (flux) vidéo d'un programme télévisé diffusé en direct, découpé en plusieurs pièces (chunks) est transporté à travers l'infrastructure du réseau IP. Il existe plusieurs applications P2P sur Internet comme Sopcast [1], Ace Stream [2] (Torrent Stream),TVU [3], TvAnts, PPLive [4], PPStream [5] . Des études ont montré que Sopcast est l'application P2PTV la plus populaire en Europe [44]. Nous avons remarqué un résultat similaire sur plusieurs de nos collectes faites lors de plusieurs manifestations sportives au cours de ces dernières années. Par exemple, lors d'un match de finale de la Champions League opposant le club FC Barcelone à Manchester United en Mai 2011, Sopcast a été utilisé par 66% des personnes qui suivaient le match sur des applications P2P2TV (voir figure 3.1).

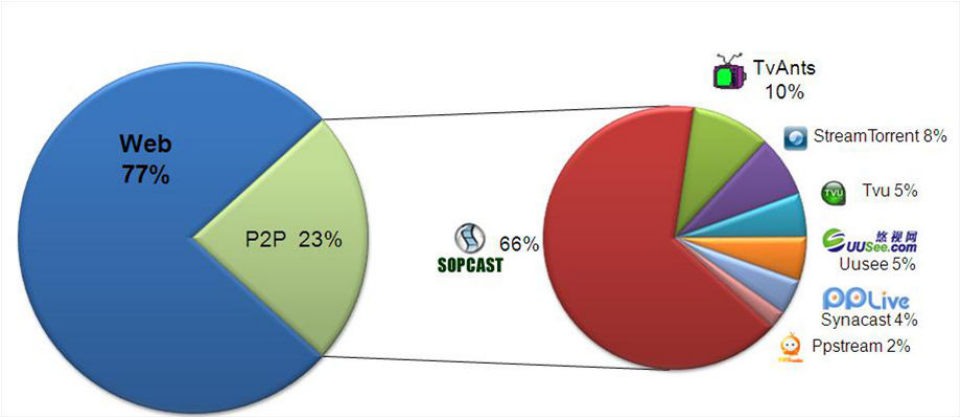

FIGURE 3.1 – Distribution

Début 2011, Sopcast avait modifié son algorithme de la phase start-up ou démarrage. L'objectif principal était de fournir aux nouveaux arrivant une liste de pairs qui -probablement- étaient les plus proches. Cette stratégie est un changement majeur dans le protocole, qui reposait précédemment sur une approche aléatoire pour le choix des adresses IP dans la liste tenue par le Tracker. Le rôle du Tracker est de maintenir la liste des pairs actifs et d'aider les nouveaux arrivants dans la découverte de leurs voisins. Cette stratégie de sélection de pairs peut-être interprétée comme une approche Peer-for-Peer (P4P), qui vise à obtenir un chemin entre deux pairs le plus court possible (en temps, en nombre de sauts, etc).

1. http ://www.sopcast.org/
2. http ://www.acestream.org/
3. http ://www.tvunetworks.com/
4. http ://www.pptv.com/
5. http ://www.pplivepp.com/

En raison de cette importance capitale du start-up (phase de construction du voisinage) dans les protocoles P2PTV, nous analyserons dans ce chapitre le comportement de Sopcast dans la phase du démarrage et la peer-list (désignée PL dans ce qui suit). Nous nous appuyons sur une approche expérimentale pour découvrir l'impact du changement de l'algorithme de sélection dans Sopcast. Nous utilisons plusieurs pairs à divers localisation à travers le monde et nous nous connectons à des flux de différente taille pour évaluer son efficacité.

3.3 SopCast

Sopcast repose exclusivement sur le protocole de transport UDP. Il offre une variété de chaînes qui peuvent être regroupées par langue, ID, région, etc. Sopcast permet aux utilisateurs de créer leur propre chaîne de diffusion s'ils sont inscrits. Pour chaque chaîne, Sopcast fournit un indicateur de qualité, présenté comme une barre bleue à côté de l'ID de la chaîne. En cliquant sur cette barre, une trame d'information apparaît affichant l'indicateur de popularité de la chaîne qui est entre zéro et un, le format d'encodage (WMV, WMA, RMVB ...), le temps de démarrage de la diffusion et le débit binaire (bitrate) allant généralement de 380 Kbps à 1168Kbps (voir la figure 3.2).

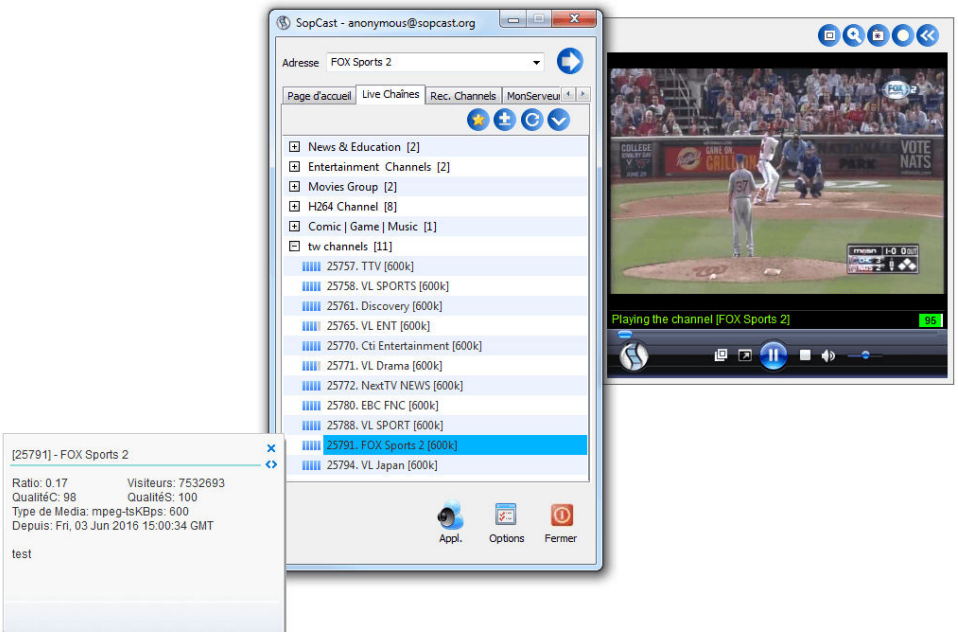

FIGURE 3.2 – Interface Sopcast

Le code source du protocole de l'application est fermé. En outre, les données échangées entre les hôtes sont cryptées. Nous nous appuyons sur une approche expérimentale pour découvrir ses performances. Comme indiqué dans la section 3.2,

le temps de démarrage d'une chaîne joue un rôle crucial dans la qualité d'expérience (QoE) des utilisateurs. Par conséquent, nous avons commencé par l'analyse de la latence au niveau du réseau entre un pair et ses voisins, car ce paramètre influe directement le temps nécessaire pour commencer à regarder une chaîne donnée. En outre, la latence influe directement sur le décalage entre l'instant ou le contenu original -source- est rendu et celui auquel les utilisateurs finaux se trouvent. Une valeur typique pour le délai maximum entre pairs, par rapport à la source initiale du flux, est d'une minute. Cela signifie qu'aucun pair ne devrait être à plus d'une minute de retard par rapport à la source originale du contenu [69]. Il est évident que la latence réseau entre pairs est une mesure clé pour évaluer la qualité de l'expérience (QoE) d'un utilisateur.

3.3.1 Architecture

Dans Sopcast, chaque chaîne correspond à un flux indépendant. Un client peut se joindre à un flux en contactant d'abord un serveur central – tracker- similaire au tracker BitTorrent [57]. Le tracker, qui garde une trace des pairs sur chaque chaîne, fournit au nouveau client une liste de 32 adresses IP (PL) à contacter. Ainsi, le tracker joue un rôle principal dans la phase de démarrage. En cas d'échec du tracker (si le tracker "tombe"), il serait impossible pour un nouveau pair de rejoindre une chaîne. Dans Sopcast, les pairs qui téléchargent le contenu à partir de la source de la vidéo directement sont nommés super-peers. Sopcast limite aussi le nombre de pairs qui peuvent se connecter à la source initiale à 11 pairs. Tous les pairs peuvent devenir des super-peers. Les données échangées entre les pairs sont cryptées. De plus, les algorithmes qui contrôlent la sélection des chunks et des pairs ne sont pas publics. Par conséquent, nous ne sommes pas en mesure de confirmer si Sopcast utilise également les algorithmes tit-for-tat (qui définit combien si pair va collaborer avec un autre) et / ou le rarest- first (qui définit quelle pièce un pair demande si l'algorithme tit-for-tat permet un échange) comme BitTorrent.

3.3.2 Analyse du protocole

Afin de comprendre le comportement des pairs dans Sopcast, nous avons analysé le trafic en provenance et à destination de nos clients que nous avons déployé sur différents sites. Les résultats présentés ci-dessous confirment ce que les autres études ont également montré : quand un pair se connecte à une chaine, différents paquets (détaillés ci-après) sont échangés, voir le tableau 3.1. Pour contacter des nouveaux voisins potentiels ainsi que pour communiquer avec le tracker, les pairs envoient d'abord un paquet pour initier la communication ou un Hello packet. Quand un client envoie un paquet hello au tracker pour rejoindre une chaîne, celui-ci répond avec un paquet de 80 bytes suivie d'un paquet de 60 bytes. Nous le désignons comme paquet de la liste des pairs (paquets PL). Les paquets PL contiennent une

TABLE 3.1 – Description des paquets Sopcast

Taille du paquet	Description
94	Initiation / Hello packet
80 à 166	Réponse au Hello packet
70	Acquittement / Acknowledgment
84	Keep alive packets
62-76-82-88	Paquets de contrôle
1000 à 1320	Données- contenu vidéo

sélection de pairs qui regardent déjà la chaîne et l'offset actuel dans celle-ci, à savoir, le temps écoulé entre le temps du début de la diffusion et le temps auquel le client envoie sa requête au tracker.

3.4 Plateforme de tests

Après le paquet initial *Handshake* avec un pair, nous pouvons observer un certain nombre de paquets dont la taille varie – en charge utile- de 60 à 166 bytes, que nous désignons comme paquets de contrôle. Les paquets de contrôle contiennent un ensemble de requêtes, qui sont envoyées afin d'obtenir un chunk à partir d'une chaîne donnée. En effet, après avoir échangé les paquets de contrôle, nous observons la transmission ou la réception de paquets de données, dont la taille varie entre 1000 et 1320 bytes, et qui contiennent des chunks vidéo. Chaque paquet de données reçu est acquitté par un paquet de 70 bytes. Pour conclure, notre analyse de paquets révèlent un échange périodique de paquets de 84 octets entre le client et le tracker, qui semblent être utilisés pour indiquer que le pair actuel est toujours actif dans le réseau. Nous nous référons à ces paquets comme paquets *keep-alive*. En effet, le contenu des paquets d'acquittement *ACK* et de *keep-alive* est toujours le même. La figure 3.3 illustre les messages échangés entre deux pairs Sopcast.

Afin de comprendre l'algorithme du tracker pour établir une première PL, nous avons analysé une liste provenant de notre client Sopcast qui tourne sur une machine Linux. En effet, lors de l'exécution Sopcast en ligne de commande, le contenu de la PL est affiché dans le terminal.

3.4.1 Configuration experimentale et algorithme de sélection

Notre plateforme de mesure se compose de trois hôtes situés sur différents réseaux. La première machine est un ordinateur équipé d'un processeur 2,70 GHz Intel Pentium, 2 Go RAM exécutant un système Windows XP. Les deux autres ordinateurs sont des serveurs équipés de processeur 3.8 GHz. Dans chaque hôte, le client Sopcast tourne sur une machine virtuelle. La PL reçue par nos clients se compose de 32 adresses IP, sauf si nous sommes connectés à une chaîne avec peu de pairs (selon le ratio de la chaîne), ils en ont moins. Nous avons également observé

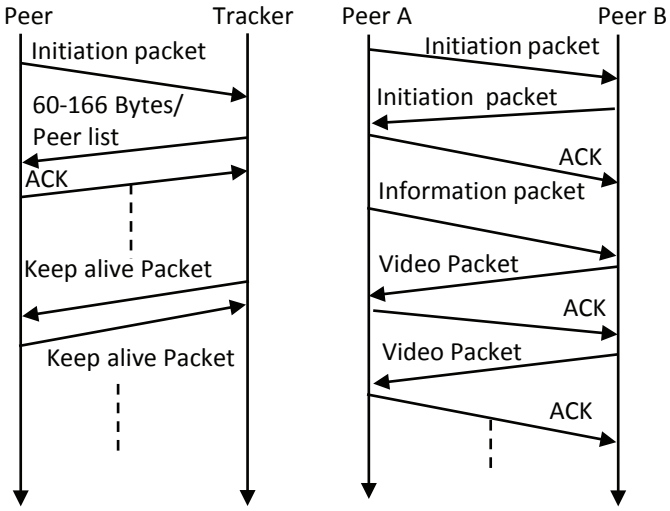

FIGURE 3.3 – Communication entre entités Sopcast

qu'une fois qu'un nouveau client arrive et établit le contact avec les pairs de sa première PL, il contacte des pairs supplémentaires obtenus grâce à un algorithme d'échange de pairs semblable à celui utilisé dans BitTorrent [57]. Cette information (adresse obtenue à partir du tracker ou d'un voisin) est indiquée clairement en ligne de commande. Avant les modifications apportées à Sopcast en 2011, les pairs ont été sélectionnés en fonction de leur taux de téléchargement, comme observé dans les études précédentes [58], [71]. Nos expériences démontrent que Sopcast utilise maintenant la distance entre les adresses IP afin de construire la première PL. Bien que la formule exacte pour cette distance est inconnue, nous pouvons supposer qu'il donne plus de poids à bits le plus signifiant de l'adresse IP. Pour calculer la distance, nous avons utilisé une forme simple de cette métrique entre deux adresses IP, IP_1 and IP_2, where $IP_j(i), j \in \{1, 2\}$ et le i-ème bit des adresses IP (i=1 est le bit le plus significatif) :

$$d(IP_1, IP_2) = \sum_{i=1}^{32} 2^{32-i} XOR(IP_1(i), IP_2(i))$$

Nous nous référons à cette distance, comme la distance IP dans la suite de ce chapitre. Notre objectif principal est d'étudier la pertinence de la distance IP pour construire la PL. En particulier, nous analysons la relation entre la distance IP de deux pairs et la latence qu'ils subissent. Celle-ci est mesurée à l'aide du protocole Ping. Nous utilisons également un application ping que nous avons développé. Notre application ping imite la procédure Handshake de Sopcast en envoyant un paquet Hello de 94 bytes, afin d'obtenir un ACK [6] d'une de taille 70 bytes du pair distant (voir le tableau 3.1).

6. acquittement de réception d'un paquet

TABLE 3.2 – Réponse des pairs aux requêtes ICMP

Chaîne	Hôte1 (class./appli.)	Hôte2 (class./appli.)	Hôte3 (class./appli.)
Pro TV	10/5	14/32	14/12
DigiSport	7/6	14/24	17/21
DigiFilm	9/13	20/26	17/14
Kanal ID	11/12	12/24	18/22
Acasa TV	14/12	12/24	12/22
Realitatea	14/19	11/26	18/6

3.5 Résultats

3.5.1 Configuration firewall

Avant d'aller plus loin dans les détails de l'algorithme de sélection de pairs utilisé par le tracker Sopcast, nous avons vérifié si le tracker suit une stratégie pour contrôler le nombre de pairs qui sont derrière un pare-feu pour composer une première PL. Pour estimer si un pair donné est derrière un pare-feu, nous exécutons à la fois un ping classique et notre application ping à celui-ci. Si le nœud répond aux deux ping, la probabilité d'être derrière un pare-feu est faible. Toutefois, si le nœud ne répond pas au ping classique, mais répond avec succès à notre application ping, alors, la probabilité que ce nœud soit derrière un pare-feu est élevée.

Le tableau 3.2 indique le nombre de pairs qui ont répondu aux pings classiques et de l'application, sur les 32 pairs dans la liste des pairs initiaux. Les chaînes dans le tableau 3.2 sont de popularité variée.

Nos résultats de ping montrent que le nombre de pairs derrière un firewall et qui composent la PL initiale ne suit apparemment pas de critères stricts. Ainsi, le tracker ne tient pas compte de la présence ou non d'un pare-feu afin de construire une telle liste.

3.5.2 Algorithme de sélection de pairs

Sopcast ne fournit aucune information sur l'algorithme utilisé par le tracker pour choisir les adresses et créer une PL pour un nouvel arrivant sur le réseau. Ainsi, le point de départ de cette étude était l'observation de la similitude entre l'adresse IP du nouveau pair et ceux qui composent la première PL.

Pour vérifier si les adresses IP du PL étaient toujours numériquement proches de l'un des nouveaux pairs, nous avons déployé trois client Sopcast sur des réseaux IP très différents. En effet, l'octet le plus significatif des adresses réseau des clients a toujours été différent. Plus tard, nous avons connecté les trois hôtes simultanément sur la même chaîne et nous avons recueilli les adresses IP obtenues à partir du tracker.

Enfin, nous avons calculé la distance numérique (ou distance de Hamming) entre chaque adresse IP du client et les adresses IP à partir de sa PL associée, ainsi que la distance entre chaque adresse IP du client et les adresses IP de la PL de nos deux autres clients. Nos résultats, représentés graphiquement sur les figures 3.4 et 3.5, montrent que la similarité des adresse IP est le principal critère utilisé par le tracker lors de la construction de la PL. En effet, la distance IP entre un client et sa PL associée est inférieure à la distance entre un client IP et la PL associé à d'autres clients situés dans d'autres réseaux. Notez que dans les figures 3.4 et 3.5, nous utilisons deux axes y différents. L'axe des y de droite correspond à la distance entre un pair et sa propre PL tandis que celui de gauche est utilisé pour afficher les distances avec les PL de nos autres clients.

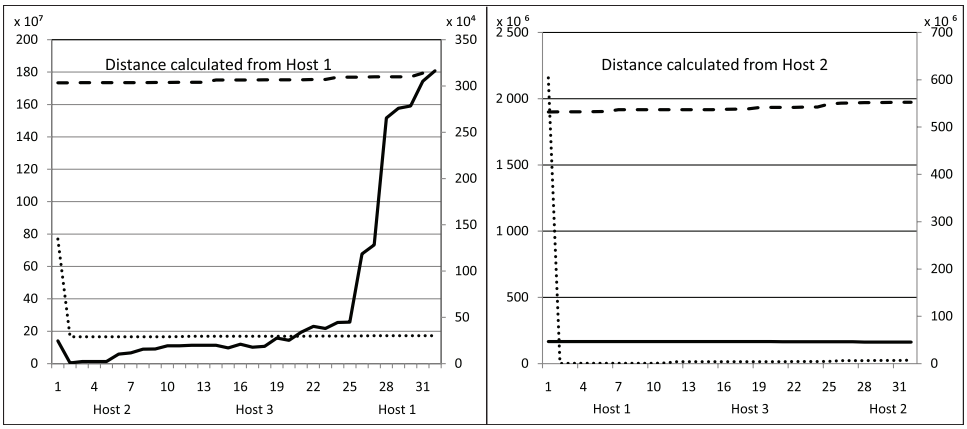

FIGURE 3.4 – Distance IP entre pairs

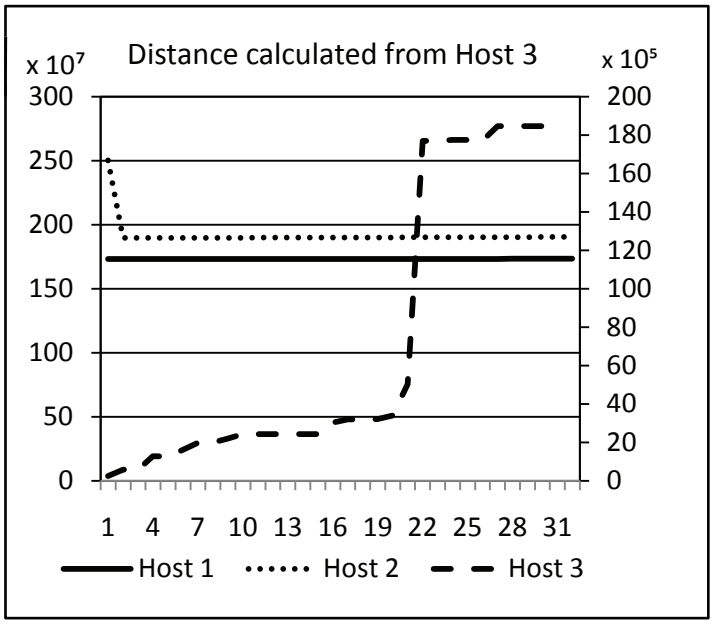

FIGURE 3.5 – Distance IP des pairs

De plus, nous voulons souligner que le nombre de pairs impliqués dans la chaîne joue un rôle important sur la distance estimée entre un client et son PL. Par exemple, si la chaîne réunit plusieurs pairs dont les adresses sont proches du client, nous obtiendrons un résultat similaire à ceux de la figure 3.4, où la fonction de la distance entre le client et ses pairs augmente ”harmonieusement”. Dans le cas contraire, ou seulement quelques pairs proches du client sont connectés à la chaîne (moins de 32) et que les autres pairs sont situés sur des réseaux très différents (ce qui peut être le cas s'ils sont géographiquement loin) alors la fonction peut afficher une pente à un certain point de la figure, semblable à celui des résultats de la figure 3.5. Dans ce dernier cas, l'impact de l'algorithme du tracker ne serait pas facilement perceptible.

