



télécom
saint-étienne

école d'ingénieurs
nouvelles technologies

Projet TIPE
GYBELS Hugo et ADELAIN Guillaume

Le Peer-to-Peer



Tuteur :
Mme PROUST-SAUVIAC Isabelle

Date :
2014-2015

Table des matières

1 - Introduction	3
1.1. Un peu d'histoire	3
1.2. Actuellement	3
2 - Fonctionnement du modèle de réseau	4
2.1. Un nouveau modèle de réseau	4
2.1.1. Modèle client/serveur classique	4
2.1.2. Modèle de réseau peer-to-peer	5
2.2. Les différents réseaux de partage de fichiers	8
2.3. Le Hashage :	8
2.1.3. Le SHA-1 :	10
2.4. Le protocole BitTorrent	11
2.4.1. Partage de fichier via le protocole BitTorrent	11
2.4.2. Récupérer un fichier via le protocole BitTorrent	13
3 - Expérimentation : téléchargement d'un fichier via le protocole BitTorrent et visualisation des trames.	16
3-1. But de l'expérience :	16
3-2. Choix du fichier :	16
3-3. Téléchargement d'un fichier audio et analyse des trames :	17
3.3.1. Handshake : serrer la main	17
3.3.2. Les trames Request et Piece	19
4 - Avenir, danger et dérive : le P2P est-il légal ?	20
5 - Conclusion	22

1 - Introduction

Avant tout, nous avons choisi d'orienter notre TIPE autour du peer-to-peer et plus particulièrement autour du partage fichier. Nous pensons que ce sujet s'inscrit directement dans le thème proposé : Ressource technologiques, numériques et informationnelles : partage, répartition et distribution. Nous allons commencer par une courte introduction sur ce modèle de réseau avant de rentrer dans la problématique a proprement parlé : le partage de fichier.

1.1. Un peu d'histoire

Il est difficile de dater la naissance du modèle de réseau peer-to-peer. En effet, ce dernier est né d'une volonté, dans les années 1960, de créer un modèle de réseau décentralisé et non hiérarchisé. Décentralisé, car le réseau P2P ne dépend, en théorie, d'aucun élément central (ou serveur central) et non hiérarchisé car il n'y a pas de notion de maître/esclave dans les échanges. Cependant, à cette époque, les applications liées au modèle de réseau peer-to-peer étaient relativement peu courantes en partie car Internet ne ressemblait en rien à celui que nous connaissons aujourd'hui. Il faudra attendre les années 90 pour que les premières applications « grand public » voient le jour. Face à la démocratisation d'Internet, les échanges se multiplient (messagerie, transfert de fichier, partage...) et la nécessité d'augmenter sans cesse les infrastructures réseau, afin de répondre au trafic grandissant, devient un problème majeur auquel doivent faire face les opérateurs. En 1999, Shawn Fanning alors étudiant à la Northeastern University de Boston crée le réseau P2P centralisé de partage de musique Napster qui rencontrera un vif succès au point de devenir l'une des applications les plus populaires de l'an 2000. Très vite, face au succès de Napster, les réseaux P2P centralisés et décentralisés se multiplient et prennent très vite une place significative dans les échanges réalisés sur internet.^{1 2}

1.2. Actuellement

A l'heure actuelle, d'après une étude réalisée par Cisco en 2010³, le trafic généré par le peer-to-peer a augmenté de 7 petabytes (7 000 000 de gigabytes) par mois jusqu'en 2014 au point de représenter cette même année environ 27% du trafic d'Internet :

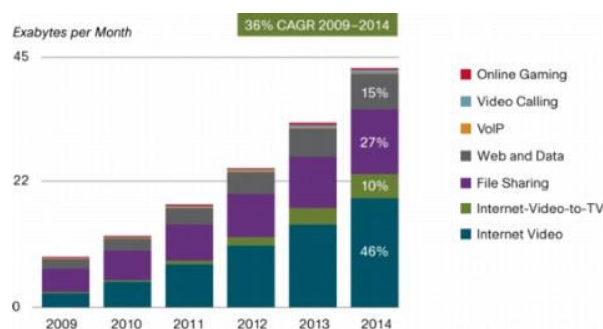


Figure 1 : Ici les échanges P2P sont situés dans la partie File Sharing (source : Cisco via TorrentFreak)⁴

¹ <http://histoire-internet.vincaria.net/post/histoire/internet/1997/Peer-to-peer>

² <http://igm.univ-mlv.fr/~duris/NTREZO/20022003/Peer-to-peer.pdf>

³ http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html

Le peer-to-peer a également connu ces deux dernières années de nouvelles applications telles que les crypto-monnaies comme le Bitcoin⁵ ou le Litecoin par exemple ou encore le « torrent-streaming » via des logiciels comme Popcorn Time BitTorrent streaming ou le site web Joker.⁶ Ces dernières années ont également été marquées par la création dans de multiples pays de législations visant à encadrer les échanges de fichier via les divers réseaux peer-to-peer comme la loi Hadopi en France ou encore la décision de blocage du site The Pirate Bay par le tribunal de grande instance de Paris en décembre 2014.⁷

2 - Fonctionnement du modèle de réseau

2.1. Un nouveau modèle de réseau

Le modèle de réseau peer-to-peer est un modèle différent de la vision client-serveur traditionnelle. En effet, chaque client qui possède tout ou partie d'un fichier devient lui-même « serveur » et peut partager ce fichier, ou une partie, avec d'autres clients membres du réseau. Il est facile de représenter les deux modèles de réseau grâce à deux schémas pour mieux comprendre leur fonctionnement (voir schémas ci-dessous).

2.1.1. Modèle client/serveur classique

Dans un premier temps, nous allons nous pencher sur la vision client/serveur classique :



Figure 2 : Modèle client / serveur (source : Wikipédia)⁸

Dans ce modèle de réseau, les postes informatiques sont les clients, ils sont tous reliés au même serveur. Ce modèle est dit centralisé, car tous les postes dépendent d'un seul serveur central qui gère le réseau. Ce modèle présente nombreux avantages, mais aussi des inconvénients. En effet, cette vision présente des taux de transfert relativement stables qui dépendent en partie du nombre d'utilisateurs présents sur le serveur et du débit de transfert alloué par les fournisseurs d'accès (du client et du serveur). Ainsi, moins un serveur est utilisé, plus les débits sont rapides. Cependant la disponibilité du fichier dépend directement du serveur. Ainsi si le fichier est supprimé ou que le serveur est surchargé le fichier devient à son tour inaccessible.⁹

⁴ <http://torrentfreak.com/cisco-expects-p2p-traffic-to-double-by-2014-100611/>

⁵ Le Bitcoin est une crypto-monnaie créée en 2009 par Satoshi Nakamoto, Gavin Andresen. Cette monnaie se veut décentralisée et infalsifiable.

⁶ Un logiciel de streaming (lecture en continue d'un flux) vidéo permet de regarder un flux audio sans avoir à le télécharger dans sa totalité au préalable.

⁷ http://www.lemonde.fr/pixels/article/2014/12/05/la-justice-ordonne-le-blocage-de-the-pirate-bay-en-france_4534837_4408996.html

⁸ http://fr.wikipedia.org/wiki/Pair_à_pair

⁹ <http://fr.wikipedia.org/wiki/Client-serveur>

2.1.2. Modèle de réseau peer-to-peer

Nous allons maintenant nous intéresser au modèle de réseau peer-to-peer. Il faut en premier lieu comprendre que dans un réseau peer-to-peer, les fichiers ne sont pas stockés sur un serveur central, mais directement chez le client (appelé pair ou peer en anglais) qui le met à disposition des autres pairs sur le réseau :

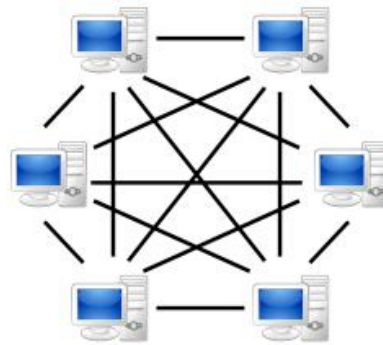


Figure 3: Module Peer-to-Peer (source: Wikipedia)¹⁰

Ainsi, chaque pair qui possède le fichier devient à son tour « serveur » ou distributeur du fichier. Donc plus un fichier est populaire sur le réseau plus il est disponible, et plus son téléchargement est rapide. De plus, chacun des pairs peut partager le fichier même s'il ne le possède pas complètement, ce qui permet entre autre de gagner en vitesse. On supprime donc la contrainte du faible débit imposée par des serveurs surchargés. Même si le fichier n'est plus disponible chez un des pairs, d'autres peuvent l'avoir téléchargé au préalable et ainsi conserver la disponibilité du fichier sur le réseau. Cependant, si un fichier est peu partagé, ses taux de transfert sont très bas, voire nuls. Les pairs peuvent être sélectionnés en fonction de leur rapidité de transfert qui peut dépendre entre autre de la proximité géographique mais aussi de la qualité de la connexion et du débit de partage alloué par le fournisseur d'accès à internet.¹¹

Il est important de différencier deux architectures propres au modèle de réseau peer-to-peer :

- L'architecture décentralisée
- L'architecture centralisée

a. Modèle de réseau peer-to-peer décentralisé

Dans le modèle décentralisé, on s'affranchit complètement de la notion de serveur au sens propre du terme. Ici chaque client est lui-même serveur, les pairs sont donc tous sur un même pied d'égalité. De plus l'anonymat est respecté puisqu'il n'y a pas de serveur central collectant tous les pairs sur le réseau.

¹⁰ http://fr.wikipedia.org/wiki/Pair_à_pair

¹¹ <http://igm.univ-mlv.fr/~duris/NTREZO/20022003/Peer-to-peer.pdf>

On peut facilement se représenter cette architecture sous forme de schéma :

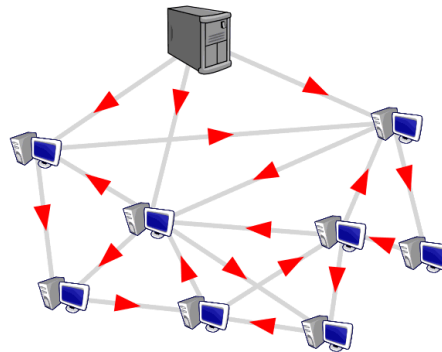


Figure 5 : Architecture décentralisée (source : sebsauvage) ¹²

Remarque : Dans ce schéma, le serveur fait partie du réseau mais n'occupe pas une place centrale. Il se comporte de la même manière que les postes : s'il est déconnecté le réseau fonctionne toujours sans aucun changement.

Cependant, cette architecture pose un problème majeur : celui de la topologie du réseau. En effet, pour qu'un nouveau pair rejoigne le réseau, il faut que ce dernier se manifeste par un message de broadcast afin d'identifier et d'être identifié par les autres pairs du réseau. Un message de broadcast étant une requête envoyé à tous les postes connectés à un même réseau (diffusion en français).

Une fois sur le réseau, le pair dispose de tous les avantages de celui-ci. En effet une telle architecture permet en théorie d'avoir un réseau de pairs infini car il n'y a aucun serveur qui limite le nombre de pair en fonction de sa puissance. De plus, il est facile d'aller et venir sur le réseau (simple message de broadcast), ce qui permet aussi de renouveler son anonymat sur le réseau. Seuls les pairs auxquels nous sommes connectés directement peuvent nous repérer.

Cependant, ce type de réseau pose de nombreux problèmes de fiabilité. En effet, on ne peut pas quantifier le temps mis par une requête pour atteindre son destinataire, pas plus qu'on ne peut être sûr que cette dernière arrivera bien à son destinataire. La notion de qualité des pairs est ici absente car on ne peut pas renouveler les pairs auxquels on se connecte sauf en cas de déconnexion d'un des pairs ou de nous-même. ^{13 14}

b. Modèle de réseau peer-to-peer centralisé

Bien que le P2P ait pour principal atout d'être décentralisé, certains réseaux (comme Napster) ont fait le choix d'opter pour une architecture centralisée.