Les expériences décrites dans cette section ont été exécutées à plusieurs reprises et nous avons obtenu des résultats similaires. A travers nos multiples expériences, nous avons également constaté que lorsqu'un client rejoint à plusieurs reprises le même canal (BeIn Sport) sur de courtes périodes de temps, la première PL reste inchangée, ce qui renforce notre hypothèse : la similitude IP est le principal critère utilisé par le tracker pour créer une PL.

3.5.3 La dynamique du réseau Sopcast

Après avoir reçu la première PL, le pair prendra contact avec chaque noeud de cette liste afin d'obtenir le contenu multimédia. Les pairs utiliseront le protocole Peer Exchange (PEX) [82] pour recevoir les PLs voisines comme dans BitTorrent.

L'objectif de ce mécanisme est double. D'une part, les pairs actifs d'un canal auront une vue mise à jour de l'architecture du réseau et, d'autre part, la collaboration des nouveaux pairs dans le réseau peut potentiellement améliorer les performances du système de diffusion (TV broadcasting).

3.5.4 Distance IP vs latence

Nous étudions ensuite la relation entre la distance IP numérique (référencé network distance sur les figures) et la latence réseau. Nous avons estimé le temps de réponse RTT (Round-Trip Time), moyenné sur quelques dizaines de collectes, entre nos clients Sopcast et leurs PL associées collectées au cours de l'expérience décrite précédemment. Nos résultats montrent que, pour les chaînes populaires (chaîne avec une popularité supérieure à 0.25 selon nos clients Sopcast), le RTT expérimenté entre le client et les dix premiers pairs de la PL (les pairs les plus proches) a tendance à être plus petit que le RTT expérimenté entre celui-ci et les pairs restants. La figure 3.6, qui montre le RTT moyen et la distance de Hamming entre le hôte 3 et ses pairs de la première PL, illustre parfaitement nos observations.

Toutefois, si la popularité de la chaîne est faible (chaîne avec une popularité inférieure à 0.1 selon le l'indice de popularité indiqué par nos clients Sopcast) et si le tracker renvoie moins de 32 pairs, le RTT et la distance de Hamming semble

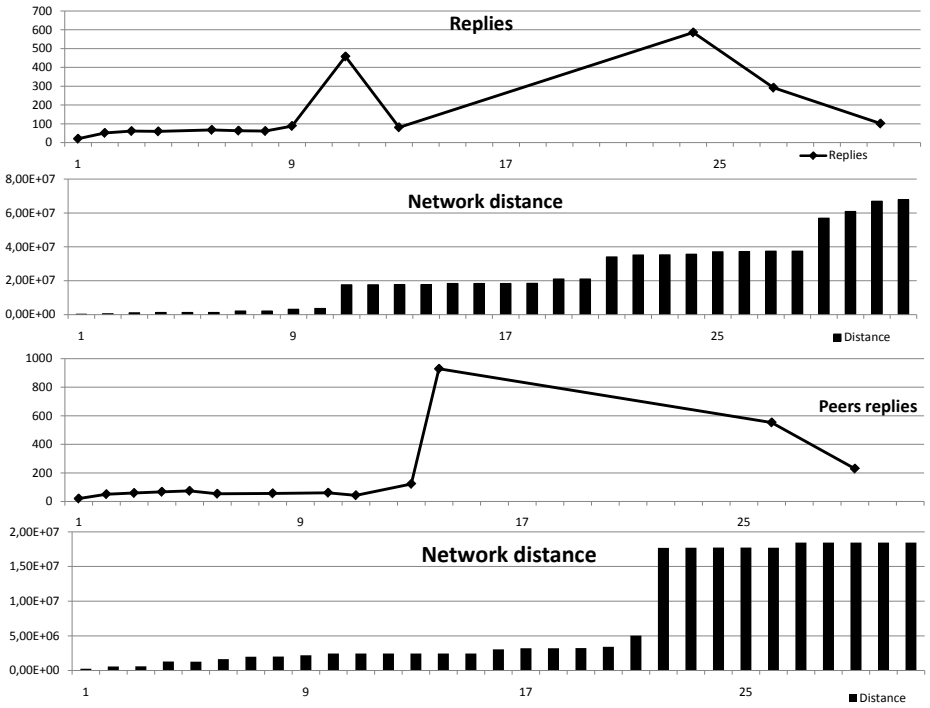

FIGURE 3.6 – Distance IP des pairs

suivre un fonctionnement hasardeux. Un tel comportement peut être vu dans la figure 3.7, qui présente le RTT moyen entre hôte 3 et sa première PL, récupérée lors de la mi-temps du même match de football France vs Espagne, déjà mentionné dans la section 3.2. En effet, souvent, les chaines qui diffusent un match de football populaire (intéressant) ont une grande popularité au cours des périodes de jeu actif, mais leur popularité diminue considérablement durant la mi-temps du jeu.

3.5.5 Géolocalisation des pairs

Jusqu'à présent, nous avons examiné deux scénarios différents : dans le premier, la popularité de la chaîne était élevée et notre client a été localisé dans la même région géographique que la majorité des pairs connectés à la chaîne, et dans le second, la popularité de la chaîne était faible et notre client était toujours situé géographiquement proche des pairs connectés à ce canal. Dans le premier scénario nous avons pu observer une corrélation entre la distance de Hamming et le RTT (le RTT pour les dix premiers pairs est généralement plus petit que celui des 22 pairs restants), tandis que dans le second scénario, nous étions incapables de trouver une corrélation entre le RTT et la distance IP des pairs. Dans cette section, nous cherchons à analyser un troisième scénario : popularité de la chaîne élevée, par contre le client est géographiquement loin du reste des pairs connectés à la chaîne. Ainsi, notre client recevra toujours une PL contenant 32 adresses différentes et qui auront la plus petite distance de Hamming avec le nouvel arrivant. Afin d'analyser ce cas, nous avons déployé un client Sopcast dans un réseau situé en Asie, se connectant à

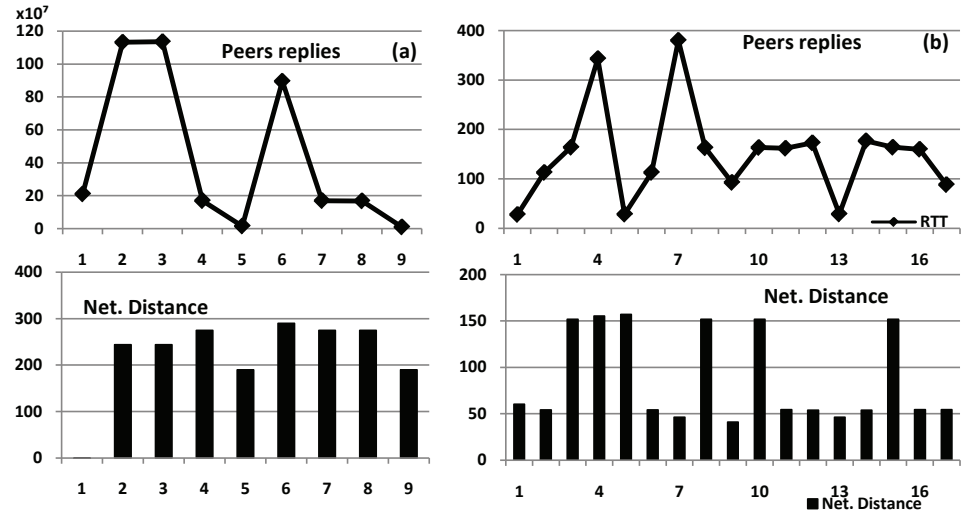

FIGURE 3.7 – Réponse des pairs VS. Distance réseau

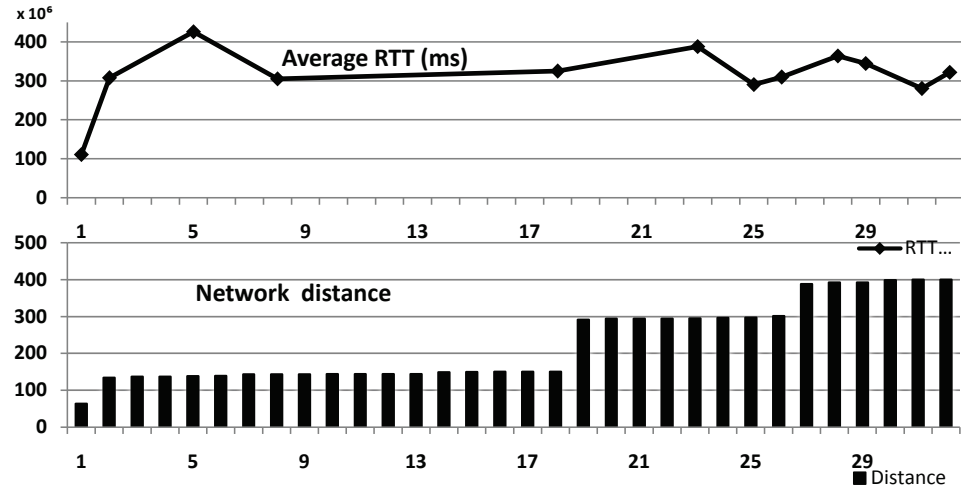

FIGURE 3.8 – Réponse des pairs VS. Distance réseau

une chaîne européenne (DiGi Sport 2, qui est une chaîne roumaine) qui diffuse un match de football de la Ligue roumaine. Le nombre de pairs connectés était proche de 14.500, exprimé par environ 0.52 en indice de popularité selon Sopcast.

Nos expérimentations montrent que, lorsque le pair demandant une connexion est géographiquement loin des sources, la politique de distance de Hamming n'a pas un impact évident sur la QoS / QoE. En effet, nous avons été incapables de trouver une fois de plus une corrélation entre le RTT expérimenté et la distance de Hamming entre le client et les pairs à qui il était connecté initialement.

En conclusion, nous pouvons dire que l'algorithme de sélection de pair appliqué par Sopcast n'est pertinent que lorsque le nombre de pairs est élevé et que le pair qui demande à rejoindre une chaîne est géographiquement proche des autres pairs déjà qui y sont déjà connectés.

Cependant, la localisation géographique des pairs devrait être le principal critère pour créer la première PL pour les clients loin de la majeure partie des pairs connectés à la chaîne. Nous convenons que la prise en compte de la situation géographique de construire un PL introduirait plus de traitement pour le tracker. Cependant, le calcul de la distance de Hamming moyenne entre le nouveau client et sa PL récupérée, et en comparant cette moyenne à un seuil donné, peut effectivement donner une estimation rapide si le nouveau client est loin des pairs présents. Si c'est le cas, le tracker devait exécuter un algorithme pour la géolocalisation pour envoyer une première PL.

3.5.6 Impact de la distance des pairs

Nous avons mené une autre expérience pour confirmer les deux aspects présentés dans les sections 3.5.3 et 3.5.4. Nous pouvons observer que la métrique distance affecte la sélection de la PL et a un réel impact sur le démarrage de Sopcast. Nous avons étudié ce problème en mesurant les paquets de données reçus de la PL. L'idée principale est de confirmer que les pairs compris dans la distance de la PL (la première valeur de distance IP envoyée par le tracker et la dernière dans la PL) contribuent au (bon) démarrage correct de la chaîne. Nous avons choisi un match de Premier League diffusé en Octobre 2012, Manchester United Vs Chelsea, diffusé dans de chaîne différentes, EuroSport (en anglais) et DigiSport 2 (en roumain). L'expérience a duré pendant environ 8 minutes. Nous utilisons Tcpdump pour permettre une surveillance passive du trafic transmis aux pairs Sopcast. Sur la première chaîne, nous avons reçu 77% des paquets significatifs (plus de 10) des pairs au sein de la valeur de la distance PL, tandis que sur la deuxième nous avons seulement reçu 10,67%. Les figures 3.9 et 3.10 présentent les paquets de données reçues de ces pairs. Nous pouvons remarquer que sur la figure 3.9 la courbe n'a pas atteint une valeur nulle, ce qui signifie qu'il n'y a pas eu d'image gelée ou pixélisée ainsi que les paquets reçus sont plus considérables que sur la figure 3.10 ou la courbe a atteint la valeur nulle à plusieurs reprises.

L'aspect culturel est très apparent dans ce cas. Sur la première chaîne, l'audience était d'environ 7000 spectateur alors qu'elle était de 9742 pour la seconde. Normalement, nous devrions recevoir plus de cette dernière, mais la diversité des pairs - en raison de l'aspect culturel - était favorable et a permis à notre adresse IP de s'insérer dans un ensemble d'adresses IP appropriée. La métrique de distance sur la seconde chaîne nous a placé dans une plage d'adresse IP où les pairs étaient localisés en Roumanie et étaient proches les uns des autres. Dans ce cas, même si les pairs nous envoyaient leurs PL (des swarms non diversifiés) nous étions toujours dans la même masse (des IP proches les une des autres, localisée en Roumanie) et ne pouvions pas échanger avec d'autres pairs.

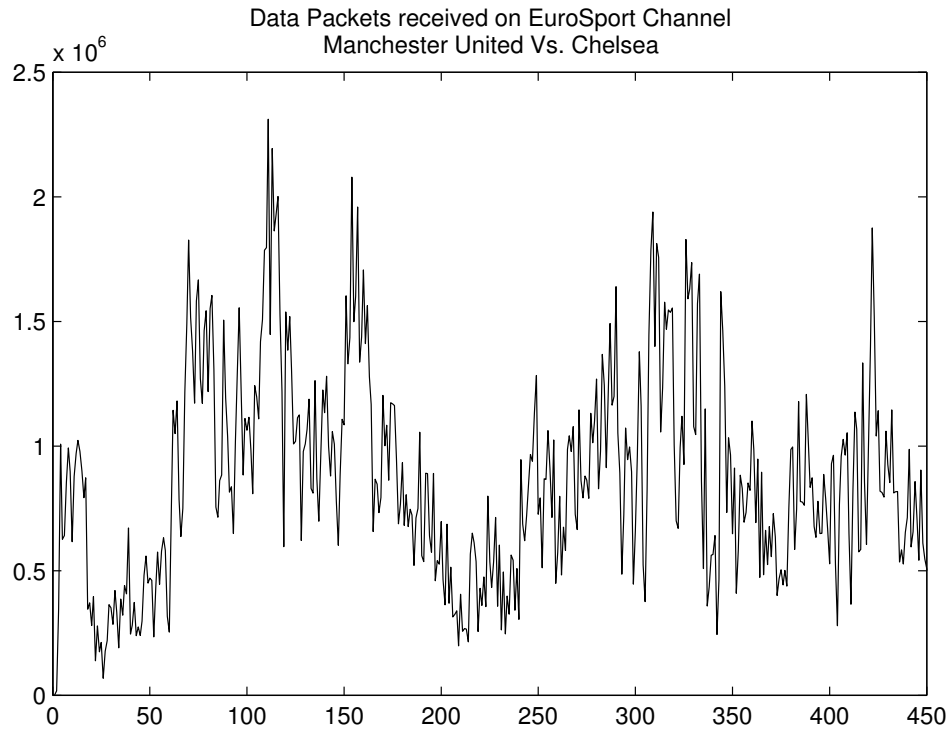

FIGURE 3.9 – Paquets reçus de la chaîne Eurosport : Manchester Vs. Chelsea

3.6 Conclusion

Des études antérieures sur Sopcast ont souligné qu'il s'agit de l'une des applications P2PTV les plus dominantes sur Internet. En 2011, Sopcast a modifié l'algorithme utilisé pour construire le voisinage d'un nouveau pair dans le swarm. Cet algorithme est un point crucial dans le fonctionnement d'un protocole P2PTV puisqu'il est fortement liée à son temps de démarrage, à savoir, le temps nécessaire pour afficher le flux vidéo d'une chaîne. Nous avons mené une série d'expérimentations afin de comprendre les caractéristiques de ce nouvel algorithme utilisé par un pair pour rejoindre une chaîne.

Nous avons d'abord démontré que ce nouvel algorithme repose sur la distance entre les adresses IP des nouveaux pairs arrivants et ceux des pairs déjà présents dans le réseau. L'algorithme est exécuté au niveau du tracker, apparement sans prendre en compte si certains pairs sont derrière un pare-feu.

Nous avons exposé le degré de corrélation entre la distance IP et la distance réseau réelle, mesurée en utilisant différents ping. Cette corrélation dépend à la fois de la taille du flux (le nombre de pairs connectés) et la relation géographique entre le nouveau pair qui rejoint une chaîne et le contenu lui-même. Si le nouveau pair est loin de la source du contenu ainsi que des pairs connectés à ce flux, la phase de démarrage pour ce dernier peut être longue puisqu'il va lui falloir plus de temps pour trouver des voisins, à l'aide de l'algorithme PEX (échange de liste de pairs

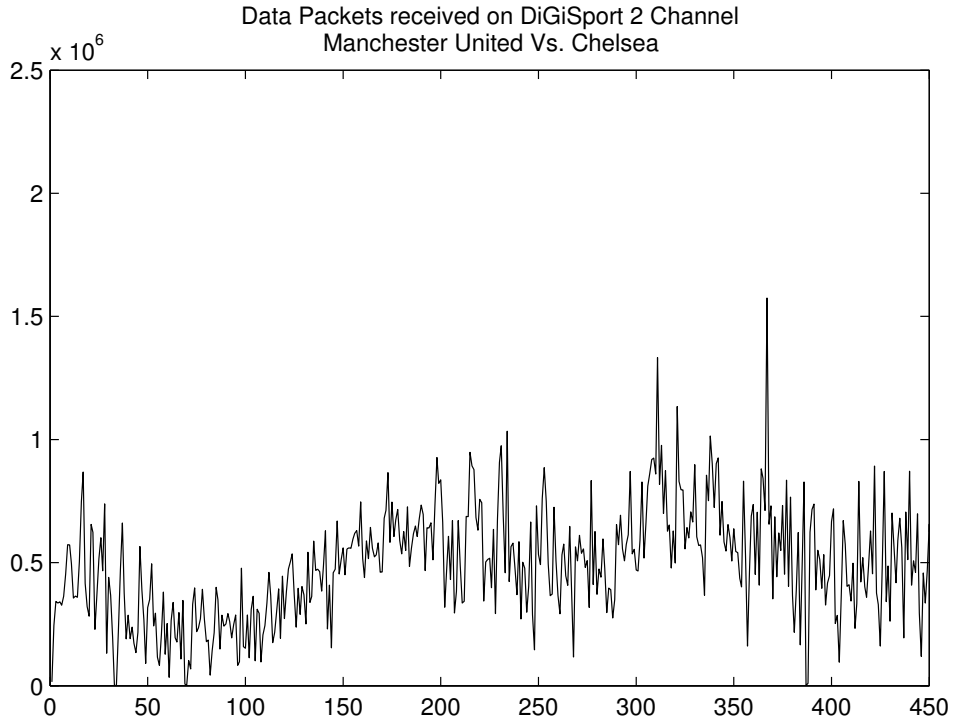

FIGURE 3.10 – Paquets reçus de la chaîne DigiSport 2 : Manchester Vs. Chelsea

entre voisins).

Le protocole Sopcast est un protocole fermé. Peu de temps après notre étude, son algorithme de construction du voisinage a a nouveau été modifié et nos conclusions sont donc fragiles. Afin de pouvoir tester de nouvelles idées pour améliorer la diffusion pair-à-pair, nous avons suivi, dans la suite de nos travaux, une approche différente en se basant sur le protocole de diffusion de contenu dominant : BitTorrent. En effet, le chapitre 2 a montré que BitTorrent est un protocole populaire pour les échanges de grand contenu (malgré la baisse d'utilisation due à Hadopi) et le présent chapitre a montré qu'un des protocoles Live les plus populaires est basé sur un algorithme voisin de BitTorrent.

Chapitre 4

Gestion collaborative d'un protocole de distribution P2P

4.1 Introduction

Une des caractéristiques les plus séduisantes de ces réseaux est le principe d'égalité entre pairs. Cependant plusieurs travaux [74],[62] montrent que les pairs sont victimes de cette politique. Par exemple dans les réseaux comme eDonkey[62], les nœuds puissants sont mis en file d'attente comme n'importe quel autre nœud, ce qui les rend passifs et leur puissance inexploitée. Au contraire, dans d'autres cas, la puissance de la bande passante d'un pair peut le privilégier par rapport aux autres. Ces pairs plus puissants peuvent alors avoir un comportement égoïste et ne s'investiront pas dans la survie du réseau. Cela engendrera la formation de clusters (groupes de pairs), qui se trouveront plus largement avancés en termes de téléchargement que d'autres. D'autre part, dans certains réseaux la notion de super nœud est inexistante, les pairs « puissants » dans ce cas ne contribueront pas à la survie du réseau en cherchant à aider les pairs en difficulté de téléchargement (dû à leurs taux de téléchargement faible).

Nous retenons de nos études précédentes que les réseaux P2P basés sur BitTorrent sont efficaces tant pour les grands transferts (chapitre 2) que pour le streaming (chapitre 3). Dans BitTorrent, le tracker [47] assure le simple rôle d'annuaire. Il envoie une liste initiale de pairs à laquelle le pair peut se connecter et collecte passivement les informations (nombres de client connectés, la taille du contenu et l'info hash pour le file sharing, le time stamp où le contenu s'est rendu dans le cas du live sharing…). Les nœuds ont, au final, une vision locale de l'état du réseau : les pairs n'ont pas une vision mis à jour et unique (unifiée) de l'ensemble du réseau, Ils ne savent pas où le swarm se situe dans le téléchargement de pièces, la pièce la plus rare globlament, le taux de téléchargement moyen, la puissance globale du swarm et sa taille réelle. Ce manque d'information limitera leurs actions dans et sur le réseau.

La façon dont ces informations (méta-données) sur l'état des pairs sont disséminées sur le réseau est un facteur important qui peut influencer les performances

d'un réseau P2P. Pour des raisons de passage à l'échelle, cette vision est locale, c'est-à-dire qu'un pair échange de l'information avec les pairs auxquels il est directement connecté. Si cette information n'est pas suffisamment de bonne qualité, cela ne permettra pas aux pairs de s'investir au vrai niveau de leurs capacités, pour maximiser les performances globales du réseau. Il y a donc un lien direct entre la qualité des méta-données diffusées entre les pairs et les performances globales du réseau.

Nous allons proposer et étudier, dans ce chapitre et le suivant des méthodes pour enrichir les méta-données, tout en gardant une diffusion locale entre voisins, et ansi améliorer les performances globales d'un swarm BitTorrent.

4.2 Proposition d'un protocole de partage P2P basée sur une gestion collaborative

Dans ce chapitre, nous illustrons cette idée de l'importance de la qualité des méta-données en prenant comme exemple la pièce la plus rare du réseau telle qu'utilisée dans un protocole de réplication comme BitTorrent[83]. Dans le protocole BitTorrent original, un pair ne fournit d'information à ses voisins [1] que sur les pièces qu'il a téléchargé et non sa propre vision locale qui serait obtenue en sommant sur l'ensemble de ses voisins. La modifications que nous allons étudier dans un premier temps dans ce chapitre, consiste à ce qu'un pair transmette des informations sur les pièces qu'il possède enrichie de l'information sur la pièce qu'il voit comme la plus rare. Nous allons illustrer l'intérêt de cette modification protocolaire dans deux cas d'utilisation d'un protocole de type BitTorrent, à savoir la distribution de contenu enregistré (file sharing) et la distribution de contenu temps réel (live broadcast)

4.2.1 Description

Nous proposons dans ce qui suit, une procédure de gestion d'un réseau de partage de contenu multimédia où les pairs échangent localement des informations dans le but d'obtenir une vision globale la plus exacte possible du réseau et choisir leur posture au sein du réseau (figure 4.1). Un ensemble de paramètres est échangé entre les pairs pour leur permettre de construire une vision précise de l'état global du réseau. Ces paramètres peuvent inclure des informations sur le contenu partagé (taux de téléchargement, pièces les plus rares sur le réseau,...), des informations sur le réseau (Bande passante globale, nombre de pairs, indice sur la qualité des connexions,...) ainsi que des informations sur la sécurité (ratio Upload/Download, indice de fausses pièces,...). Un des messages dans lequel ces informations peuvent être insérées est le message *Have*.

1. Il faut noter que les voisinages de 2 voisins en BitTorrent sont a priori distincts.

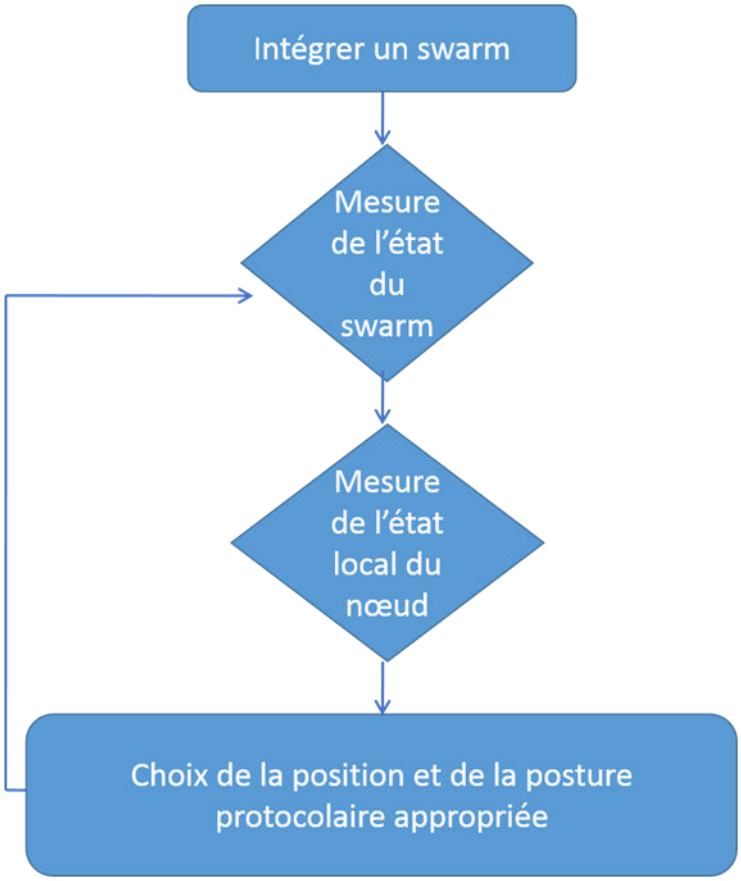

FIGURE 4.1 – Comportement d'un nouveau pair lors de sa connexion

Après chaque téléchargement d'une pièce, le pair recalcule à partir de ses informations et les informations incluses dans les messages *Have* reçus de son voisinage les nouveaux paramètres du réseau et les envoie à son tour à son voisinage dans un message Have (figure 4.2). En fonction de son état (CPU, Bande passante, qualité de connexion, ...) et des informations du réseau, qui s'affinent et se précisent après chaque téléchargement de pièce, le pair choisira le fonctionnement protocolaire le plus adéquat pour une meilleure qualité de service.

La méthode proposée d'échange d'informations entre pairs permettra d'apporter une information précise non seulement du voisinage mais de l'état du réseau dans sa globalité. A partir de cette information et en fonction de son état, le pair agira avec l'attitude protocolaire la plus adaptée à sa position et à la situation du réseau. Dans ce procédé, le protocole n'est pas figé ni imposé sur les pairs mais plutôt mis en place par les pairs et modifiable en fonction de leurs états et de l'état du réseau. Nous présentons dans ce qui suit les méthodes d'estimation de quelques paramètres ainsi qu'une partie des postures que les pairs peuvent prendre en fonction de leurs états et de l'état du réseau.

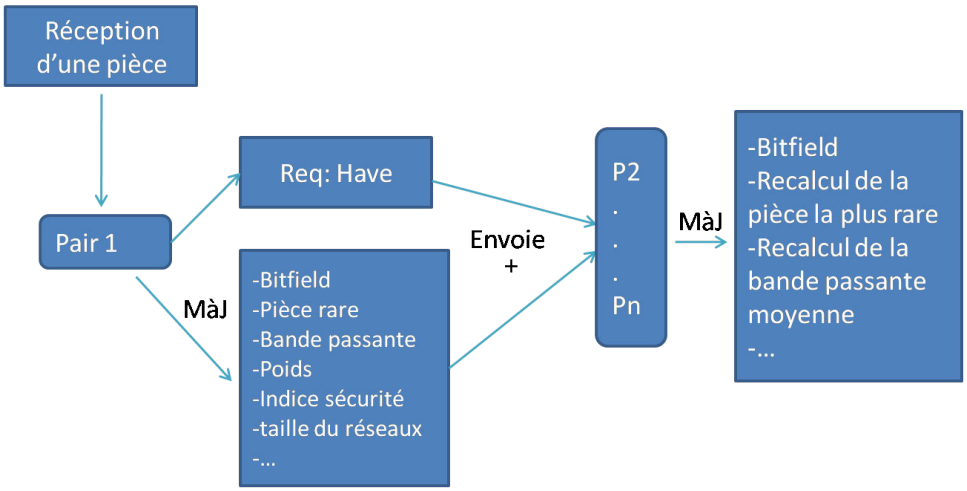

FIGURE 4.2 – Processus d'échange entre les pairs du réseau

4.3 Calcul des paramètres réseaux

Parmi les paramètres réseaux qui peuvent être estimés : les pièces les plus rares et la bande passante.

4.3.1 Les pièces les plus rares

Sur un réseau d'échange BitTorrent de contenus, les pairs peuvent avoir quelques pièces, aucune ou la totalité des pièces d'un contenu. Au premier contact les pairs signalent, par un message Bitfield, les pièces déjà téléchargées. Il s'agit d'un vecteur de taille le nombre de pièce totale avec un 0 pour une pièce manquante et de 1 pour une pièce téléchargée. Par la suite, à chaque téléchargement de pièce ils informent leurs voisinages par un message Have de l'indice de la pièce téléchargée. Nous proposons que le pair inclût également en plus de l'indice de la pièce téléchargée, l'indice de son estimation de la pièce la plus rare du réseau. Le calcul de la pièce la plus rare sur le réseau se fait par rapport à l'ensemble des Bitfields stockés localement ainsi que les indices des pièces rares reçus dans les messages Have du voisinage. A noter que cette approche diffère de la version habituelle de BitTorrent où chaque voisin enfin son propre Bitfields et le récepteur ne peut qu'estimer la pièce la plus rare localement.

4.3.2 La bande passante

De la même manière un calcul de la bande passante peut-être fait et transmis aux autres pairs voisins dans le Have. À l'instant où le pair commence à télécharger des pièces une estimation de sa bande passante est faite (débit montant et descendant). Ceci va permettre d'estimer la bande passante moyenne du réseau à partir de la bande passante calculée localement et les bandes passantes reçues dans les messages Have du voisinage. Nous pouvons intégrer également le poids comme

information indiquant le nombre de pairs qui y ont contribués. La bande passante et les pièces les plus rares du réseau serviront de moyen d'orientation : un pair qui partage une moyenne élevée rapporte "le confort" de son voisinage. Ceci veut dire que la vitesse de téléchargement (la bande passante) de son voisinage est élevée. Les pairs ayant des difficultés à démarrer d'une manière cohérente leurs téléchargements s'orienteront par eux même vers ces derniers pour pouvoir suivre la vitesse de téléchargement du réseau. D'autres paramètres réseau (taille du réseau, indices de sécurité, temps d'attente, qualité de connexions, file d'attente...) peuvent être estimés et échangés dans le message Have pour apporter une information précise du réseau à chaque pair. **Nous allons nous concentrer dans ce chapitre et le chapitre V, sur la pièce rare du réseau et la bande passante.**

4.4 Exemple de modification protocolaire : La pièce rare du réseau

Au premier contact, les pairs signalent, par un message *Bitfield*, les pièces déjà en leur possession (téléchargées). Par la suite, à chaque téléchargement d'une nouvelle pièce les pairs informent leurs voisinage par un message *Have* qui indique l'indice de cette pièce. Nous proposons d'ajouter dans le message *Have* l'indice de l'estimation de la pièce rare calculée par chaque pair. Ce calcul est fait par rapport à l'ensemble des Bitfields stockés localement ainsi que les indices des pièces rares reçus dans les messages *Have* du voisinage. Dans ce qui suit nous présentons un exemple qui permet d'illustrer la différence entre la méthode actuelle dans BitTorrent de choix de la pièce demandée et notre protocole modifié.

4.4.1 Exemple de calcul

Comme indiqué précédemment, un pair maintient un *Bitfield* par voisin mis à jour après chaque réception de message *Have*. Le calcul par défaut dans BitTorrent de l'indice de la pièce rare (Idpr) se fait de la manière suivante :

$$Idpr = \text{argmin}_{\text{indices des valeurs}>0}(\sum bitfields)$$

La formule précédente permet de sélectionner l'ensemble des pièces ayant le moins de réplicas parmi les pièces disponibles (d'où la condition sur les valeurs positives) dans le voisinage. Soit l'exemple suivant avec 3 pairs :

Bitfield P1	1111111011111111
Bitfield P2	1000000011101001
Bitfield P3	0011100011100110
Somme des BitFields	21222110333122222

Nous obtiendrons :

$$\text{Idpr}_{\text{orig.}} = \{2, 6, 7, 12\}$$

qui sont les indices des pièces ayant un seul réplica. Le client choisira alors arbitrairement un de ces indices.

Dans la variante que nous proposons, les pairs insèrent dans le message Have la pièce qu'ils considèrent comme étant la plus rare. Supposons, dans l'exemple précédent que P1, P2 et P3 suivent notre protocole modifié et envoient les messages Have suivants avec la nouvelle pièce téléchargée suivi de la pièce qu'ils considèrent comme la plus rare :

— P1(1,5)
— P2(11,9)
— P3(3,13)

Dans ce cas, le pair P0 va recalculer l'Idpr comme suit : Premièrement, le pair va recalculer la somme des Bitfields en intégrant le sien : tableau 4.1

Bitfield P0	1000000000000000
Bitfield P1	1111111011111111
Bitfield P2	1000000011101001
Bitfield P3	0011100011100110
Somme	3122211033312222

TABLE 4.1 – Bitfield : étape 1 de calcul

Deuxièmement, il prendra en compte les pièces rares signalées par les pairs P1, P2 et P3 en décrémentant l'indice de la pièce signalée comme rare par ces derniers : : tableau 4.2

Ancienne vision globale des indices	3122211033312222
Nouvelle vision globale des indices	3122111023311222

TABLE 4.2 – Bitfield : étape 2 de calcul

Lors de la décrémentation, nous empêchons l'apparition de valeurs négatives. Par contre, nous différencions une pièce à 0 car aucun répliqua n'existe dans le voisinage d'une pièce à 0 pour lesquelles un ou plusieurs réplicas existent mais qui a été décrémentée à 0. Un pair cherchera alors à obtenir les pièces les plus rares parmi celles ayant une valeur positive ou une valeur à 0 due à la décrémentation. Pour l'exemple ci-dessus, nous choisirons dans les indices :

$$\text{Idpr}_{\text{modif.}} = \{2, \underline{5}, 6, 7, 12, \underline{13}\}$$

La différence entre $\text{Idpr}_{\text{orig.}}$ et $\text{Idpr}_{\text{modif.}}$ est donc la présence de deux nouvelles pièces que le pair tentera de télécharger. Elles sont apparues dans l'$\text{Idpr}_{\text{modif.}}$ car elles étaient considérées comme rares par certains pairs du voisinage. Nous allons démontrer dans la section suivante que cette simple modification peut jouer sur les performances d'un protocole comme BitTorrent en adoptant des scénarios réalistes. Nous pouvons néanmoins faire une première remarque : l'impact de cette modification n'aura que peu d'incidence au début du téléchargement, où toutes les pièces sont rares. C'est une fois que les pièces commencent à être largement diffusées

entre les leechs (pairs n'ayant pas la totalité du contenu) que l'impact de notre modification protocolaire peut faire une différence par rapport à l'approche par défaut de BitTorrent.

4.5 Plateforme de test

La plateforme de test est constituée d'un serveur Linux ayant les spécifications suivantes : Intel(R) Pentium(R) 4 CPU 2.80GHz et 1.5 Go de RAM. Ce serveur nous permet de faire tourner 50 clients BitTorrent originaux ou modifiés. Nous utilisons la version mainline 3.9.1. Nous adoptons une solution de virtualisation réseau pour séparer les clients en reliant chaque client à une interface réseau virtuelle à l'aide de l'application Linux iptables. Cette dernière nous permet également de contraindre les débits descendants des pairs. Nous simulons le réseau par une multitude d'alias sur la carte réseau locale du serveur test. Par la suite, nous associons à chaque client du serveur le trafic d'un alias par l'intermédiaire de règles IPtables.

Exemple :

Un client sim1001 est associé à l'interface eth1 :1001 (IP :192.168.209.1). Nous associons la marque 0x3e9 aux paquets qui ont été générés par sim1001 :

```
iptables -L -t mangle MARK  all  --  anywhere   anywhere  owner UID match sim1001 MARK set 0x3e9
```

Nous définissons la règle mangle :

```
iptables -A OUTPUT -t mangle -m owner --uid-owner sim1001  -j MARK --set-mark 0x3e9
```

Les paquets qui ont la marque 0x3e9 sont modifiés de façon à définir l'IP source comme étant 192.168.209.1 :

```
iptables -A SNAT       all  --  anywhere   anywhere   mark match 0x3e9 to:192.168.209.1
```

Et pour définir la règle NAT :

```
iptables -A POSTROUTING -t nat -m mark --mark 0x3e9  -j SNAT --to-source 192.168.209.1
```

A noter que cette approche de virtualisation réseau est moins lourde qu'une approche de virtualisation basée sur des hyperviseurs (VMWare, Xen,...) ou même sur des containers (OpenVZ, LXC,...) et permet d'effectivement faire tourner 50 clients sur un serveur aux performances modestes, sans être contraints par les performances du matériel. Nous utiliserons néanmoins un serveur plus puissants pour certaines expérimentations (voir section 4.7.1). Nous considérons deux scénarios. Le premier correspond à un fichier de 39 Mo pour les expérimentations concernant la partie Live. Cette taille correspond à la taille standard d'un buffer HD (tampon) en streaming vidéo. Le second scénario correspond à un deuxième fichier AVI de 316 Mo pour la partie file sharing. Le choix de faire des tests de type live avec un protocole conçu pour le file sharing n'est pas du au hasard. Tout d'abord, de nombreux protocoles de distribution P2P live utilisent une approche similaire à BitTorrent comme l'organisation en maille des pairs, par exemple PPlive, Sopcast (les codes sources de ces logiciels restent propriétaires) ou dérivent directement de

BitTorrent, tel Tribler [75] . En second lieu, l'approche suivie est toujours d'avoir une fenêtre active (ce que nous avons appelé un buffer) qu'il convient de remplir le plus vite possible. En regardant un contenu de la taille typique d'un buffer, nous pourrons donc tirer des enseignements quant à l'impact de notre modification de l'algorithme de dispersion de l'information sur la pièce la plus rare dans un cas P2P streaming. Les pairs en réseau P2P ne s'échangent pas directement un fichier. Ce fichier est préalablement découpé en pièces de taille fixe et ce sont ces pièces que les pairs s'échangent entre eux. La taille des pièces va être fixée entre 64 et 256 Ko pour le cas du live 128 à 1024 Ko dans le cas du file sharing. Les 50 pairs sont divisés en deux catégories : les sources ou *seeds* qui ont une copie complète du fichier au début de l'expérimentation et les clients ou *leechs* qui ne possèdent initialement aucune pièce. Nous avons fixé arbitrairement le nombre de seeds et leechs à 3 et 47 respectivement. Les expérimentations se font au sein d'un réseau fermé pour éviter tout bruit dû à un trafic externe. Un autre élément important est le débit de chaque pair. Pour le cas du live, nous avons d'abord considéré un cas homogène où les pairs ont tous le même débit. Plus précisément, le débit montant des leechs est limité à 50Ko/s et celui des seeds est limité à 200Ko/s. L'étude du cas homogène est intéressante pour montrer l'impact de nos modifications protocolaires introduites dans la section suivante et les séparer clairement d'un effet dû aux différences de performances des pairs.