¹² <http://sebsauvage.net/comprendre/p2p/>

¹³ <http://igm.univ-mlv.fr/~duris/NTREZO/20022003/Peer-to-peer.pdf>

¹⁴ <http://madchat.fr/reseau/P2P/%5BDocument%5D%20Etude%20des%20reseaux%20P2P.pdf>

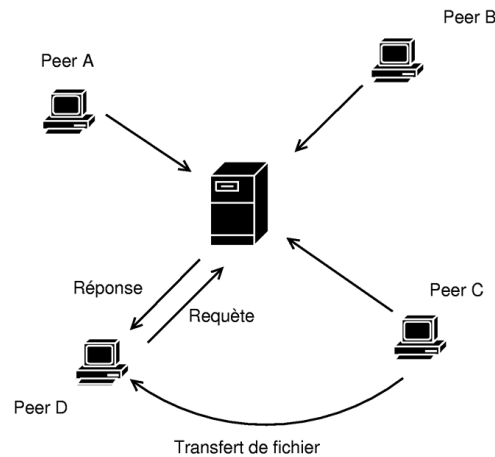


Figure 3 : Réseau Peer-to-Peer à architecture centralisée monoserveur (source : *Developez*)¹⁵

On remarque ici que les transferts de fichiers se font toujours entre deux postes informatiques, ou plus, (chacun des postes est nommé peer ou pair en français). On est donc bien dans un modèle de réseau peer-to-peer. Cependant, il y a la présence d'un serveur central auquel tous les pairs sont connectés. Les serveurs sont des postes informatiques puissants capables de gérer un grand nombre de requêtes. Cette architecture est un peu le chaînon entre le P2P décentralisé et la vision client-serveur. Ici le serveur central dialogue avec chacun des pairs mais ne possède pas les fichiers. Il remplit cependant plusieurs fonctions essentielles au bon fonctionnement du réseau :

- Il connaît le nom, la taille, le format et l'adresse de chacun des fichiers présents sur le réseau
- Il recense en temps réel les utilisateurs présents ou non sur le réseau, leurs noms, leurs adresses ainsi que les fichiers qu'ils possèdent

L'avantage majeur de ce type d'architecture réside dans le fait qu'il n'y a pas de problème de connexion au bon pair pour télécharger la ressource. En effet, le serveur sait à l'avance quel pair possède ou non le fichier demandé et peut ainsi établir une liste des pairs utilisable pour télécharger le fichier. Il n'y a pas non plus de difficulté de connexion au bon serveur car chaque pair possède le même logiciel configuré au préalable par l'administrateur du réseau. La recherche de fichier est également facilitée car le serveur liste au préalable les fichiers disponibles.

Le principal inconvénient est la sécurité. En effet, si le serveur central est hors ligne (souvent à cause du nombre de requêtes trop important), le réseau est totalement inutilisable. De plus, le serveur central connaît tous les pairs qui y accèdent et cela crée un problème d'anonymat.¹⁶

Pour pallier au problème de disponibilité du serveur, il est possible de mettre en place un système de file d'attente pour accéder au serveur ou encore de s'intéresser à une architecture multiserveurs :

¹⁵ http://schuler.developez.com/articles/p2p/images/p2p_central.gif

¹⁶ <http://schuler.developez.com/articles/p2p/#LIII-A>

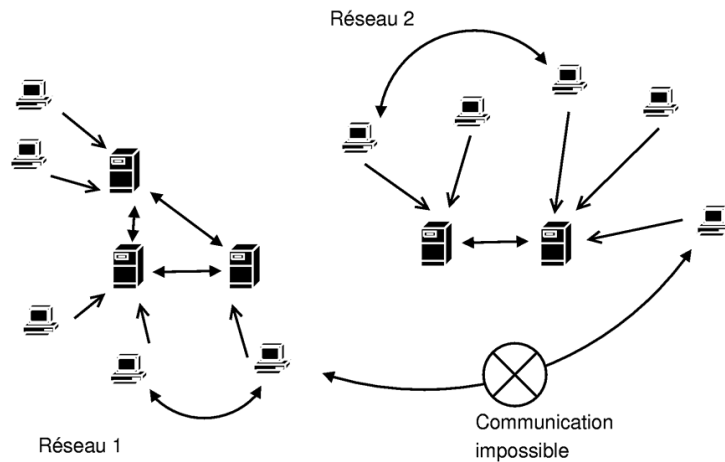


Figure 4 : Architecture centralisée multiserveurs (source : Developpez)¹⁷

Dans le cas du multiserveurs, les pairs sont repartis sur plusieurs serveurs, tous reliés entre eux, ce qui limite la quantité d'information à traiter par un seul serveur et ainsi éviter le plus possible le plantage.¹⁸

2.2. Les différents réseaux de partage de fichiers

Il existe plusieurs plateformes publiques de partage de fichier via Peer-to-Peer. Chacune de ces plateformes utilise des protocoles particuliers nécessitant de posséder un logiciel de partage de fichiers adapté comme par exemple eMule pour le réseau eDonkey. Chacune de ces plateformes forme donc un réseau de fichier singulier : en effet, chaque plateforme présente ses particularités et des fonctionnements différents. Un fichier partagé sur une d'elles n'est pas disponible sur toutes les autres. Ainsi les logiciels ont tendance à se spécialiser dans un type de fichier ; par exemple, eDonkey contient en grande partie des musiques et des films.^{19 20}

Cependant, on observe plus tard l'apparition de certains logiciels comme Shareaza qui supportent plusieurs de ces réseaux. En effet, Shareaza supporte à la fois les réseaux eDonkey, Gnutella, Gnutella2.²¹

2.3. Le Hashage :

Une des grandes raisons de la fiabilité du P2P dans le domaine du partage est la méthode de hashage des fichiers. En effet, sans ce dernier, la gestion des fichiers serait extrêmement compliquée. Le but premier du hashage est d'identifier un fichier par un code complexe généré automatiquement par le logiciel, appelé hash, qui sera unique pour chaque fichier ; contrairement au nom qu'aurait pu lui donner un utilisateur. Ainsi, l'identification du fichier est indépendante du nom de ce dernier, deux fichiers différents auront obligatoirement des hash différents même si leur nom est le même. Dans les logiciels de P2P, le hash est le seul paramètre pris en compte pour identifier un fichier.