Nous avons ensuite considéré un cas plus réaliste en terme de distribution des débits en prenant un scénario hétérogène :

— 12 leechs à 10 Ko/s
— 12 à 20Ko/s
— 12 à 30Ko/s
— 11 leechs à 50Ko/s
— des seeds à 200Ko/s.

Pour le file sharing, nous avons considéré seulement un scénario hétérogène en augmentant les débits compte tenu de la taille plus importante du contenu à partager :

— 12 leechs à 400 Ko/s
— 12 à 300Ko/s
— 12 à 1600Ko/s
— 11 leechs à 100Ko/s
— des seeds à 800Ko/s.

Un dernier paramètre important est le nombre de connexions totales/entrantes par pair. Nous avons limité à 10 ces 2 valeurs. Les valeurs par défaut pour BitTorrent sont 80 et 40 pour ces 2 paramètres, mais puisque nous travaillons avec 50 pairs, nous avons réduit ces valeurs.

Avant de rentrer dans le détail des résultats, nous devons mettre l'accent sur une spécificité de BitTorrent que nous avons découvert lors des expérimentations en environnement contrôlé : son comportement (relativement) déterministe, à savoir que pour une configuration donnée en nombre de pairs et ordre d'arrivée, le protocole

va donner la même trajectoire, ou du moins une trajectoire très similaire, à chaque fois. La figure 4.3 illustre cette observation pour le cas de 5 expérimentations faites dans les mêmes conditions pour le protocole BitTorrent original sur notre plate-forme de test. Les versions modifiées que nous présentons dans cette thèse offrent le même comportement. L'ensemble des résultats montrés dans ce chapitre et le suivant sont affectés par ce phénomène que nous avons découvert a posteriori. Nous n'avons pas eu le temps de modifier notre protocole expérimental pour tenir compte de cette spécificité de BitTorrent. Même si cela est regrettable, nous notons tout de même que ce résultat est intéressant dans le sens où il est contre-intuitif puisque BitTorrent est censé avoir des sources d'entropie internes, notamment son algorithme *choke* qui fait un *optimistic unchoke* (choix au hasard d'un pair à qui envoyer des données sans tenir compte de ce qu'il a envoyé) ou l'algorithme *rarest-first* qui doit faire un choix au hasard si plusieurs pièces sont également rares.

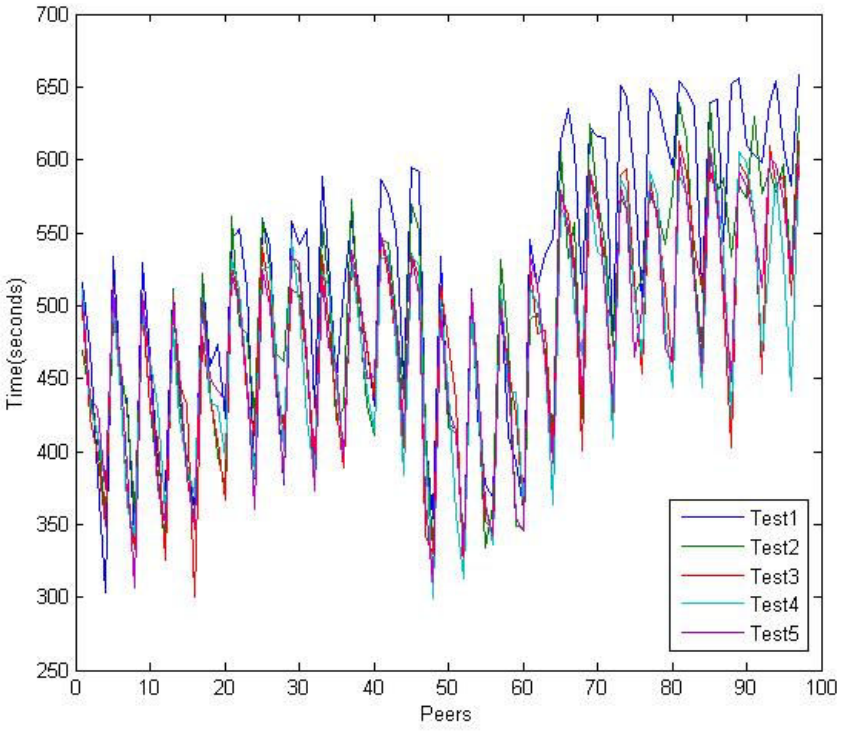

FIGURE 4.3 – Déterminisme de BitTorrent - protocole original non modifié

4.6 Résultats

Nous présentons dans cette section une comparaison de performance entre le protocole de distribution original et le protocole modifié quant au choix de la pièce rare. Les métriques que nous regardons sont :

— Temps de téléchargement : représente le temps complet de téléchargement des leechs. Nous nous intéressons aux résultats de chaque leech, mais aussi à la moyenne et à la variance entre leechs. Dans le cas des expérimentations live, la variance est une métrique très importante car elle représente une mesure du retard des pairs.
— Débits : le débit considéré ici est le débit moyen de téléchargement par pièce. Chaque client retourne à la fin du téléchargement de chaque pièce, le débit obtenu pour cette pièce. Nous moyennons ensuite sur toutes les pièces. Il est important de comprendre que ce débit par client ne tient pas compte du temps d'attente potentiel passé entre chaque pièce. Par exemple, si un client télécharge une première pièce avec un débit de 100 Ko/s puis attend 10 s et télécharge une seconde pièce avec un débit de 100 Ko/s à nouveau, son débit moyen sera de 100 Ko/s. Le temps d'attente de 10s ne sera pas pris en compte dans la mesure de débit, par contre il influera bien entendu le temps de téléchargement. D'où l'intérêt de considérer les deux métriques dans l'analyse des résultats.

4.6.1 Résultats du Live

Les premières expérimentations que nous présentons concernent la gestion d'un contenu à contraintes temporelles (Live). De manière générale, le *Live* est un flux continu de vidéo où les chunks doivent être reçus dans l'ordre afin de reconstituer une séquence d'images complète. Les protocoles live actuels progressent par fenêtre glissante (tampon ou buffer) en appliquant bien souvent une approche de type BitTorrent, c'est-à-dire en cherchant à répliquer la fenêtre courante au plus vite, sans notion d'ordre. Des mécanismes supplémentaires sont mis en œuvre si le client se rend compte qu'une pièce non encore téléchargée va être lue très prochainement. Il va alors se concentrer sur cette (ou ces) pièce(s). Ce sont ces mécanismes que nous ne prenons pas en compte dans notre étude.

Pour la survie du réseau, les protocoles P2P actuels [77], [55] appliquent une politique favorisant les pairs ayant une bande passante puissante en les servant en priorité. Ce phénomène a également été observé dans le cas du protocole BitTorrent original [66]. Pour éliminer toute préférence de ce type et pouvoir évaluer seulement l'impact de la modification apportée au protocole, nous avons choisi un premier scénario où tous les pairs sont sur un pied d'égalité avec un débit descendant de 50Ko/s pour tous les leechs. Dans l'ordre, nous lançons les seeds suivis des leechs qui vont s'auto-organiser en établissant des connections. Le tracker va délivrer à chaque nouveau leech une liste des pairs qui possèdent une partie (un autre leech) ou la totalité du contenu (un seed). Les résultats présentés ci-après sont obtenus en moyennant sur la base de quatre expérimentations pour chaque version du protocole (original et modifié). Les variations entre expérimentations sont faibles

dans notre environnement de test, ce qui justifie de ne faire que quatre expérimentations à chaque fois. Le débit et le temps de téléchargement mesurés au niveau de chaque pair sont représentés sur les figures 4.4 et 4.5 respectivement.

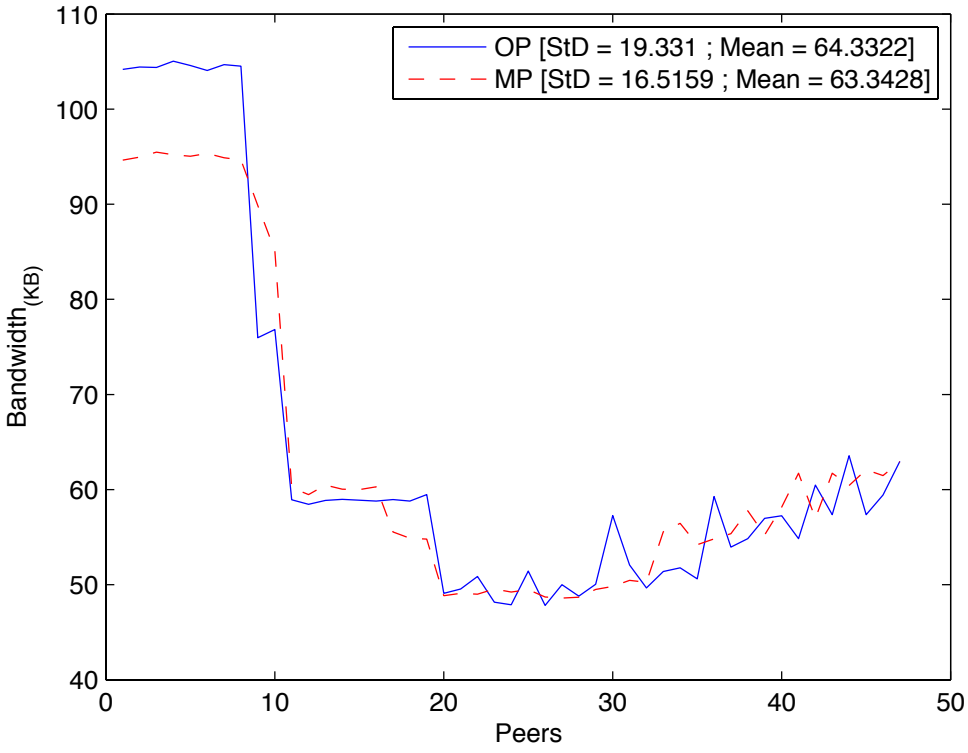

FIGURE 4.4 – Débit descendant. Expérimentation Live, chunks de 128 Ko, Débits homogènes

La gestion de la bande passante du protocole original (OP) semble plus profitable que celle du protocole modifié (MP). Comme l'indique la figure 4.4, le débit moyen par pair sur les 4 expérimentations est de 19,3 Ko/s en OP contre 16,5 Ko/s en MP. Cependant sur la figure 4.5, nous pouvons constater que MP a été plus performant en terme de temps de téléchargement. Cette différence entre débit de téléchargement par pièce et temps de téléchargement global est dû à la disponibilité des pièces : avec MP, les pairs passent moins de temps à chercher la pièce suivante à télécharger. Inciter les pairs à répandre la pièce rare du réseau fait qu'il y a une optimisation de la bande passante globale. Ceci explique l'écart-type faible entre pairs sur la figure 4.4 (std =16.51 pour MP contre 19.33 pour OP). Ce premier résultat illustre bien l'intérêt d'une gestion globale de la pièce rare dans le cas d'une réplication d'un contenu Live.4.4.

Pour analyser un scénario plus réaliste, nous avons fait varier le débit descendant des pairs. Les leechs sont organisés en groupe de 12 et ont respectivement des débits de 10, 20, 30 et 50Ko/s. Les seeds ont un débit de 200Ko/s. Ainsi avec le même contenu, toujours découpé en pièces de 128Ko, nous lançons le téléchargement et nous obtenons les résultats sur la figure 4.6 et 4.7. Avec cette configuration,

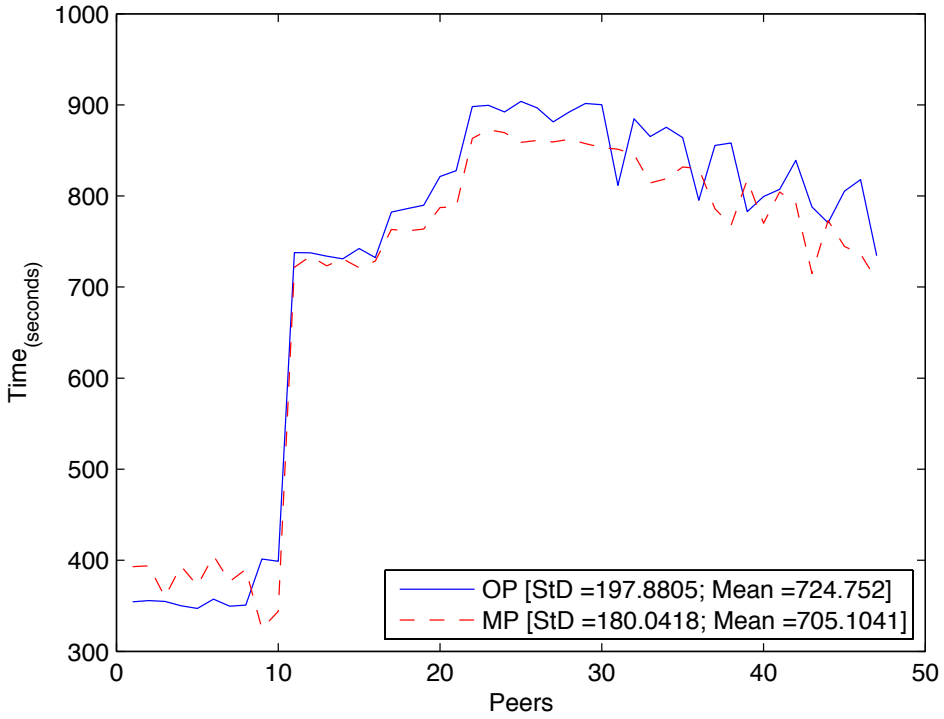

FIGURE 4.5 – Temps de téléchargement. Expérimentation Live, chunks de 128 Ko, Débits homogènes

la tendance des résultats reste la même. Le temps que met MP dans la recherche de la pièce suivante est réduit et ce gain se reflète sur le temps de téléchargement global du contenu (980.85s pour MP contre 995.79s pour OP). Plus important encore, dans le cas du live, l'écart-type du temps de téléchargement est réduit en MP (364.36s pour OP contre 310.65s pour MP) ce qui reflète un temps d'attente moindre d'une pièce à l'autre. La réduction de ce délai fait que la différence d'indice temporel au sein de la vidéo entre le premier et le dernier leech est atténuée. Nous comparons ensuite avec cette même configuration en accentuant l'impact sur le temps de téléchargement (moyenne mais surtout variance), l'effet de la gestion de la pièce rare par l'augmentation du nombre de pièces.

Nous doublons le nombre de pièces (pour le même contenu) en utilisant des pièces de 64Ko. Ceci suscitera plus de temps à chercher la pièce suivante à télécharger. Les figures 4.8 et 4.9 présentent l'effet de ce découpage sur le débit moyen par pièce et le temps de téléchargement. La différence entre MP et OP en terme de temps de téléchargement est amplifiée par rapport au scénario précédent en valeur absolue : 15 s de moins en OP dans le cas de pièces de 128 Ko et 105 s avec des pièces de 64 Ko. Le fait qu'il y ait plus de pièces impose encore plus de temps à chercher la pièce suivante. Malgré ceci, MP diminue ce temps d'attente en réduisant l'écart-type entre pairs dû à la recherche des pièces, ce qui se reflète sur les temps enregistrés par chacun des protocoles.

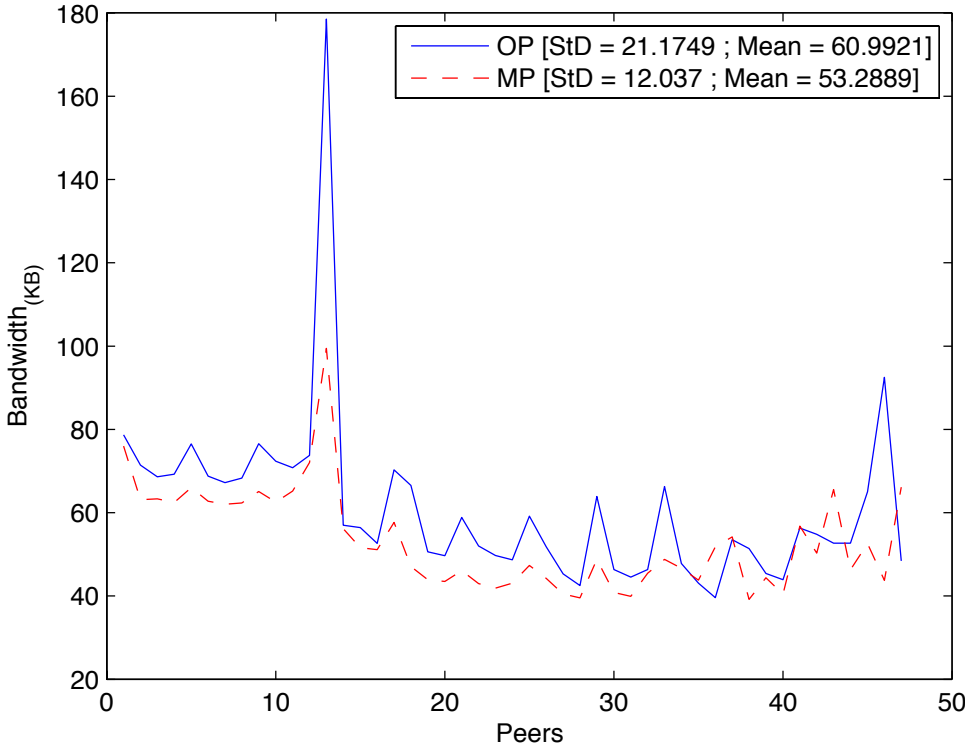

FIGURE 4.6 – Débit descendant. Expérimentation Live, chunks de 128 Ko, Débits hétérogènes

4.6.2 Conclusion sur le live

Nous déduisons des expérimentations précédentes qu'en live, la gestion de la pièce rare est très importante. Les pairs quand ils partagent leur vision locale créent une vision plus large sur les pièces manquantes dans le réseau P2P. Ce comportement permet une meilleure coopération entre pairs qui va harmoniser le téléchargement sur le réseau. La disponibilité des pièces dans ce cas permettra aux pairs de passer moins de temps dans la recherche des pièces suivantes, réduisant ainsi la différence en délai de visualisation entre les premiers arrivants et les derniers.

4.7 Résultats du file sharing

4.7.1 Ajustement de la plateforme

Dans cette partie, nous avons dû faire quelques changements sur la plateforme de test ainsi que sur les paramètres réseaux des leechs. En raison du coût élevé du téléchargement en termes de mémoire et d'espace disque, nous avons migré sur un serveur plus performant ayant les spécifications suivantes : la plateforme de test est constituée d'un serveur Linux Intel(R) Atom(TM) 8 Core, CPU 2.40GHz et 8Go de RAM. Quant aux pairs, nous avons augmenté leur débit compte tenu de la taille du fichier à répliquer et de temps nécessaire pour son téléchargement et, donc, à la

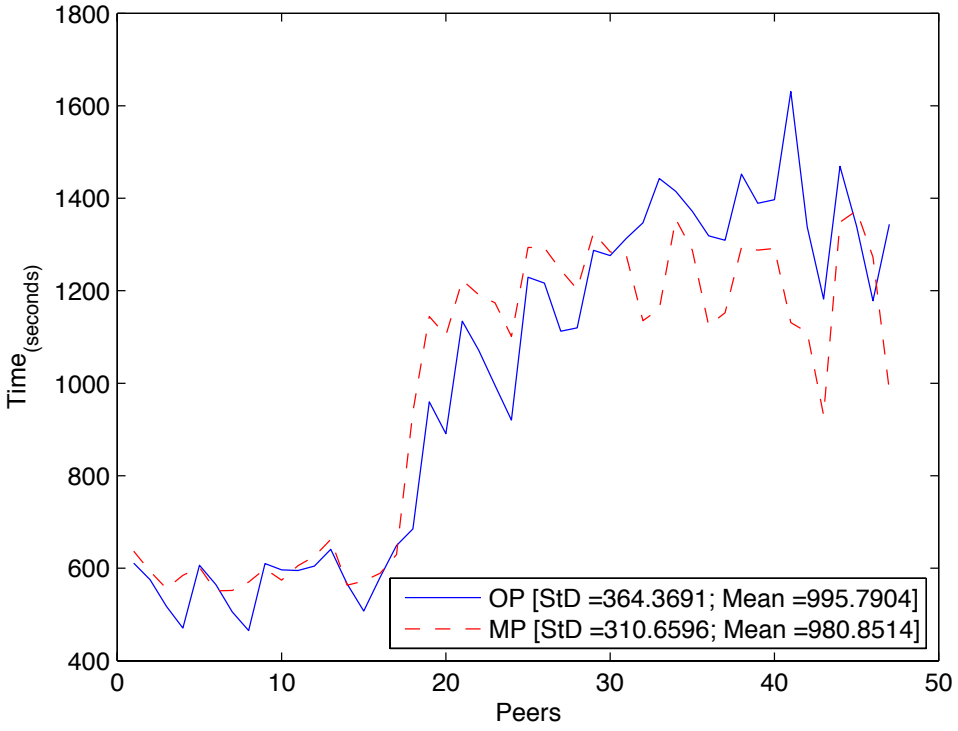

FIGURE 4.7 – Temps de téléchargement. Expérimentation Live, chunks de 128 Ko, Débits hétérogènes

réalisation des tests. Nous avons dû également ajuster le nombre de connexions entrantes/sortantes des leechs. En effet, avec 10 connexions entrantes, les pairs directement connectés aux seeds initiaux n'arrivaient pas à terminer leur téléchargement et restaient bloqués à 35% du contenu. Ce blocage est dû à la politique de choke appliquée par les seeds [54],[65] : afin d'éviter un déni de service d'un client qui monopoliserait les seeds, les seeds vont arrêter de servir les leechs initiaux après un certain temps. Avec les contraintes sur le nombre de connexions sortantes et le fait que les seeds ne ferment pas les connexions mais se contentent de ne plus transmettre, les leechs initiaux ne pouvaient plus ouvrir de nouvelles connexions. Ils se trouvaient alors à former un groupe isolé et incapable de finir le téléchargement. A noter que ce problème est dû à la limite de 10 connexions que nous avons imposé. Avec la valeur par défaut de 80, il faudrait un contenu de l'ordre de 20 Go et 400 pairs pour observer ce même blocage. Ce problème ne s'est pas manifesté dans le cas Live de par la taille plus faible du contenu. Pour éviter ce blocage dans nos expérimentation file sharing, nous avons autorisé deux connexions sortantes de plus et trois autres entrantes.

La taille des pièces pour un contenu de type file sharing est nécessairement plus importante que pour un contenu live. Nous avons choisi un contenu de 316 Mo avec un découpage de 1024 Ko. Nous aurons approximativement le même nombre de pièces qu'en Live avec le contenu de 39Mo découper en 128 Ko (316 Mo/1024Ko $\rightarrow$ 308 pièces ; 39Mo/128Ko $\rightarrow$ 304 pièces). La figure 4.11 montre qu'avec une

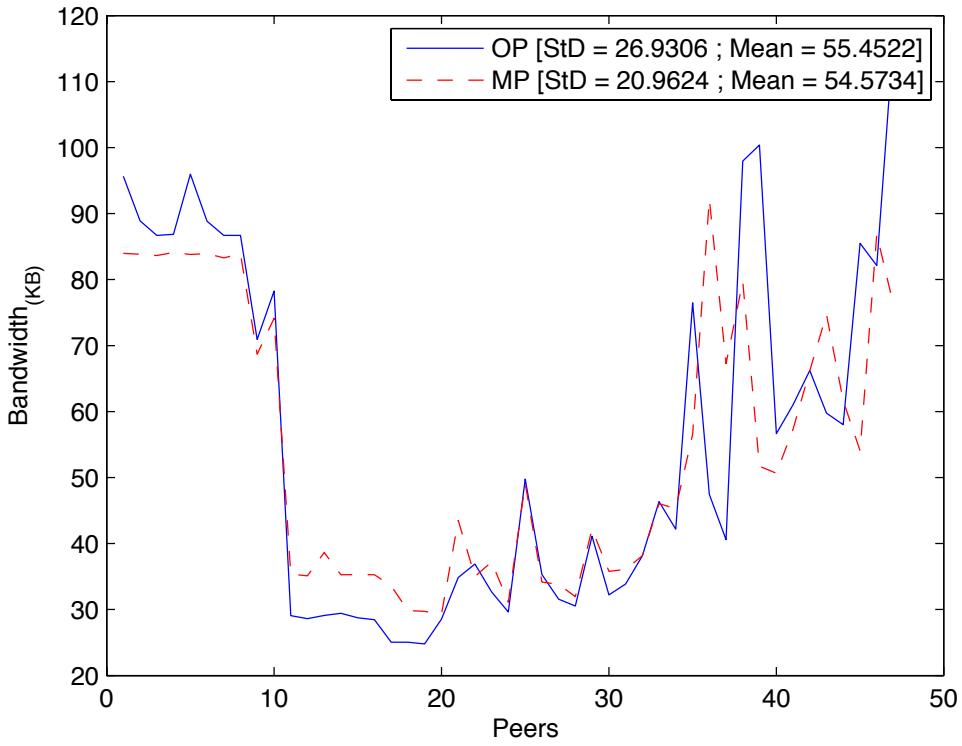

FIGURE 4.8 – Débit descendant. Expérimentation Live, chunks de 64 Ko, Débits hétérogènes

taille de contenu plus importante, le protocole original (OP) obtient un meilleur temps de téléchargement que notre version modifiée (MP). Ce résultat s'explique ainsi : Nous pouvons voir le téléchargement vu d'un pair comme une succession de périodes de téléchargement de pièces (qui éventuellement se recoupent dans le temps) et de périodes d'attente. Nous avons observé dans les expérimentations du live qu'MP engendrait une diminution des débits par pièces au profit d'une diminution du temps d'attente inter-pièces, ainsi le temps total de téléchargement s'en trouvait diminuer. L'augmentation de la taille des pièces va favoriser OP puisque le temps de téléchargement va devenir important par rapport au temps d'attente inter-pièces. C'est un tel phénomène que nous observons sur les figures 4.10 et 4.11. Pour bien mettre en évidence l'impact de la taille de la pièce, nous avons ensuite diminué la taille des pièces de 1024Ko et 512Ko. Le contenu faisant toujours 316 Mo, nous aurons maintenant 617 pièces.La figure 4.13 montre que malgré la taille du contenu, la disponibilité de la pièces rare favorise MP qui assure un meilleur temps de téléchargement. La bande passante moyenne de OP restera néanmoins toujours plus élevée figure 4.12.

4.7.2 Conclusion du file sharing

Dans le file sharing et avec un découpage de pièce plus important, OP compense le temps inter-pièce par des temps de téléchargement par pièce plus rapide que MP et ainsi un temps de téléchargement global plus faible. En revanche dès

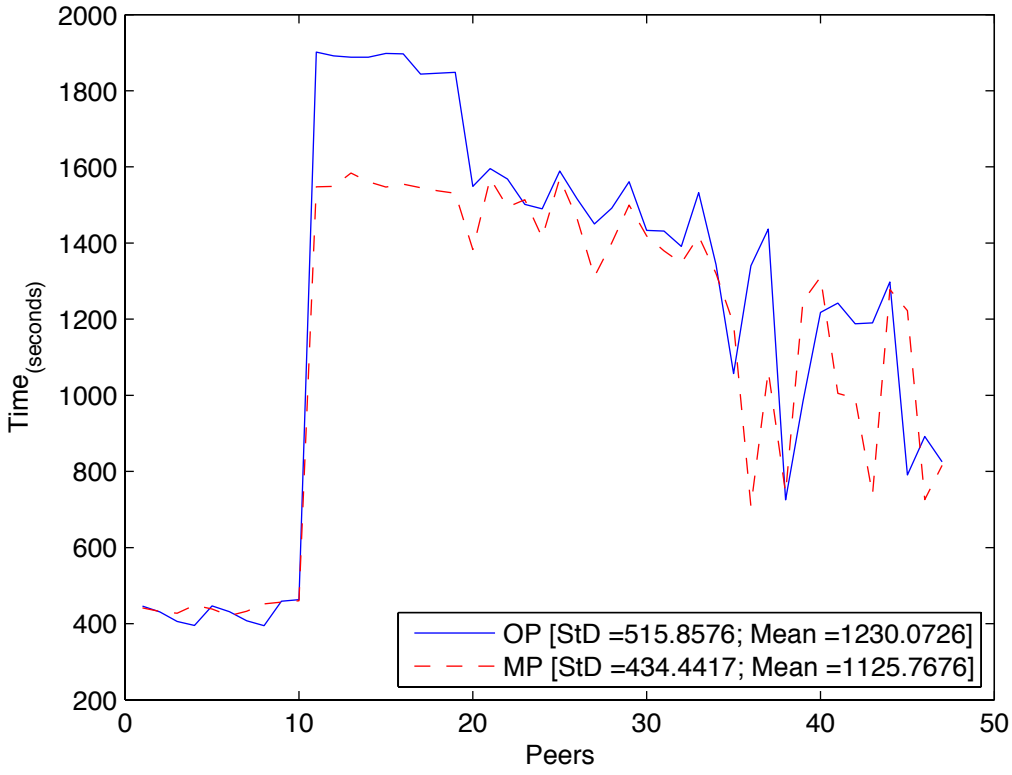

FIGURE 4.9 – Temps de téléchargement. Expérimentation Live, chunks de 64 Ko, Débits hétérogènes

que le nombre de pièces augmente (pour un même contenu), MP reprend le dessus grâce à la disponibilité plus importante de la pièce suivante à télécharger. En comparant les résultats entre le cas 1024 et 512 Ko - (voir les figures 4.13 et 4.11), nous pouvons constater que des pièces plus grandes est un facteur de meilleure performance pour des grands contenus. Ce résultat est similaire à ce qui a été observé dans la littérature[68].

4.8 Conclusion

Le protocole BitTorrent a été étudié depuis plus de 10 ans. De nombreux paramètres ont été testés comme le nombre de pairs vers lesquels un pair télécharge ou la taille du peer set. La formation de cluster a également été observée. A notre connaissance, notre étude est la première à étudier l'enrichissement des méta-informations pour rendre le protocole plus efficace. Nous avons montré ici qu'a travers une modification simple (ajout de la pièce la plus rare), on pouvait améliorer les performances de BitTorrent tant pour le cas Live que pour le cas du partage de fichier. Nous continuons à suivre cette piste dans le chapitre suivant où nous allons modifier la façon dont les pairs émettent les données en se basant sur une nouvelle méta-information : le débit de téléchargement perçu par chaque pair d'un voisinage.

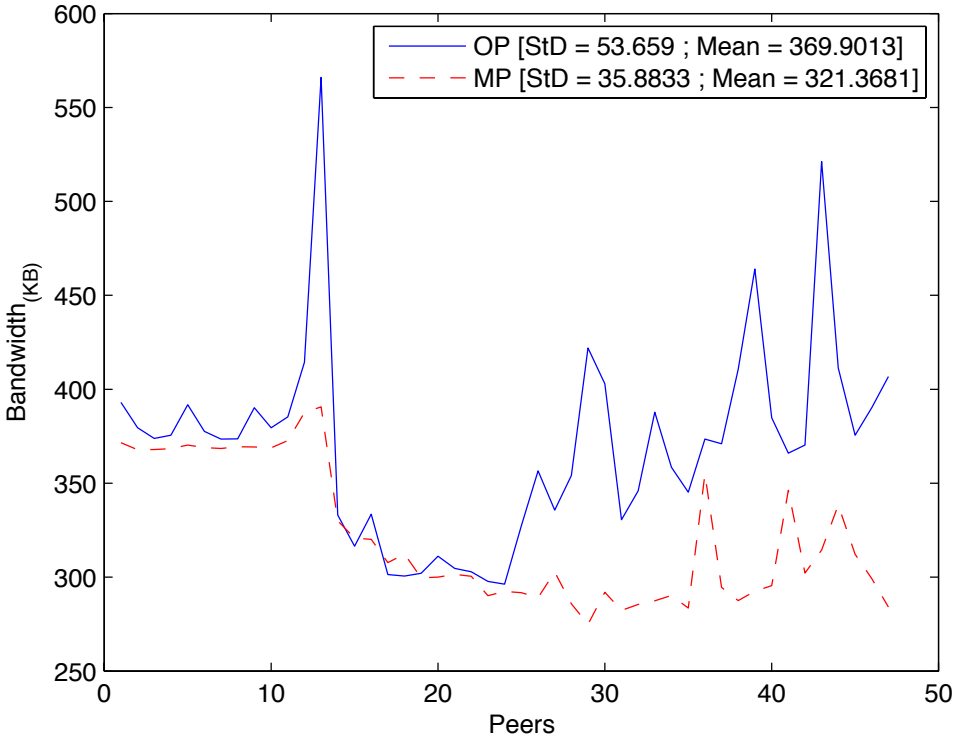

FIGURE 4.10 – Débit descendant. Expérimentation file sharing, chunks de 1024 Ko, Débits hétérogènes

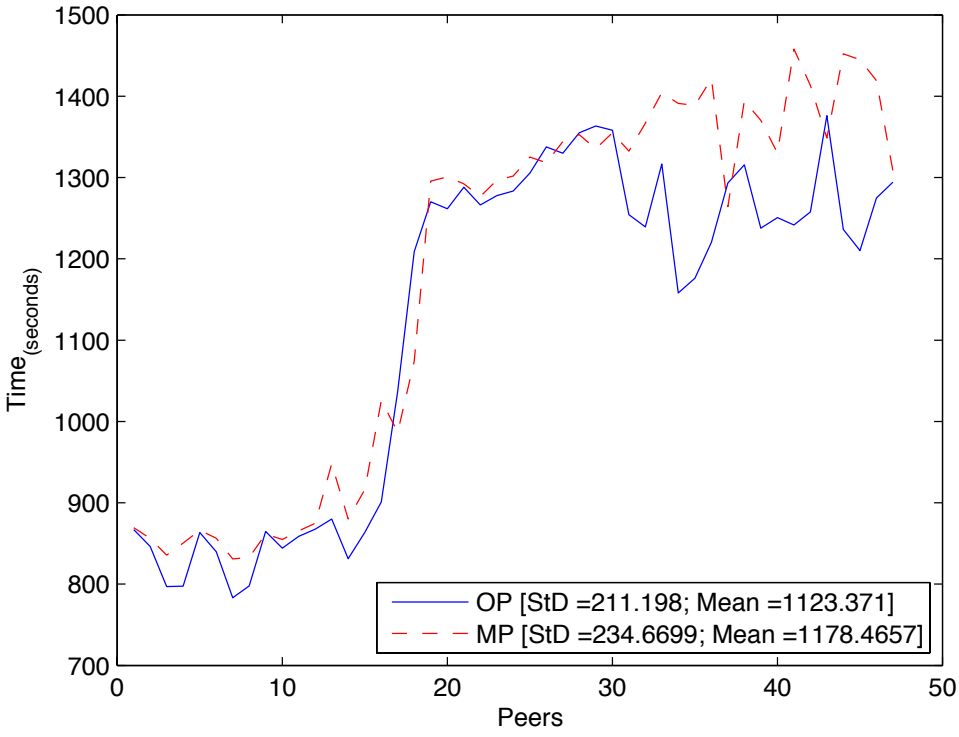

FIGURE 4.11 – Temps de téléchargement. Expérimentation file sharing, chunks de 1024 Ko, Débits hétérogènes

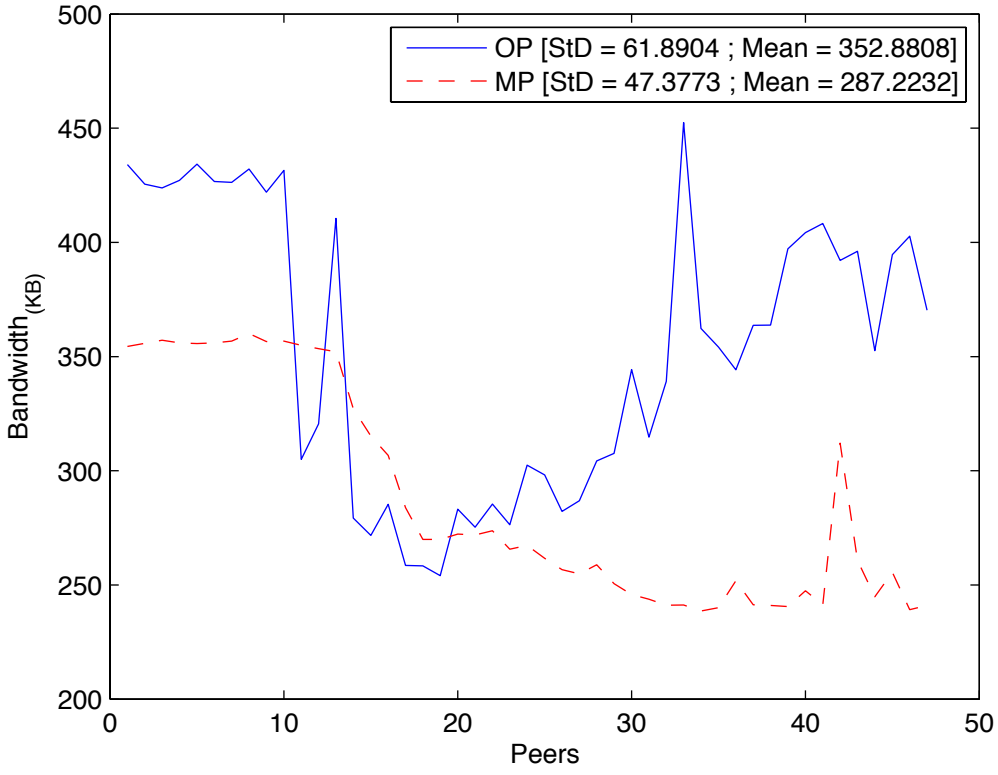

FIGURE 4.12 – Débit descendant. Expérimentation file sharing, chunks de 512 Ko, Débits hétérogènes

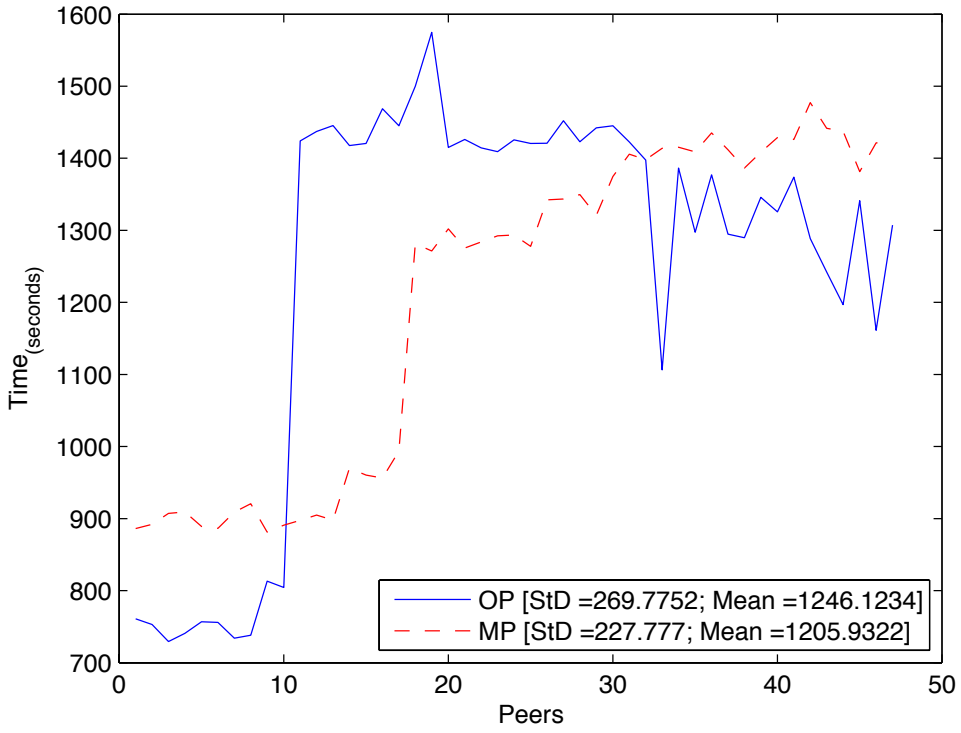

FIGURE 4.13 – Temps de téléchargement. Expérimentation file sharing, chunks de 512 Ko, Débits hétérogènes

Chapitre 5

Introduction du Push

Dans le chapitre précédent, nous avons proposé une modification de BitTorrent en améliorant la qualité et la diffusion des méta-données sur lesquelles le protocole base ses décisions. Nous nous sommes servis d'une requête de communication échangée régulièrement pour insérer ces calculs (méta-données). Les calculs effectués permettent d'accroitre la qualité des méta-données en donnant à chaque pair une vision plus globale du réseau.Cette nouvelle vision globale introduite précédemment nous a permis, dans le chapitre 4 de mieux distribuer la bonne pièce rare du réseau pour garantir une bonne entropie. Nous allons nous servir de cette nouvelle vision et l'associer à un concept introduit initialement dans les protocoles qui adoptent l'architecture en arbre : la notion de « Push ». Plus précisément, nous allons combiner à la vision globale du réseau exposée dans le chapitre précédent (chapitre 4) la notion de « Push ». Il s'agit non seulement d'une modification des méta-données, mais d'une modification de l'algorithme même de BitTorrent. Plus précisément, il s'agit de l'ajout d'un nouvel algorithme à côté des algorithmes *rarest first* et *choke*.