¹⁷ http://schuler.developpez.com/articles/p2p/images/p2p_serveurs.gif

¹⁸ <http://schuler.developpez.com/articles/p2p/#LIII-B>

¹⁹ <http://litis.univ-lehavre.fr/~duvallet/enseignements/Cours/M1INFO/Reseau/MI-Cours-Reseau-Cours9.pdf>

²⁰ <http://fr.wikipedia.org/wiki/EDonkey2000?oldformat=true>

²¹ <http://shareaza.sourceforge.net/?lang=fr>



Figure 5 : Fichiers différents ayant le même nom (Source : sebsauvage)²²

Sur le schéma ci-dessus, on voit trois fichiers images ayant le même nom et la même extension mais ne contenant pas la même image. Leur contenu étant différent, leur empreinte SHA-1 sera donc différente.

Finalement, un autre avantage du hachage est qu'il optimise la gestion des fichiers. En effet, le hash étant indépendant du nom du fichier, deux fichiers identiques avec des noms différents auront le même hash. Cela permet donc d'éviter la création de doublons à chaque fois qu'un utilisateur télécharge un fichier et le rend disponible sur le réseau sous un autre nom. De plus, cela offre une plus grande disponibilité des fichiers car tous les peers possédant le même fichier sont regroupés, quel que soit le nom de ce dernier.



Figure 5 : Fichiers identiques ayant des noms différents (Source : sebsauvage)²³

Sur ce schéma, on voit trois fichiers images renommés différemment ayant le même contenu, ils auront donc la même empreinte SHA-1 et pourront être distribués indépendamment de leur nom à un leecher souhaitant obtenir ce fichier image.

Pour finir, le hash permet, lors de l'échange de fichiers, de vérifier l'intégrité d'un fichier. En effet, le logiciel destinataire recalcule le hash et vérifie qu'il soit identique à celui qui lui a été transmis. Cette méthode est très fiable car même si l'erreur est infime, il y a très peu de chance qu'elle ne soit pas détectée : même si l'on modifie le fichier d'un seul bit, l'impact sur le hash sera très important. Par exemple, le hash du mot « Partage » (en SHA-1) est «000b836ba5ce060338d1c0b31fc 568f201b059d9 » et si on ajoute simplement un espace à la fin du mot, le hash devient « 748ec4c8b28ba85ce1d5ef073640bafae37eb24 ». Le résultat serait encore entièrement différent si l'on ne mettait pas de majuscule au mot Partage.

Il existe de nombreux types de hachage tels que le MD4, MD5, le Tiger, ou le SHA-0 ; mais nous allons détailler le fonctionnement d'un seul hachage car il est très utilisé de nos jours notamment dans le protocole BitTorrent : le SHA-1.

²² <http://sebsauvage.net/comprendre/p2p/fichiers1.gif>

²³ <http://sebsauvage.net/comprendre/p2p/fichiers2.gif>

2.1.3. Le SHA-1 :

Le SHA-1 est une des méthodes les plus utilisées actuellement pour le hashage des fichiers. Il a été mis en place par la NSA avec son ancêtre le SHA-0. Son fonctionnement est relativement complexe. Pour résumer, il prend un message d'entrée codé au maximum sur 2^{64} bits soit approximativement 20×10^{18} bits. L'algorithme traite ensuite ce message en plusieurs étapes de la façon suivante :

- Tout d'abord, le message d'entrée sera composé des bits composant le fichier. Il dépendra donc de la taille et du contenu du fichier et non de son nom.
- Pour commencer, on utilise **quatre fonctions booléennes** qui prennent 3 mots de 32 bits en paramètre et retournent seulement un mot de 32 bits. Cette première étape permet, grâce à plusieurs règles de calculs, de réduire le message et de commencer à le coder en fonctions de tous les bits entrés en paramètres. C'est notamment pendant cette étape que se fait la différence avec son précurseur le SHA-0 : la fonction de rotation permettant de déplacer les bits vers la gauche de façon circulaire avait une structure linéaire dans le SHA-0 (ce qui la rendait attaquable) qui a été corrigée dans le SHA-1.
- On ajoute ensuite un bit à 1 à la fin du message obtenu puis une série de bit à 0 suivi du message initial codé sur 64 bit. À cette étape, le message est obligatoirement **un multiple de 512** (on adapte la taille de la série de 0).
- Suite à cela, on calcule 80 valeurs de 32 bits, en effectuant 80 tours d'un algorithme réalisant des transformations sur le résultat obtenu précédemment. Afin d'effectuer ces transformations, l'algorithme utilise des constantes calculées à partir du message d'entrée. Elles sont donc différentes d'un message à l'autre.
- Le SHA-1 applique ensuite de nombreuses opérations : addition, soustraction, rotation utilisant les constantes spécifiées par le standard ou calculées précédemment. On effectue à nouveau 80 tours de cet algorithme. Chaque tour met à jour certaines constantes. En résumé, le SHA-1 change totalement de règle de calcul tous les 20 tours. Cela rend donc l'algorithme totalement imprévisible.
- Pour finir, on additionne ce résultat avec la valeur initiale. On obtient une signature sur **160 bits** qui sera le hash en SHA-1.

Comme nous venons de le voir, le hash en SHA-1 est très complexe à calculer, et les nombreuses rotations entraînent d'importants changements dans le hash même pour un changement infime du message d'entrée. Ceci en fait donc un bon indicateur de l'intégrité du fichier et un bon générateur de nombres aléatoires, ces nombres étant très difficiles à obtenir en informatique.²⁴

²⁴ <http://fr.wikipedia.org/wiki/SHA-1>

2.4. Le protocole BitTorrent

Le BitTorrent est un protocole de communication basé sur le P2P permettant l'échange de données. Ce protocole est conçu en 2001 et gagne très vite en popularité si bien qu'en 2004 une société américaine du nom BitTorrent Inc.²⁵ crée et commercialise des services autour de ce protocole (torrent-streaming, synchronisation des fichiers sur divers appareils via le protocole...) ; en 2007 elle retire le protocole du monde Open Source. Ce protocole est le plus utilisé aujourd'hui car il est très facile d'utilisation et il est bien plus difficile de tracer un utilisateur sur le réseau.