5.1 Introduction

L'engouement qu'a connu initialement le « Push » est vite retombé pour cause d'inefficacité. Cette méthode a été introduite par les architectures P2P *tree-based* et consistait à envoyer des requêtes d'un noeud-père à un noeud-fils qui enverra à un paquet reçu sans demande préalable et sans vérifier si le pair contacté l'a en sa possession[67] [79].

Une politique de ce type est mieux adaptée à des réseaux stables. Le voisinage dans ces derniers n'est généralement pas dynamique et les départs et arrivées des pairs sont constants. Dans le cas de la distribution de contenu, et spécialement pour le Live où les contraintes temps réel sont très accrues, ce broadcast forcé et *naïf* génère un taux de trafic important et conduit à l'inondation du réseau avec de l'information jugée inutile. Pour cette raison, la plupart des protocoles de distribution adoptent plutôt la méthode *Pull* qui permet aux pairs d'envoyer une requête pour demander et recevoir l'information exacte qu'ils souhaitent recevoir. Cette double contrainte information pertinente /temps réel est la motivation qui nous a conduit à

associer le push à la nouvelle vision globale du réseau. Contrairement au file sharing, recevoir le bloc de données pertinent avant l'instant de sa lecture en streaming est une nécessité pressante qui évitera des images figées, non complètes ou pixelisées. En outre, les clients doivent expérimenter des délais minimaux quelles que soient leurs ressources pour garantir une meilleure QoE.
Coolstreaming est un protocole qui combine le modèle push et pull dans sa dernière version [84]. A la différence d'un push basé sur la rareté des pièces, sa politique est de permettre au nouvel arrivant de sélectionner dans sa PL le noeud-père à qui il enverra une requête pull pour sélectionner le bloc de données qu'il souhaite. En réponse, le noeud-père va pusher contiuellement les chunks disponibles du bloc sélectionné.

Notez que la notion du push peut désigner la requête envoyée dans certains protocoles (Gnutella) pour la traversée de NAT. Cette requête est envoyée par un client A pour atteindre un client B se trouvant dans sa PL (*peer list*) mais derrière un pare-feu. Le client A demande au client B par un push : son *PeerID*, Index du fichier demandé, *adrIP* et le numéro de port où envoyer le fichier. Là encore, dans un protocole tel que Gnutella, le client A enverra continuellement des requêtes Push (jusqu'a atteinte du TTL qui est généralement égal à 7) au client B si la connexion n'est pas établie, ce qui causera l'inondation du réseau.

5.2 Algorithme du Push

Pour déployer le push, tout en évitant les redondances de transmission de données, nous devons cibler ce que nous allons pousser et à qui l'envoyer.

Pour le premier point, nous allons profiter des résultats de notre politique de distribution basée sur le nouveau calcul de la pièce rare et obtenu par la vision globale que possède l'ensemble des pairs du réseau. Deuxièmement, nous devons déterminer à qui « pusher » cette pièce serait profitable. Nous allons exclure les pairs avec une bonne bande passante en raison de leur plus grande chance d'être connecté directement aux seeds et plus généralement de leur meilleure capacité de téléchargement. Les faibles en bande passante seront écartés également parcequ'en général leur bande passante ralentit la redistribution rapide des pièces. Nous allons augmenter leur chance d'être contacté en « pushant » la pièce rare du réseau aux pairs avec une bande passante *moyenne*.
Concrètement, une nouvelle méta-information est transmise dans le réseau : à chaque récéption de pièce le pair transmet à ses voisins dans les messages Have, en plus de la pièce qu'il perçoit comme la plus rare, son débit montant et descendant. Ensuite, chaque pair opère un double tri sur les bandes passantes de son voisinage et deux conditions seront imposées :

1. Les pairs avec une bande passante montante (*Upload*) élevée seront désignés comme *Super peer*. Il faut préciser que même si un pair peut avoir une bonne bande passante en upload mais qu'il est saturé, lui donner un rôle de

super peer n'a pas de sens ; et donc les super peers seront ceux avec une bande passante en upload **libre** et élevée.

2. Le deuxième tri sera appliqué pour désigner les pairs qui recevront le push. Cette fois-ci nous viserons les pairs avec un débit descendant (*download*) moyen. Pour ce faire, les *Super peer* choisissent les pairs au milieu de leur liste classée. Il s'agit donc des pairs ayant un débit médian.

Avec cet algorithme, nous garantirons la réception de la pièce rare par tout le swarm, sans redondance et en profitant des bandes passantes opportunes présentes sur le réseau mais non exploitées.

5.3 Plateforme de test

Nous avons gardé la même plateforme de test, décrite dans le chapitre précédent section 4.7.1. Nous avons également gardé le même contenu en Live : un fichier de 39 Mo, qui correspond à la taille d'une fenêtre en streaming HD et qui sera découpé en pièce de 128 Ko et 256 Ko. Pareil pour la partie file sharing : nous effectuons nos tests avec le fichier AVI de 316 Mo découpé en 1024 Mo (voir la section 4.7.1). Nous avons doublé le nombre de pairs sur le réseau, puisque la migration sur le nouveau serveur nous permet de faire tourner 100 clients sans être contraints par les performances du matériel. Une extension du swap a été également prévue en cas de nécessité ainsi qu'une verification des ressources établie sur les durées des tests à l'aide d'un *rddtool* [1]. Des exemples d'analyse avec rddtool sont présentées en figures 5.1, 5.2, 5.3. On voit que la majorité de la mémoire est utilisée pour du cache et que le système atteint des pics en I/O wait, ce qui correspond aux lectures et écritures dans les fichiers par les clients BitTorrent. Le swap n'est néanmoins pas utilisé, ce qui signifie qu'il n' a pas de saturation de la mémoire

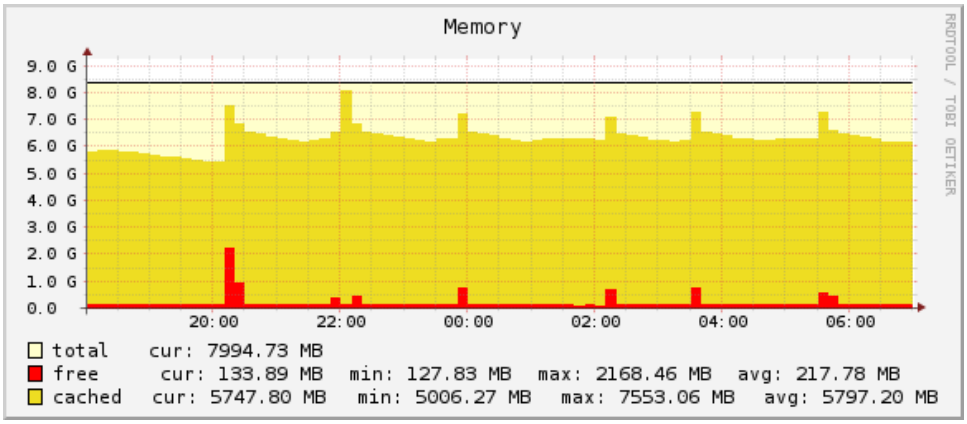

FIGURE 5.1 – memory

Les 100 pairs sont divisés, comme dans le chapitre précédent, en deux catégories : les *seeds* et les *leechs*. Nous avons fixé arbitrairement le nombre de seeds et

1. http ://oss.oetiker.ch/rrdtool/

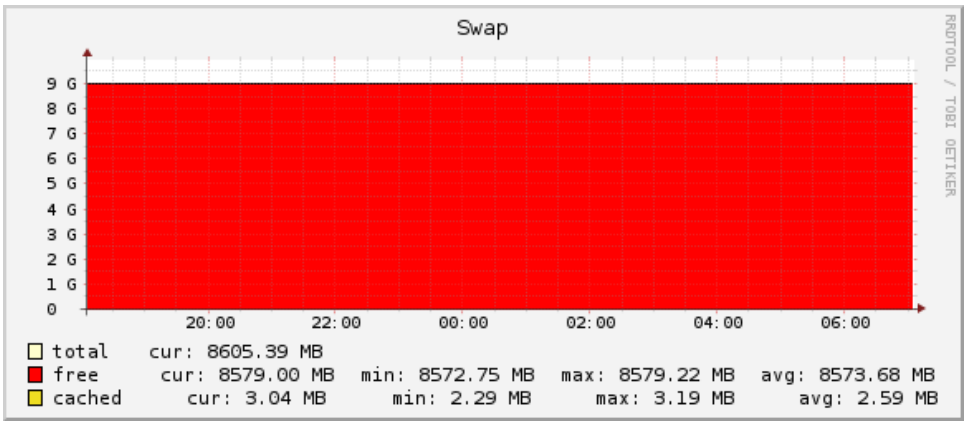

FIGURE 5.2 – swap

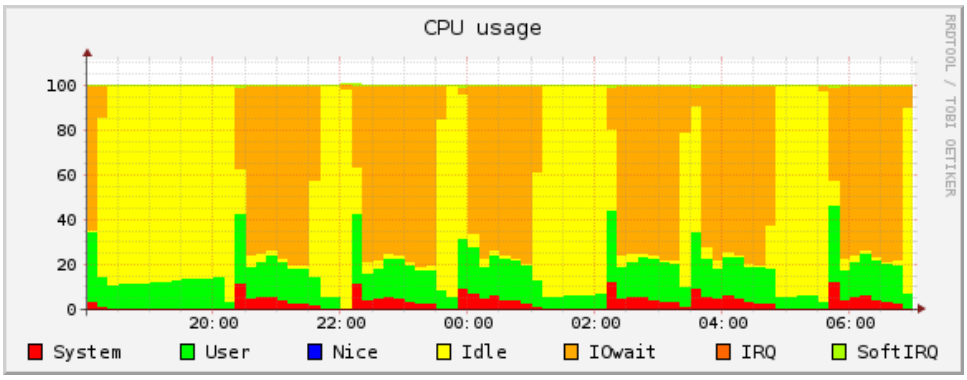

FIGURE 5.3 – cpu

leechs à 3 et 97 respectivement. Les expérimentations se font au sein d'un réseau fermé pour éviter tout bruit dû à un trafic externe. Nous avons également ajusté les bandes passantes des *seeds* ainsi que des *leechs* pour optimiser les durées des tests avec 100 pairs. Pour les tests du Live les débits sont comme suit :

— 25 leechs à 20Ko/s
— 24 à 40Ko/s
— 24 à 60Ko/s
— 24 leechs à 100Ko/s
— des seeds à 400Ko/s.

et pour le cas du file sharing :

— 25 leechs à 200 Ko/s
— 24 à 320Ko/s
— 24 à 600Ko/s
— 24 leechs à 800Ko/s
— des seeds à 1600Ko/s.

5.4 Résultats

Nous présentons dans cette section une comparaison de performance entre le protocole de distribution original (OP) le protocole modifié (RP) basé sur la pièce rare et son optimisation (MP) basé sur le push de la pièce rare.

À l'instar du chapitre précédent, nous allons comparer les performances en temps de téléchargement et en bande passante.

5.4.1 Résultats du Live

Comme expliqué précédemment (chapitre 4), le téléchargement du live / streaming se fait par réplication de la fenêtre courante au plus vite sans notion d'ordre pour la réception des chunks. Nous commençons nos expérimentations par un contenu live découpé en chunks de 64 Ko. Dans un premier temps, nous fixons le nombre de Super-Peers (SP) autorisés à pusher par swarm à 2 SP. Le temps opérationnel d'un SP sera de 30 secondes pour (le lot de) la première expérimentation. Les résultats obtenus sont représentés sur les figures 5.4 et 5.5.

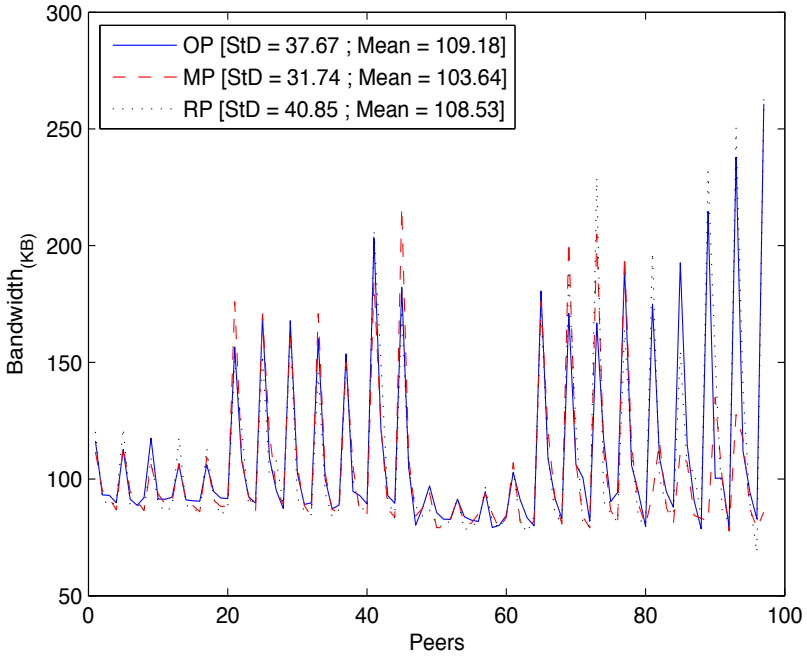

FIGURE 5.4 – Débit descendant. Expérimentation Live, chunks de 64 Ko, 2SP durant 30s

Nous pouvons constater que MP consomme moins en bande passante par rapport à RP et OP (103,64 Ko/s contre 108,53 Ko/s et 109,18 Ko/s). L'écart type faible affiché par MP signifie que la consommation était également réduite entre pairs du réseau. En ce qui concerne le temps de téléchargement, MP a pu réduire

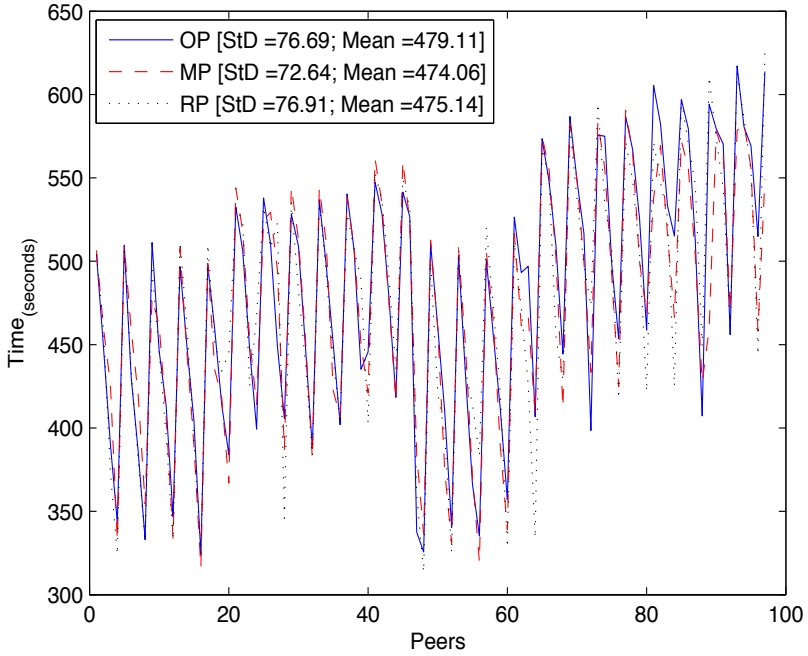

FIGURE 5.5 – Temps de téléchargement. Expérimentation Live,chunks de 64 Ko, 2SP durant 30s

de 5 secondes le temps moyen de la session de téléchargement par rapport à OP. La disponibilité des pièces créée par la nouvelle vision qu'ont les pairs, complétée par l'action de la distribuer à ceux qui ne l'ont pas en leur possession, permettra aux pairs de passer moins de temps dans la recherche des pièces suivantes. Ceci va réduire la différence en délai de visualisation entre les premiers arrivants et les derniers.

Nous pouvons également remarqué sur la figure 5.4 qu'à partir du pair 80, OP et RP consomment énormément en bande passante. Ceci va leur permettre de rattraper le retard en temps de téléchargement qu'ont les noeuds connectés plus tard, mais représente un délai important entre leurs moyennes et celle des pairs qui se sont connectés au démarrage de la session.

Durée en SP

Nous avons effectué la même expérimentation mais avec une session de 15 secondes pour les SP (fig 5.6 et 5.7). Nous écourtons le temps d'élection des SP pour céder ce rôle aux pairs susceptibles d'avoir une bande passante montante libre plus élevée. Si c'est le cas, le SP actuel cède sa place en gardant ses nouvelles connexions et continuant à distribuer les pièces qu'il a entamé. Le nouveau SP qui le remplacera va se charger de distribuer les pièces rares du réseau qu'il a en sa

possession. Dans le cas contraire et si la bande passante du SP n'est pas saturée, il gardera son rôle jusqu'au prochain round.

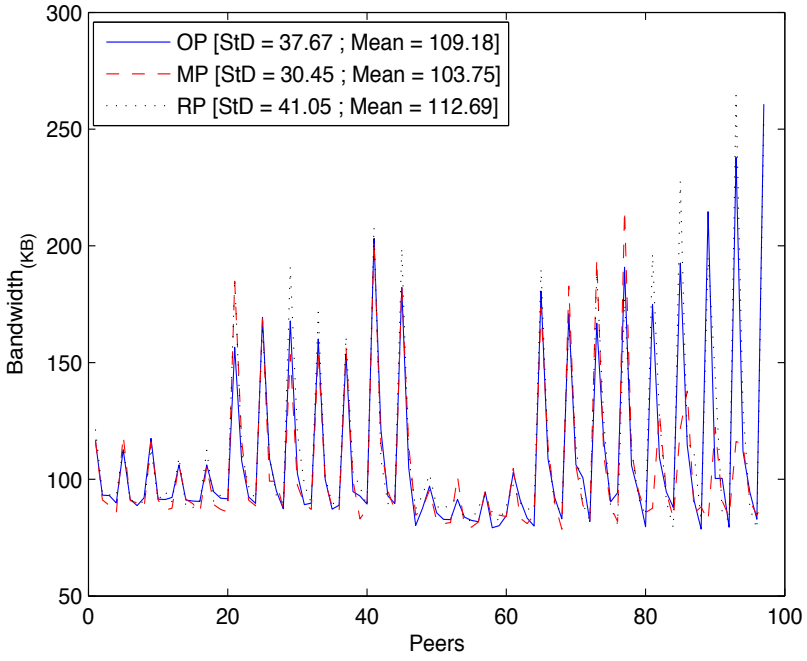

FIGURE 5.6 – Débit descendant. Expérimentation Live, chunks de 64 Ko, 2SP durant 15s

La figure 5.7 montre que MP a marqué une amélioration mais qui reste légère. la tendance en bande passante et en temps de téléchargement reste la même, nous avons donc baissé le temps d'éléction du SP à 10 secondes pour s'assurer que ce paramètre n'affecte pas les performances du push. Nous remarquons sur la figure 5.9 une légère dégradation au niveau du temps de téléchargement donc nous fixons dans la suite des tests 15s comme paramètre pour la durée opérationnelle d'un SP.

Taille des pièces

Nous découpons ensuite le contenu en pièces de 128 Ko, figures 5.10 et 5.11. Nous avons moins de pièces qu'avec le découpage précédent. Si nous nous basons sur le résultats de ce scénario par rapport au chapitre (4) MP et RP devaient marquer une nette amélioration. Or le changement de la taille du réseau et le nombre de pairs à servir a fait que nous ayons un peu près les mêmes tendances. Malgré cela, MP reste toujours plus performant en terme de moyenne de temps de téléchargement par rapport au protocole original (figure 5.11). Nous avons essayé de changer le nombre de SP dans le swarm en le réduisant à 1SP pour savoir si ce paramètre affecte les résultats (réduire le nombre de requête sur le réseau) mais le temps de téléchargement reste pratiquement inchangé. (figure 5.13)

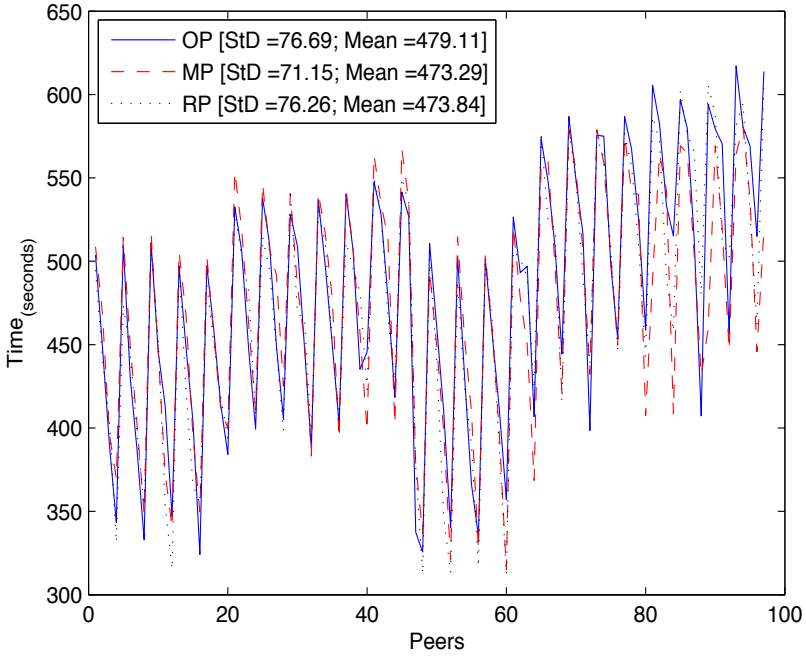

FIGURE 5.7 – Temps de téléchargement. Expérimentation Live,chunks de 64 Ko, 2SP durant 15s

5.4.2 Résultats du File sharing

Nous discutons dans cette section les effets du push sur les performances du téléchargement dans le cas du file sharing. Pour commencer, nous avons fixé le temps d'exploitation des Super-peers à 30 secondes. Nous avons choisi de fixer le nombre des SP à 5 SP. C'est le nombre des connexions actives généralement dans un swarm. Cette configuration est équivalente à permettre à tous les pairs du swarm de pusher s'ils ont suffisamment de bande passante libre pour le faire. Ce choix était motivé par l'absence des contraintes temps réel à respecter. Nous avons découpé le contenu file sharing (317 Mo) en pièces d'1Mo. Les résultats sont représentés sur la figure 5.14 et 5.15.

Les figures 5.14 et 5.15 montrent une optimisation tant pour la consommation en bande passante que pour la moyenne de temps de téléchargement (275s de moins qu'OP et 49s de moins que RP). Ceci s'explique par la disponibilité des pièces à télécharger et le fait que les pairs passent moins de temps à les chercher, réduisant leur besoin en bande passante. Le phénomène du cluster reste plus prononcé dans le cas du file sharing. Ceci dit, cela n'a pas de fort impact vu qu'il ne s'agit pas d'un téléchargement avec des contraintes temps réel. En file sharing, la rapidité du temps globale de la session de téléchargement est plus importante.

Nous avons essayer d'optimiser ces résultats en réduisant le nombre de Super-peers sur le swarm (moins de SP=moins de requêtes). Nous avons fixé leurs nombre

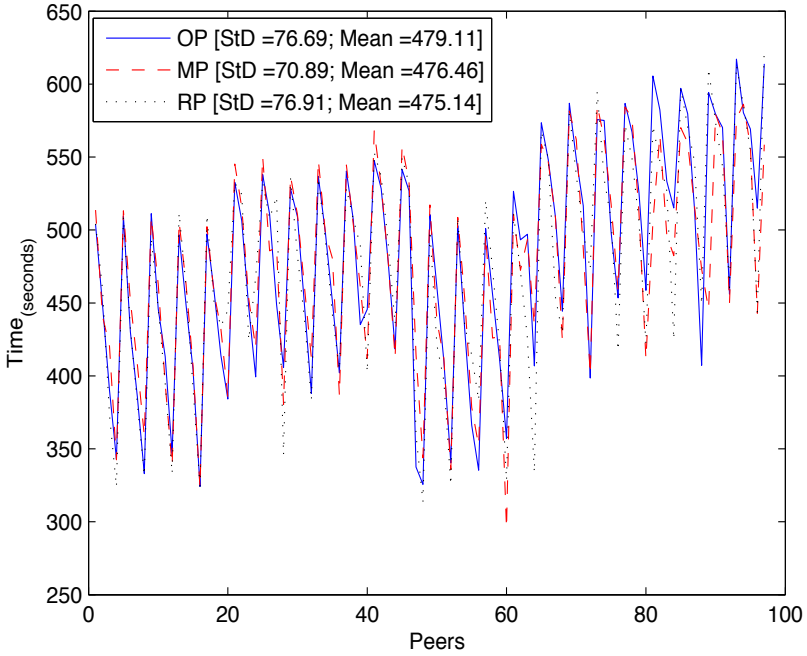

FIGURE 5.8 – Débit descendant. Expérimentation Live, chunks de 64 Ko, 2SP durant 10s

à 3 SP par swarm (figure 5.16 et 5.17). Nous pouvons remarquer que la bande passante n'a pas été affectée par ce changement (figure 5.16), tandis qu' une amélioration de 23 secondes en temps de téléchargement a été observée (figure 5.16).

5.5 Conclusion

Nous avons exploité la richesse des méta-données partagées entre les pairs du réseau pour améliorer leur expérience de téléchargement. Dans un premier temps, nous avons expérimenté l'efficacité de la dissémination de la pièce rare calculée avec la nouvelle méthode et nous avons profité de l'enrichissement des méta-données avec l'information des débits montants et descendants des pairs. Ceci nous a permis d'attribuer plus de responsabilité aux pairs bénéficiant d'un confort en ressources, c'est-à-dir les *super-peers*. Cela s'est traduit par des sessions de téléchargement plus courtes, tant pour le live que le partage de fichier et sans avoir à consommer plus en bande passante en raison de la disponibilité des pièces.

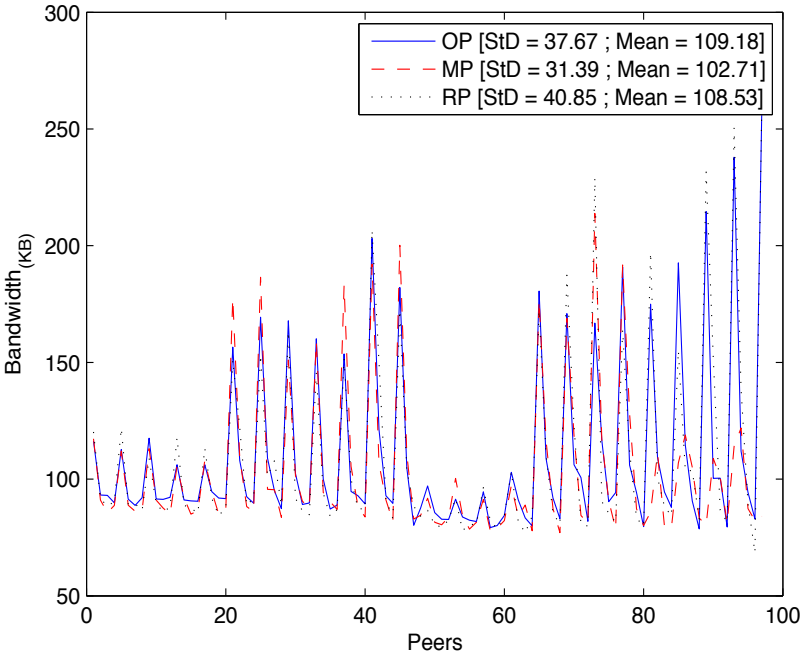

FIGURE 5.9 – Temps de téléchargement. Expérimentation Live,chunks de 64 Ko, 2SP durant 10s

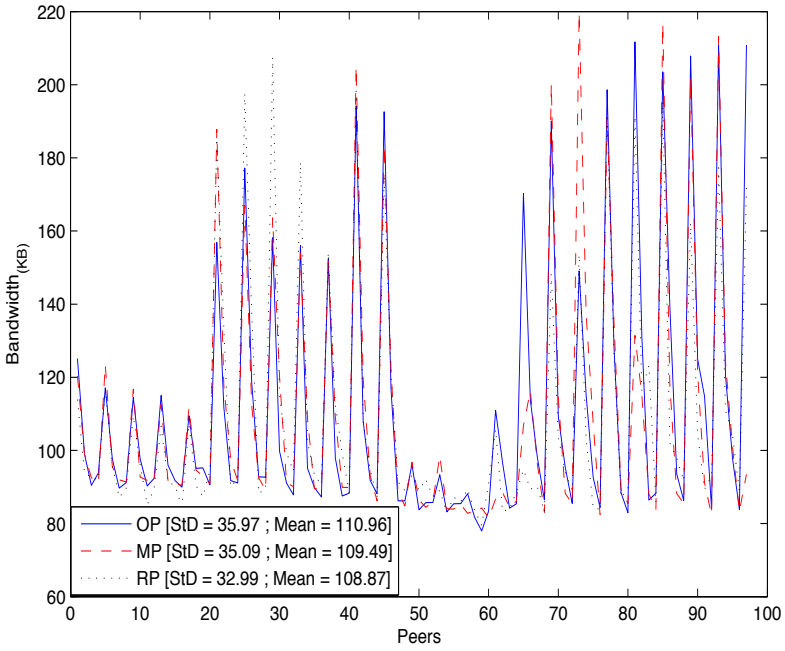

FIGURE 5.10 – Débit descendant. Expérimentation Live, chunks de 128 Ko, 2SP durant 15s

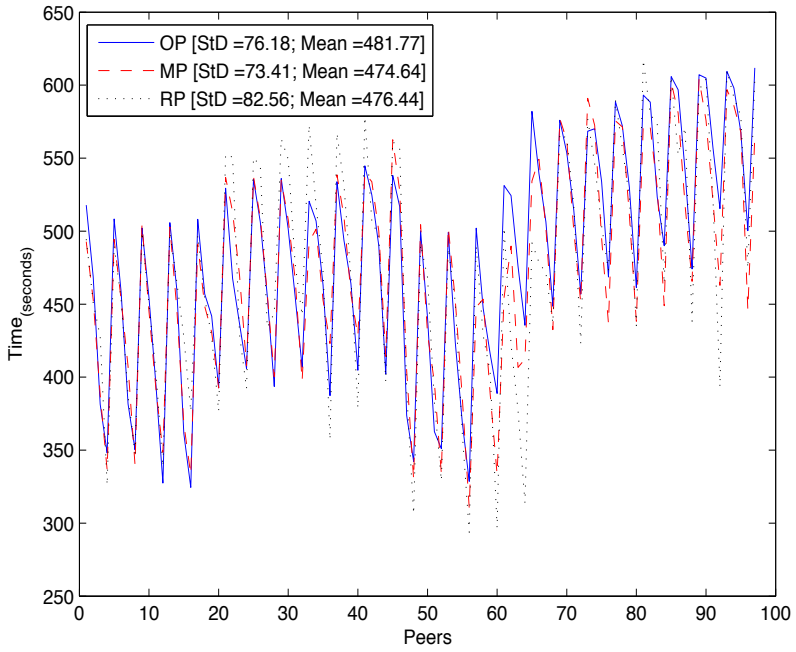

FIGURE 5.11 – Temps de téléchargement. Expérimentation Live,chunks de 128 Ko, 2SP durant 15s

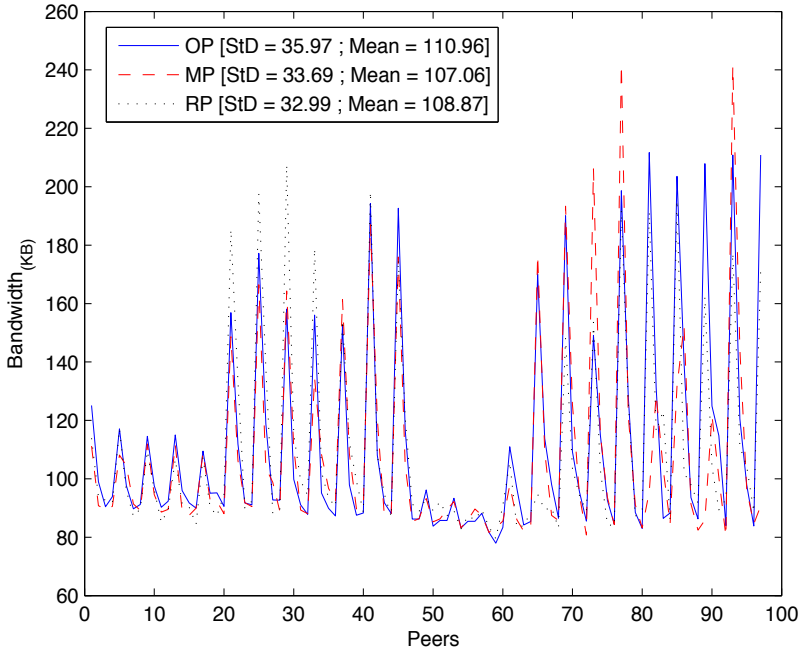

FIGURE 5.12 – Débit descendant. Expérimentation Live, chunks de 128 Ko, 1SP durant 15s

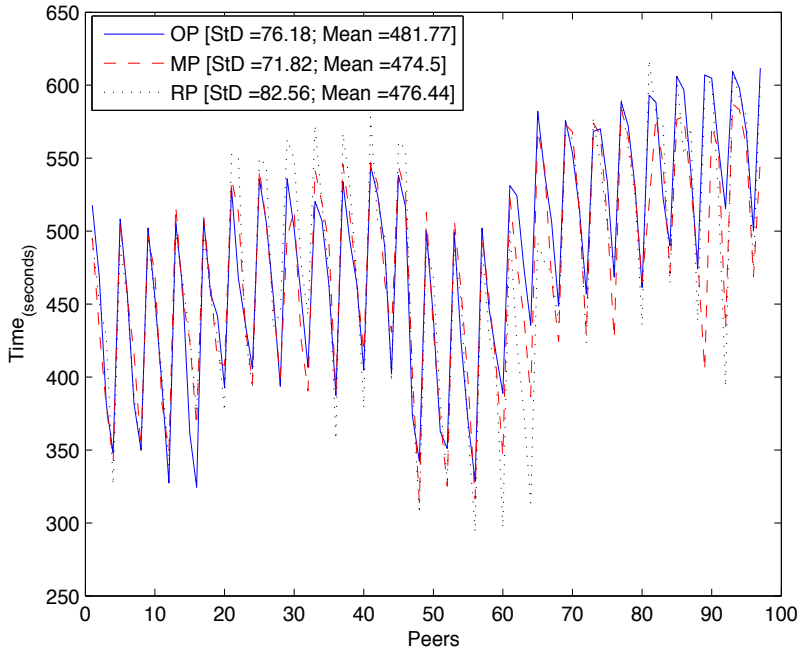

FIGURE 5.13 – Temps de téléchargement. Expérimentation Live,chunks de 128 Ko, 1SP durant 15s

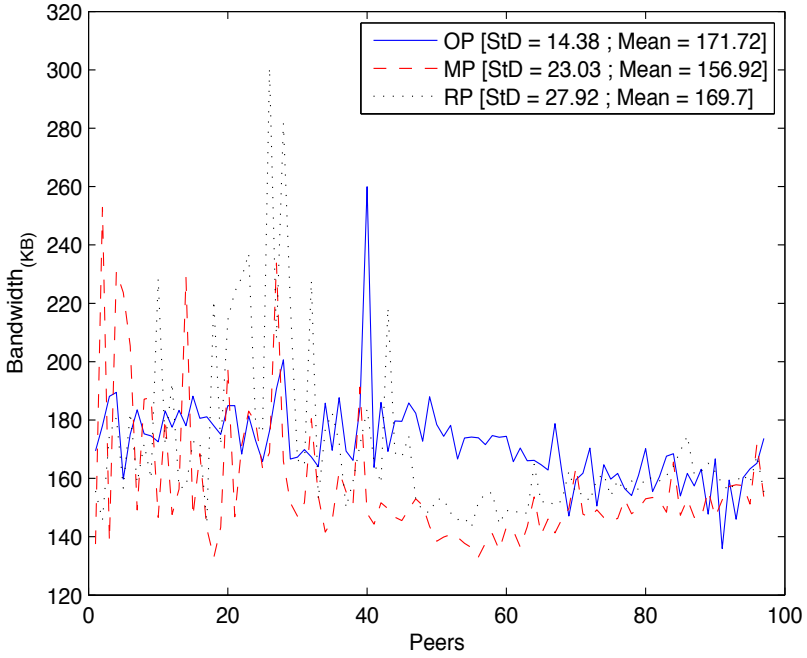

FIGURE 5.14 – Débit descendant. Expérimentation File sharing, chunks d'1Mo, 5SP durant 30s

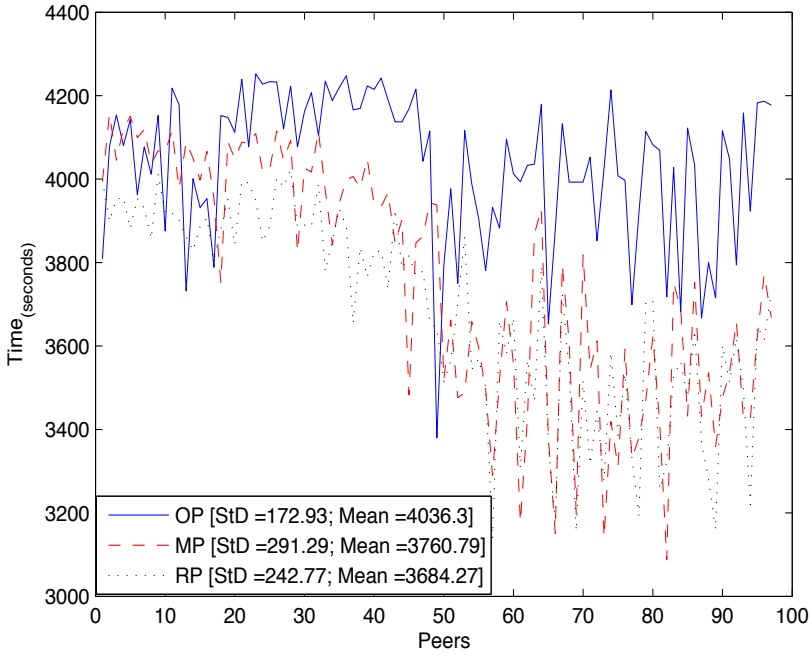

FIGURE 5.15 – Temps de téléchargement. Expérimentation File sharing, chunks d'1Mo, 5SP durant 30s

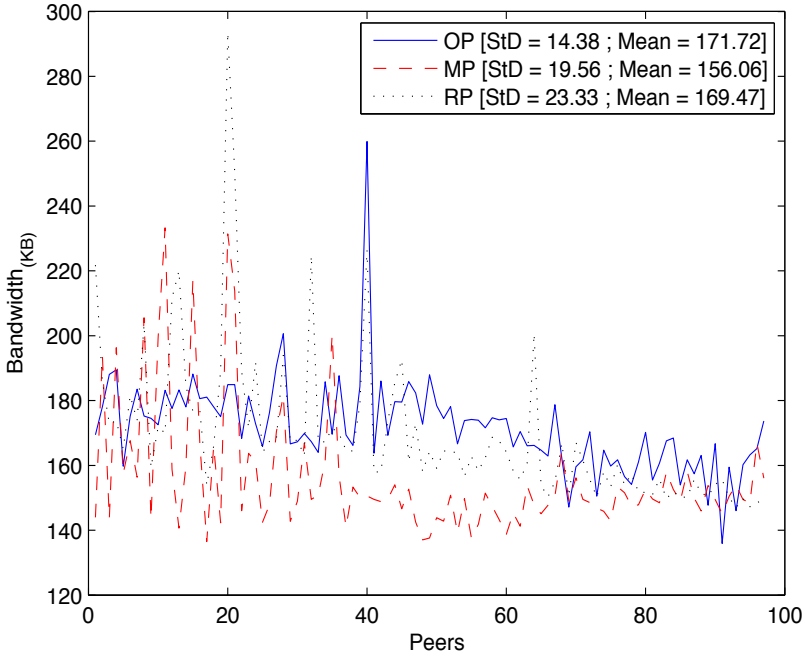

FIGURE 5.16 – Débit descendant. Expérimentation File sharing, chunks d'1Mo, 3SP durant 20s