2.4.1. Partage de fichier via le protocole BitTorrent

Étant donné qu'il n'existe pas de réseau BitTorrent avec ses utilisateurs et un annuaire associé (comme sur eDonkey ou Gnutella), il est nécessaire de bien comprendre comment on partage l'information avec ce protocole.

L'utilisation de ce protocole nécessite d'avoir :

- Un logiciel appelé client BitTorrent (comme BitTorrent, µTorrent, Vuze...)
- Un fichier .torrent (de quelques ko) ou un lien magnet associé au fichier que l'on veut télécharger

Penchons-nous sur le fichier .torrent, ce fichier contient toutes les informations dont a besoin le client BitTorrent pour télécharger le fichier à savoir : ²⁶

- Le ou les trackers associés au fichier. Le tracker est un serveur informatique qui va permettre de relayer les informations de partage entre les pairs. Chaque pair est repéré par un identifiant unique sur le tracker. Le tracker connaît donc les pairs disponibles ayant le fichier que l'on souhaite télécharger. A l'inverse le tracker peut nous enregistrer comme possesseur d'un fichier que l'on souhaite partager. Le tracker le plus utilisé est Open BitTorrent.^{27 28}
- Le hash du fichier qui est l'empreinte numérique du fichier calculée avec une méthode de calcul particulière²⁹ (le SHA1 est la méthode de hachage la plus utilisée pour les transferts via le protocole BitTorrent)
- Le nom, la description du fichier
- La date et le logiciel utilisé pour la création du fichier .torrent

Une fois que l'on a créé un fichier .torrent, on le diffuse sur Internet aux personnes intéressées qui pourront venir récupérer le fichier et devenir à leur tour distributeur du fichier. L'ensemble du processus de récupération d'un fichier via le protocole BitTorrent est expliqué ci-après.

Nous avons essayé de notre côté de créer un fichier .torrent pour le rapport de notre TIPE. Pour se faire, nous avons utilisé le client µTorrent qui permet aussi de créer un fichier torrent. La fenêtre de création de torrent se présente de la forme suivante :

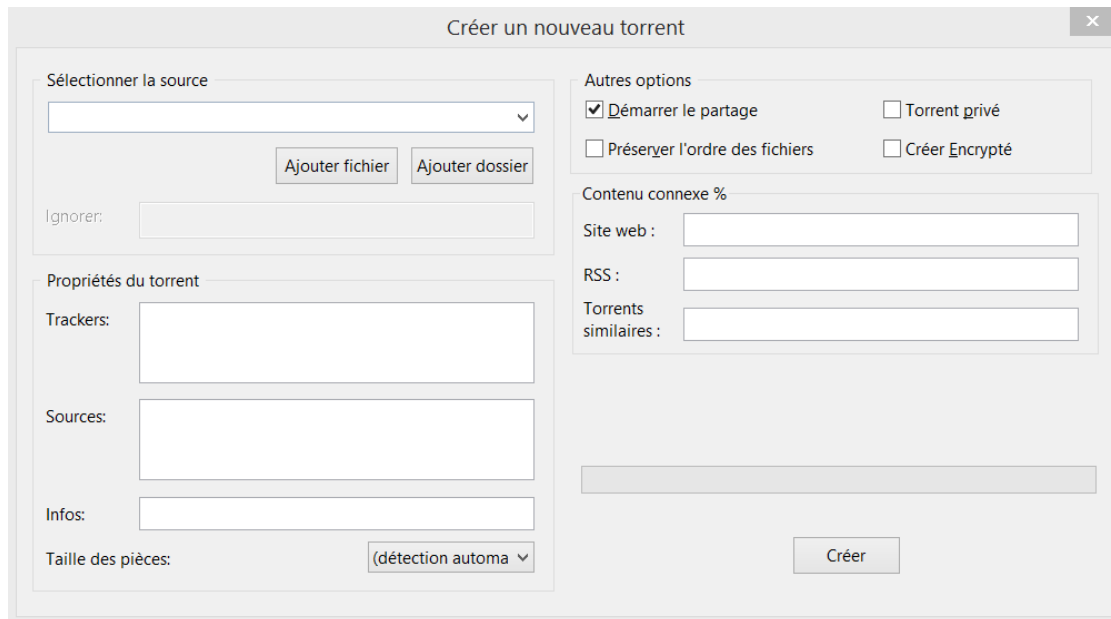
²⁵ <http://www.bittorrent.com/intl/fr/>

²⁶ <http://www-igm.univ-mlv.fr/~dr/XPOSE2004/bittorrent/structure.html>

²⁷ http://fr.wikipedia.org/wiki/Tracker_BitTorrent

²⁸ <http://korben.info/openbittorrent-le-tracker-libre-successeur-de-thepiratebay.html>

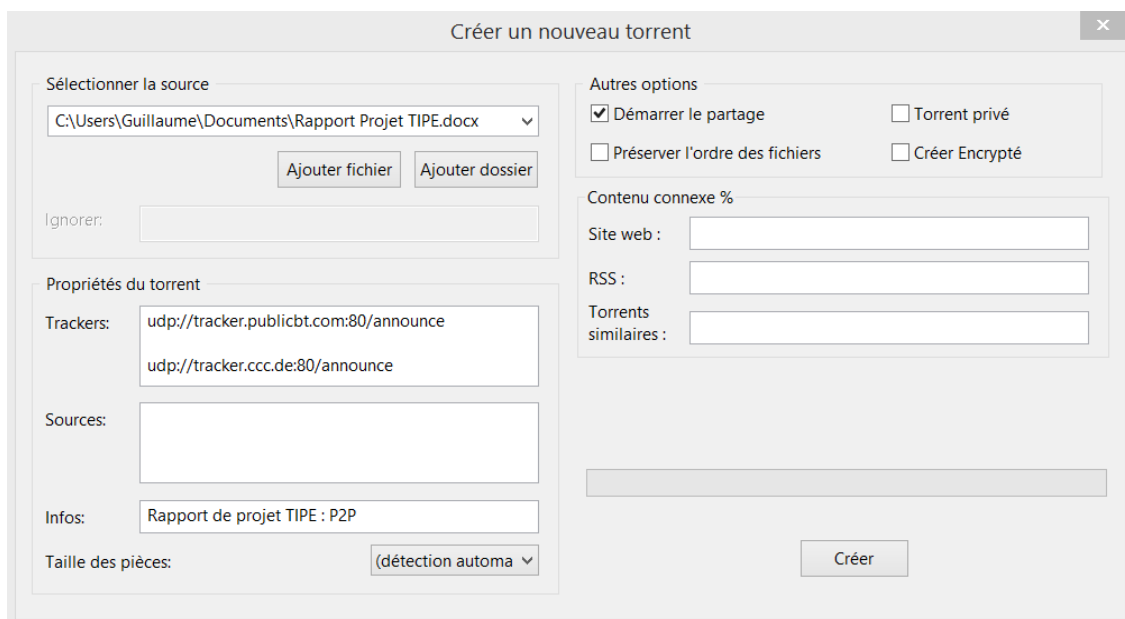
²⁹ http://fr.wikipedia.org/wiki/Fonction_de_hachage



Nous avons choisi d'utiliser 3 trackers connus, très utilisées³⁰ et qui ne nécessitent pas d'inscription préalable à leurs utilisations :

- `udp://tracker.openbittorrent.com:80/announce`
- `udp://tracker.publicbt.com:80/announce`
- `udp://tracker.ccc.de:80/announce`

On remarque que les adresses des trackers commencent tous par `udp://` qui est une partie du protocole TCP/IP appartenant à la couche transport (couche 4 du modèle OSI). UDP signifie User Datagram Protocol et permet la transmission de données de manière très simple. Ici on voit que l'on se connecte à ces trackers via le port 80.³¹ Une fois la fenêtre de création complétée on obtient :



³⁰ http://fr.wikipedia.org/wiki/Tracker_BitTorrent?oldformat=true#Trackers_public

³¹ http://fr.wikipedia.org/wiki/User_Datagram_Protocol?oldformat=true

En appuyant sur créer, on obtient le fichier torrent.

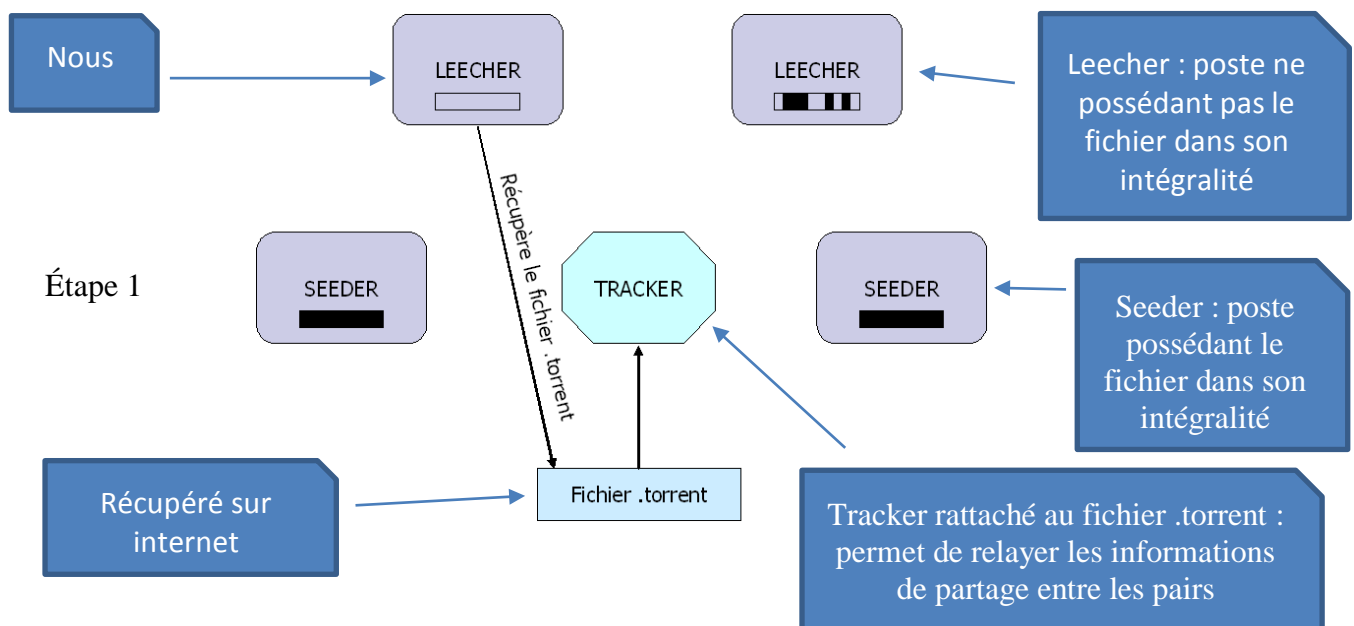
On aurait pu également partager le fichier via un lien magnet. Le lien magnet est une alternative au fichier .torrent qui est de plus en plus utilisée de nos jours car elle ne nécessite pas le stockage sur des serveurs distants des fichiers .torrent³². En effet un simple lien suffit au client BitTorrent pour repérer le fichier et le télécharger. Le lien magnet est de la forme :

 <magnet:?xt=urn:sha1:YNCKHTQCWBTRNJIV4WNAE52SJUQCZO5C>

Le champ xt (extra topic) correspond à l'adresse URN du fichier. URN signifie Uniform Resource Name qui est en fait un identifiant donné à une ressource sur internet qui permet d'identifier une ressource indépendamment de sa localisation³³. Ici dans le lien donné en exemple l'URN donné n'est autre que le hash en SHA1 du fichier. Il est important de noter que le lien magnet permet uniquement d'identifier un fichier. Une recherche par le client BitTorrent est nécessaire pour trouver les pairs associés sur le tracker. Cette recherche se fera sur les trackers usuels (comme Open BitTorrent et Public BT). Cependant, si le tracker dans lequel est référencé le fichier est un tracker privé (c'est-à-dire créer par une personne uniquement pour lui-même ou un nombre limité de personne), il est nécessaire de l'ajouter dans le lien magnet. Pour ajouter un tracker au lien magnet, on rajoute la balise tr suivi du lien du tracker.