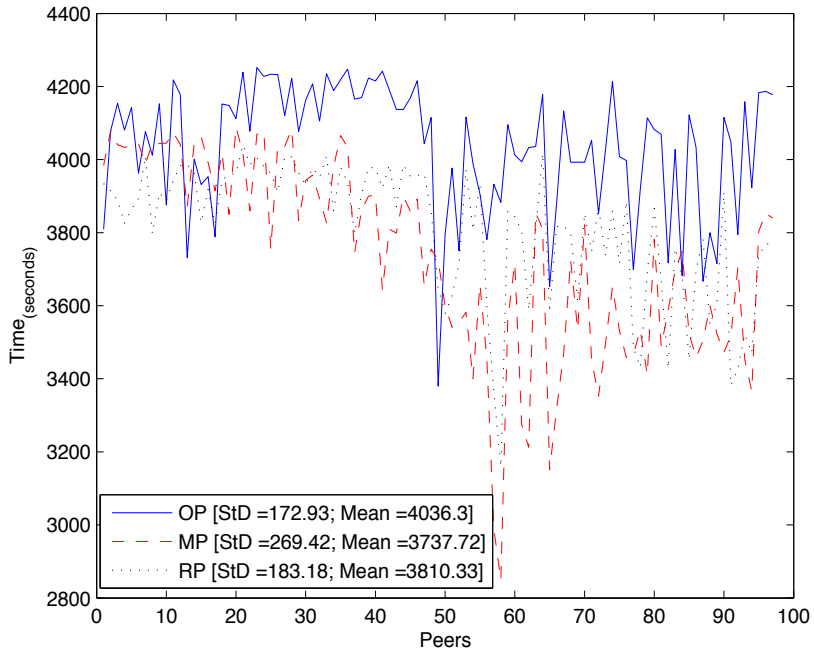

FIGURE 5.17 – Temps de téléchargement. Expérimentation File sharing, chunks d'1Mo, 3SP durant 20s

Chapitre 6

Conclusion

Nous avons abordé cette thèse en ayant pour objectif de proposer de nouvelles méthodes de distribution de contenus sur Internet au travers de l'amélioration des protocoles existants. Pour ce faire, nous nous sommes basés dans un premier temps sur des données réelles collectées par TMG sur une durée de trois ans et demi. Dans un premier temps, nous avons analysé les choix des utilisateurs que nous estimons liés à la qualité de service et d'expérience que proposent les protocoles de distribution. Nous avons également examiné l'impact de la loi Hadpoi sur le comportement des utilisateurs : le trafic a diminué significativement après la première campagne massive de courriers électroniques envoyés aux utilisateurs engagés dans le téléchargement illégal. Par ailleurs, la fermeture de Megaupload n'a donné lieu à aucune hausse d'activité des principaux réseaux P2P étudiés. Nous appuyons l'hypothèse, comme cela a été suggéré dans la littérature, que de nouveaux moyens plus furtifs sont utilisés à présent pour distribuer des copies illégales de contenu protégé par des droits d'auteur. En outre, nous avons remarqué que parmi les protocoles qui ont maintenu leur activité, BitTorrent a su gagner du terrain grâce à sa gestion du temps de téléchargement et la discrétion qu'il offre à ses utilisateurs en les exposant moins à la loi Hadopi. Nous avons aussi révélé que les utilisateurs sont attirés par des contenus de taille croissante : en contenu audio, les compilations et les albums sont plus sollicités et en vidéo, les vidéos encodées avec des qualités supérieures deviennent de plus en plue populaires.

Dans la deuxième partie de cette thèse, nous nous sommes intéressés aux protocoles de diffusion avec des contraintes de temps réel. Ces différents protocoles reposent sur un fonctionnement BitTorrent mais avec une disposition architecturale différente. Sopcast a retenu notre attention. Dans la littérature, de nombreuses études ont souligné qu'il s'agit de l'une des applications P2PTV les plus dominantes. Les statistiques fournit par la plateforme TMG confirmait cette observation. En 2011, ce protocole a modifié son algorithme de sélection de peer-list pour construire le voisinage d'un nouveau pair dans le swarm. Nous avons étudié ce changement puisqu'il s'agit d'une étape cruciale dans le fonctionnement des applications streaming. Cet algorithme opère lors du démarrage de la diffusion, c'est-à-dire le temps nécessaire pour afficher le flux vidéo d'une chaine. Nous avons d'abord démontré que ce nouvel algorithme repose sur la distance entre les adresses IP des nouveaux pairs arrivants et ceux des pairs déjà présents dans le réseau. Nous

avons exposé le degré de corrélation entre la distance IP et la distance réseau réelle, mesurée à travers différents *pings* (le ping système et un ping que nous avons développé imitant la procédure Handshake de Sopcast). Cette corrélation dépend à la fois de la taille du réseau et la relation géographique entre le nouveau pair qui rejoint une chaîne et le contenu diffusé. Si le nouveau pair est loin de la source du contenu et des pairs connecté à ce flux, il peut être victime d'une phase de démarrage longue : il lui faut plus de temps pour trouver des voisins adéquats, à l'aide d'autres algorithmes tel PEX (échange de liste de pairs entre voisins).

Après ces différentes analyses des protocoles de distribution, nous avons orienté nos travaux vers la propositions d'un protocole de distribution similaire à BitTorrent compte tenu de sa popularité et son éfficacité tant pour les grands transferts que pour le streaming. Cependant, ce dernier (BitTorrent) offre une vision limitée du réseau : La façon dont les informations (méta-données) sur l'état des pairs sont disséminées sur le réseau est un facteur important qui peut influencer les performances d'un réseau P2P. Or cette vision est locale, c'est-à-dire qu'un pair du réseau n'échange de l'information qu'avec les pairs auxquels il est directement connecté. Si cette information n'est pas suffisamment de bonne qualité, cela ne permettra pas aux pairs de s'investir au vrai niveau de leurs capacités, pour maximiser les performances globales du réseau. Il y a donc un lien direct entre la qualité des méta-données diffusées entre les pairs et les performances globales du réseau.

Cette réflexion nous a mené à proposer des méthodes pour enrichir les méta-données, tout en gardant une diffusion locale entre voisins, améliorant ainsi les performances globales d'un swarm BitTorrent. Nous avons proposé un protocole de partage P2P basé sur une gestion collaborative qui implique tous les nœuds du réseau dans la prise de décision et la survie du réseau. Lors des échanges entre pairs, un calcul des aspects dynamiques du nœud et de son environnement sera fait simultanément. A l'inverse d'une estimation sur l'ensemble du réseau qui est coûteuse, chaque nœud fait une estimation moyenne de ses ressources avec celles de ses voisins à l'aide d'un échange de méta-données enrichi par rapport au protocole initial.

Nous avons exploité une première information qui est la pièce rare du réseau, calculée avec le nouvel algorithme proposé. Nous avons démontré qu'impliquer les pairs dans la définition du besoin en termes de pièces manquantes et les inciter à distribuer la plus rare en priorité a permis de garantir une bonne entropie. Le nouveau protocole a enregistré des sessions de téléchargement plus réduites à la fois pour le Live et le file sharing. Les résultats obtenus nous ont conduit à proposer une première amélioration au travers d'une stratégie de push. Cette méthode assure l'efficacité de la dissémination de la pièce rare combinée à une nouvelle méta-donnée à savoir, les débits montants et descendants des pairs. Cette méthode nous a permis d'affecter des rôles plus dynamiques aux pairs bénéficiant d'un confort en ressources. Les expérimentations effectuées en milieu contrôlé ont montré des sessions de téléchargement plus courtes, tant pour le live que le partage de fichier.

Une réduction également de l'écart-type entre pairs en terme de bande passante a été remarquée ainsi qu'au sein des clusters pour les derniers derniers arrivant sur le réseau. Cela se traduit par une modération des délais de visualisation entre les noeuds du réseau.

Une étape clé dans la continuation de ses travaux sera, de toute évidence, l'expérimentation du protocole modifié dans un environnement réel, non controlé. Nous pourrions commencer par des expérimentations sur Planetlab (https://www.planet-lab.org/). Par ailleurs, nous continuerons d'étendre l'exploitation des méta-données sur de nouveaux aspects comme la sécurité. Nous pouvons imaginer un indice des fausses pièces distribuées permettant de détecter les pairs malveillants : une prévention contre les attaques sera mise en place ainsi qu'un mécanisme d'anti-piratage des contenus qui sera géré localement par chaque noeud du réseau.

Nous prévoyons aussi l'intégration du nouveau protocole à un environnement Web au niveau des navigateurs et à travers l'intégration des normes WebRTC (https://webrtc.org/). L'apport du nouveau protocole sur des échanges video-conférence nous semble aussi une piste intéressante à explorer. Nous allons viser dans cette partie le cas du multi-noeuds avec des resources et puissances en débit très différentes.

D'autre perspectives de recherche avec un volet économique sont apparues à l'issue de l'observation de la consommation grandissantes des contenus vidéos de haute qualité. En effet, nous avons observé une richesse et une diversité de contenu grandissante sur le web d'une part, et l'arrivée des nouvelles définitions dont la 4K et UHD d'une autre. Cette richesse masque néanmoins les problèmes des distributeurs de contenus tels YouTube ou Netflix (le premier en tant que réseau communautaire, le second en tant que service de vidéo à la demande) dont les coûts en bande passante sont prohibitifs. La réduction de la consommation en bande passante en s'appuyant sur le P2P est un axe très prometteur.

Bibliographie

[1] URL : http://www.ncsa.illinois.edu/enabling/mosaic.

[2] URL : http://www.ncsa.illinois.edu/.

[3] URL : http://isp.netscape.com/.

[4] URL : http://sillydog.org/narchive/.

[5] URL : http://www.w3c.org.

[6] URL : http://www.sites.univ-rennes2.fr/urfist/internet_chiffres.

[7] URL : http://www.napster.com/.

[8] URL : http://imesh.com.

[9] URL : http://freenetproject.org.

[10] URL : http://www.emule.com/.

[11] URL : http://www.shareaza.fr/.

[12] URL : http://www.skype.com/.

[13] URL : http://www.statisticbrain.com/skype-statistics/.

[14] URL : https://tox.chat/index.html.

[15] URL : https://www.google-melange.com/gsoc/homepage/google/gsoc2014.

[16] URL : https://www.google-melange.com/gsoc/homepage/google/gsoc2014/tox.

[17] URL : http://www.peersafe.cn/EN/index.html.

[18] URL : http://www.sha-talk.com/EN/about.html.

[19] URL : http://developer.servalproject.org/dokuwiki/doku.php?id=content:servalchat:main_page.

[20] URL : http://www.servalproject.org/.

[21] URL : http://opengarden.com/FireChat/.

[22] URL : http://opengarden.com/home/.

[23] URL : http://tuseapp.com/about.

[24] URL : http://tuseapp.com/.

[25] URL : http://maidsafe.net/.

[26] URL : http://maidsafe.net/safecoin.html.

[27] URL : https://www.teamdrive.com/.

[28] URL : http://tresorit.com/.

[29] URL : https://www.mediafire.com/.

[30] URL : http://www.egnyte.com/.

[31] URL : https://www.sugarsync.com/.

[32] URL : https://onedrive.live.com/.

[33] URL : https://www.google.com/intl/fr_fr/drive/.

[34] URL : http://www.apple.com/fr/icloud/.

[35] URL : https://www.microsoft.com/fr-xf/servicesagreement/.

[36] URL : https://getsync.com/.

[37] URL : http://www.legislation.govt.nz/act/public/2011/0011/latest/DLM2764312.html.

[38] URL : http://www.legislation.gov.uk/ukpga/2010/24/contents.

[39] URL : http://www.bbc.com/news/technology-16391727.

[40] URL : http://www.copyrightinformation.org/.

[41] URL : http://www.copyrightinformation.org/the-copyright-alert-system/what-is-a-copyright-alert/.

[42] Michael ARNOLD et al. « Graduated Response Policy and the Behavior of Digital Pirates : Evidence from the French Three-strike (Hadopi) Law ». In : (2014).

[43] Salman A BASET et Henning SCHULZRINNE. « An analysis of the skype peer-to-peer internet telephony protocol ». In : *arXiv preprint cs/0412017* (2004).

[44] Ignacio BERMUDEZ, Marco MELLIA et Michela MEO. « Investigating overlay topologies and dynamics of P2P-TV systems : The case of SopCast ». In : *Selected Areas in Communications, IEEE Journal on* 29.9 (2011), p. 1863–1871.

[45] D. CARRA, G. NEGLIA et P. MICHIARDI. « On the Impact of Greedy Strategies in BitTorrent Networks : The Case of BitTyrant ». In : *Peer-to-Peer Computing , 2008. P2P '08. Eighth International Conference on*. 2008, p. 311–320. DOI : 10.1109/P2P.2008.45.

[46] Bram COHEN. *Incentives Build Robustness in BitTorrent*. 2003.

[47] Bram COHEN. *The BitTorrent protocol specification*. 2008.

[48] Landon P COX, Christopher D MURRAY et Brian D NOBLE. « Pastiche : Making backup cheap and easy ». In : *ACM SIGOPS Operating Systems Review* 36.SI (2002), p. 285–298.

[49] Cameron DALE et Jiangchuan LIU. « A measurement study of piece population in BitTorrent ». In : *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*. IEEE. 2007, p. 405–410.

[50] Brett DANAHER et al. « The Effect of Graduated Response Anti-Piracy Laws on Music Sales : Evidence from an Event Study in France ». In : *Journal of Industrial Economic* (2013).

[51] Sylvain DEJEAN, Thierry PENARD et Raphael SUIRE. « The French "Three Strikes" Law against digital piracy and the change in usages of pirates ». In : *The Internet, Policy & Politics Conferences* (2010).

[52] Sylvain DEJEAN et Raphaël SUIRE. « Digital files dealers and prohibition in the context of the French 3 strikes (HADOPI) law ». In : *Available at SSRN 2422933* (2014).

[53] Roger DINGLEDINE, Nick MATHEWSON et Paul SYVERSON. *Tor : The second-generation onion router*. Rapp. tech. DTIC Document, 2004.

[54] Kolja EGER et al. « Efficient simulation of large-scale p2p networks : packet-level vs. flow-level simulations ». In : *Proceedings of the second workshop on Use of P2P, GRID and agents for the development of content networks on p 9–16*. 2007.

[55] R FORTUNA et al. « QoE in pull based P2P-TV systems : overlay topology design tradeoffs ». In : *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on p 1–10*. 2010.

[56] Krishna P GUMMADI et al. « Measurement, modeling, and analysis of a peer-to-peer file-sharing workload ». In : *ACM SIGOPS Operating Systems Review*. T. 37. 5. ACM. 2003, p. 314–329.

[57] Xiaojun HEI, Yong LIU et Keith W ROSS. « Inferring network-wide quality in P2P live streaming systems ». In : *Selected Areas in Communications, IEEE Journal on* 25.9 (2007), p. 1640–1654.

[58] A HORVATH et al. *Dissecting pplive, sopcast*. Rapp. tech. TVAnts. Technical report, NAPA-WINE project, 2009.

[59] Anthony J HOWE. « Napster and Gnutella : a comparison of two popular Peer-to-Peer protocols ». In : *Universidade de Victoria* 11 (2000).

[60] D. IRVINE. *Maidsafe.net*. EP Patent App. EP20,070,824,635. 2009. URL : http://www.google.com.na/patents/EP2118808A2?cl=fr.

[61] Orin S KERR. « Internet surveillance law after the USA Patriot Act : The big brother that isn't ». In : *Available at SSRN 317501* (2003).

[62] Yoram KULBAK, Danny BICKSON et al. « The eMule protocol specification ». In : *eMule project, http ://sourceforge. net* (2005).

[63] Martin LANDERS, Han ZHANG et Kian-Lee TAN. « Peerstore : Better performance by relaxing in peer-to-peer backup ». In : *Peer-to-Peer Computing, 2004. Proceedings. Proceedings. Fourth International Conference on*. IEEE. 2004, p. 72–79.

[64] Nikolaos LAOUTARIS, Damiano CARRA et Pietro MICHIARDI. « Uplink Allocation Beyond Choke/Unchoke : Or How to Divide and Conquer Best ». In : *Proceedings of the 2008 ACM CoNEXT Conference*. CoNEXT '08. Madrid, Spain : ACM, 2008, 18 :1–18 :12. ISBN : 978-1-60558-210-8. DOI : 10.1145/1544012.1544030. URL : http://doi.acm.org/10.1145/1544012.1544030.

[65] Arnaud LEGOUT, Guillaume URVOY-KELLER et Pietro MICHIARDI. « Rarest first and choke algorithms are enough ». In : *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement p 203–216*. 2006.

[66] Arnaud LEGOUT et al. « Clustering and Sharing Incentives in BitTorrent Systems ». In : *Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*. SIGMETRICS '07. San Diego, California, USA : ACM, 2007, p. 301–312. ISBN : 978-1-59593-639-4. DOI : 10.1145/1254882.1254919. URL : http://doi.acm.org/10.1145/1254882.1254919.

[67] Yong LIU, Yang GUO et Chao LIANG. « A survey on peer-to-peer video streaming systems ». In : *Peer-to-peer Networking and Applications* 1.1 (2008), p. 18–28.

[68] Pawel MARCINIAK et al. « Small is not always beautiful ». In : *Proceedings of the 7th international conference on Peer-to-peer systems, IPTPS'08, Tampa, FL, USA, February 25-26, 2008*. 2008, p. 9. URL : http://www.iptps.org/papers-2008/55.pdf.

[69] G MARFIA et al. « Will IPTV ride the peer-to-peer stream ». In : *IEEE Communications Magazine, Special Issue on Peer-to-Peer Streaming* (2007).

[70] James P MARTIN et Harry CENDROWSKI. « Electronic Communications Privacy Act ». In : *Cloud Computing and Electronic Discovery* (2014), p. 55–74.

[71] José MENDES, Paulo SALVADOR et António NOGUEIRA. « P2P-TV service and user characterization ». In : *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*. IEEE. 2010, p. 2612–2620.

[72] Hou MENG-SHU, Lu XIAN-LIANG et Wang TAO. « Enforcing sharing of peer-to-peer storage resources ». In : *Advanced Communication Technology, 2005, ICACT 2005. The 7th International Conference on*. T. 2. IEEE. 2005, p. 792–796.

[73] Satoshi NAKAMOTO. *Bitcoin : A peer-to-peer electronic cash system*. 2008.

[74] Ricardo Lopes PEREIRA et Teresa VAZAO. « On the cohabitation of Adaptive Search Radius enabled peers with regular eMule peers ». In : *Communications, 2007. ICC'07. IEEE International Conference on (pp. 6356-6361). IEEE*. 2007.

[75] Johan A POUWELSE et al. « TRIBLER : a social-based peer-to-peer system ». In : *Concurrency and Computation : Practice and Experience* 20.2 (2008), p. 127–138.

[76] Frahbakhsh REZA et al. « Investigating the reaction of bittorrent content publishers to anti piracy actions ». In : *13-th IEEE International Conference on Peer-to-Peer Computing* (2013).

[77] Ana Paula Couto da SILVA et al. « A bandwidth-aware scheduling strategy for P2P-TV systems ». In : *Peer-to-Peer Computing, 2008. P2P'08. Eighth International Conference on p 279–288*. 2008.

[78] Yu-Wei Eric SUNG, Michael A BISHOP et Sanjay G RAO. « Enabling contribution awareness in an overlay broadcasting system ». In : *Multimedia, IEEE Transactions on* 9.8 (2007), p. 1605–1620.

[79] Ms R THANGAMANI et Mr G SIVAKUMAR. « SURVEY ON VARIOUS PEER DISCOVERY TECHNIQUES IN P2P VIDEO ON DEMAND SYSTEMS (VOD) ». In : *Journal of Radix International Educational and Research Consortium* (December 2012).

[80] Quang Hieu VU, Mihai LUPU et Beng Chin OOI. « Architecture of peer-to-peer systems ». In : *Peer-to-Peer Computing*. Springer, 2010, p. 11–37.

[81] Hao WANG. « Skype VoIP service-architecture and comparison ». In : *INFOTECH Seminar Advanced Communication Services (ASC)*. Citeseer. 2005, p. 4.

[82] Di WU et al. « Understanding peer exchange in bittorrent systems ». In : *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*. IEEE. 2010, p. 1–8.

[83] Raymond Lei XIA et Jogesh K MUPPALA. « A survey of BitTorrent performance ». In : *Communications Surveys & Tutorials, IEEE* 12.2 (2010), p. 140–158.

[84] Susu XIE et al. « Coolstreaming : Design, theory, and practice ». In : *IEEE Transactions on Multimedia* 9.8 (2007), p. 1661–1671.

[85] Ye ZHU et al. « Correlation-based traffic analysis attacks on anonymity networks ». In : *Parallel and Distributed Systems, IEEE Transactions on* 21.7 (2010), p. 954–967.