2.4.2. Récupérer un fichier via le protocole BitTorrent

Il est possible de résumer le téléchargement d'un fichier via le protocole BitTorrent avec 6 schémas-étapes³⁴ :

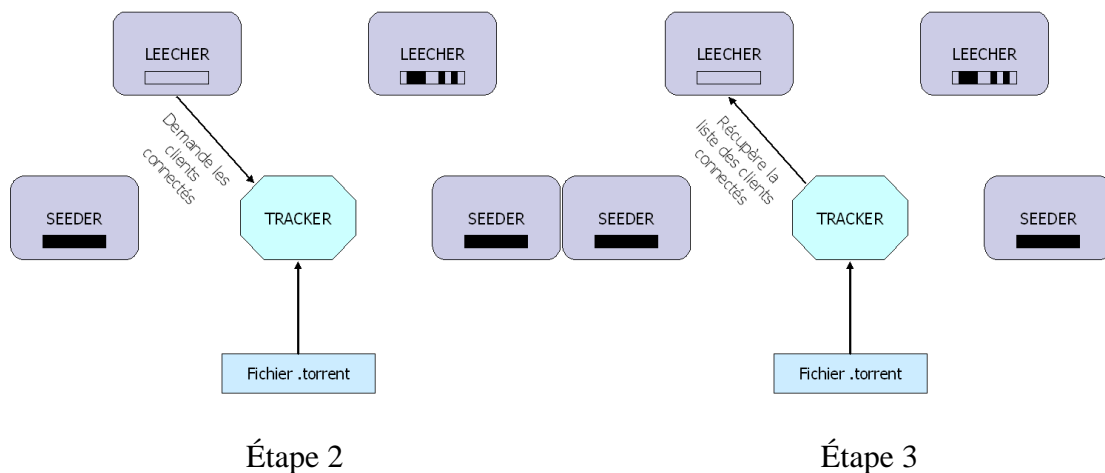


Dans la première étape, un leecher (c'est-à-dire un pair qui ne possède pas la totalité du fichier) récupère un fichier .torrent sur la toile. Ce fichier est rattaché à un ou plusieurs trackers. Un seeder est un pair qui possède la totalité du fichier. L'ensemble des seeders et des leechers forment le swarm (l'essaim).

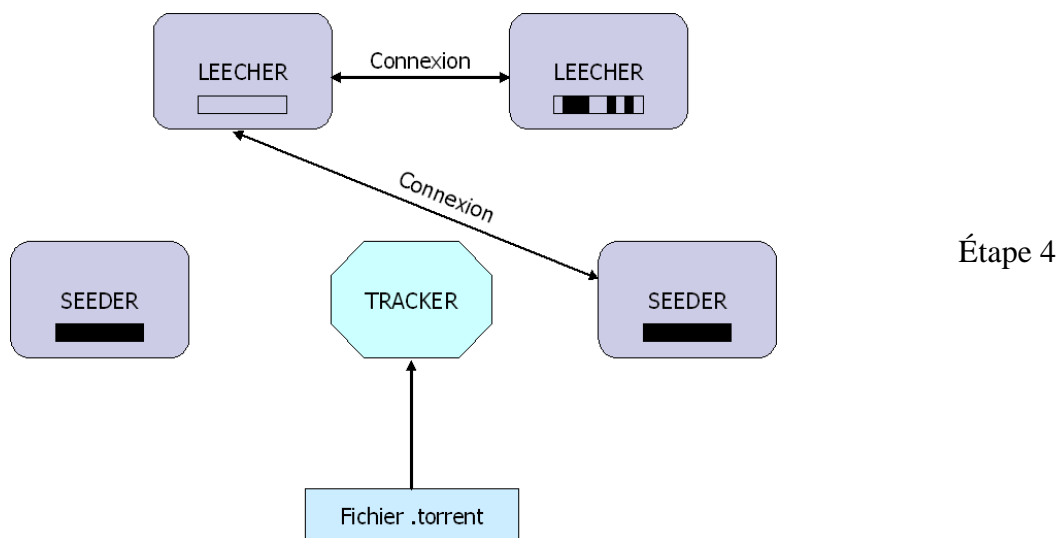
³² [http://fr.wikipedia.org/wiki/Magnet_\(standard\)](http://fr.wikipedia.org/wiki/Magnet_(standard))

³³ http://fr.wikipedia.org/wiki/Uniform_Resource_Name

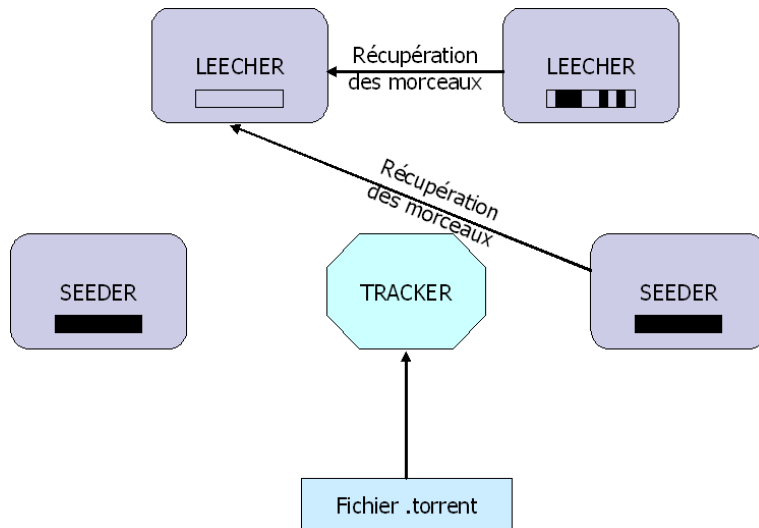
³⁴ <http://www-igm.univ-mlv.fr/~dr/XPOSE2004/bittorrent/echanges.html>



Les étapes 2 et 3 sont primordiales avant la mise en relation des pairs entre eux. Le leecher qui veut récupérer le fichier rattaché au fichier .torrent demande au(x) tracker(s) associé(s) au fichier .torrent la liste des clients connectés et récupère cette liste. Grâce à cette liste, le leecher va pouvoir se connecter à l'étape suivante aux autres pairs.

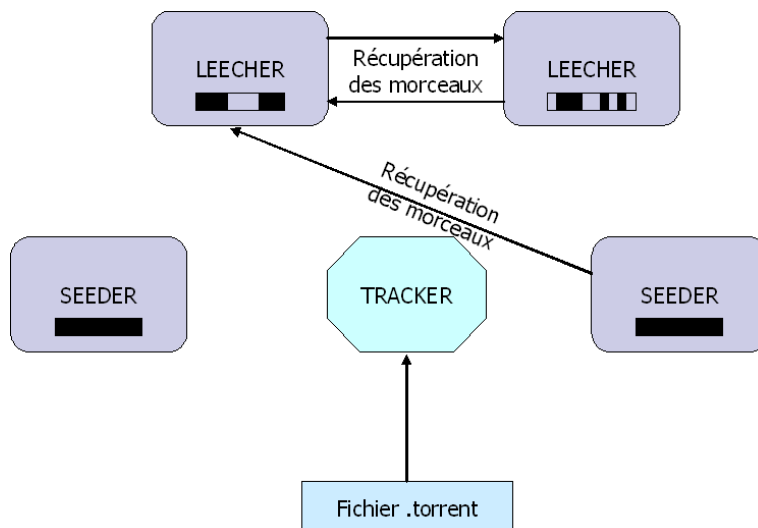


Dans cette quatrième étape le leecher qui veut récupérer le fichier se connecte aux pairs qui l'intéressent et envoie une trame de Handshake (serrage de main). Parfois cette trame peut aussi être envoyée au tracker qui tient une liste de l'ensemble des connexions entre les pairs. On remarque que l'on peut entrer en relation avec un pair qui n'a pas terminé de télécharger le fichier. Cela vient du fait lorsqu'on télécharge un fichier via le protocole BitTorrent, on récupère des bouts de fichiers (appelé piece) que l'on assemble ensuite. Chaque pair sait quelle piece il possède et quelle piece il ne possède pas.



Étape 5

Une fois la connexion entre les pairs établie, le téléchargement peut commencer. Le téléchargement revient à récupérer les morceaux du fichier chez différents pairs et ceci s'accorde sur deux type de trames différentes : la trame request et la trame piece. L'ensemble des trames utiles à la récupération d'un fichier seront expliquées lors de l'expérimentation.



Étape 6

Au fur et à mesure que notre fichier se télécharge, il est possible qu'un leecher à qui nous sommes connectés ait besoin de récupérer une pièce chez nous. Dans le cas du schéma, on voit que l'on possède certaines parties du fichier que nous avons récupéré chez le seeder et que le leecher auquel nous sommes connecté ne les possède pas. Nous allons donc pouvoir partager ces parties avec lui.

Ces 6 schémas nous ont permis de comprendre comment fonctionne la récupération d'un fichier via le protocole BitTorrent. On a vu que récupérer un fichier via ce protocole nécessitait d'avoir un swarm important autour du fichier si l'on voulait avoir des débits convenables. Nous allons donc maintenant essayer de voir si la théorie exprimée plus haut sur le protocole BitTorrent se vérifie dans la pratique. Nous allons tenter de télécharger un fichier via ce protocole et analyser les trames reçues et envoyées.

3 - Expérimentation : téléchargement d'un fichier via le protocole BitTorrent et visualisation des trames.

Pour notre expérimentation, nous avons choisi de réaliser une capture de trames avec le logiciel Wireshark ³⁵ pendant le téléchargement d'un fichier via le protocole BitTorrent. Wireshark est un logiciel libre permettant d'analyser des paquets. Il est couramment utilisé dans l'analyse de réseaux informatiques ou encore le développement de protocoles.

Pour l'expérimentation, il aurait pu être intéressant de réaliser un logiciel de partage de fichier basé sur le P2P. Cependant ce projet était beaucoup trop ambitieux et surtout volumineux dans le cadre du TIPE. Nous avons donc préféré développer les trames essentielles au bon fonctionnement d'un transfert de fichier via le protocole BitTorrent.

3-1. But de l'expérience :

L'expérience consistera à télécharger un fichier via le protocole BitTorrent avec le logiciel μ Torrent et d'analyser les trames envoyées et reçues par l'ordinateur.

Lors de cette expérience nous souhaitons mettre en évidence les trames échangées durant la transmission du fichier. Cette expérience nous permettra de mieux comprendre le fonctionnement du BitTorrent et du peer-to-peer centralisé qui est le modèle le plus utilisé aujourd'hui.

3-2. Choix du fichier :

Le choix du fichier est essentiel dans la réalisation de l'expérimentation. Nous avons le choix entre :

- Une image ISO
- Un fichier vidéo
- Un fichier texte
- Un fichier audio

Nous nous sommes orientés vers un fichier de type audio en grande partie à cause de la taille de ce fichier qui oscille entre 3 Mo pour un fichier de type MP3 de qualité médiocre et 40 Mo pour un fichier de type FLAC (format d'encodage de fichier audio sans pertes ni compression). Dans ce type d'expérimentation, la taille du fichier est primordiale car il faut pouvoir capturer assez de trames afin d'avoir un échantillon représentatif (c'est-à-dire avec un nombre de trames différentes relativement important) mais aussi pouvoir répéter l'expérimentation un nombre assez important de fois (en introduisant des perturbations par exemple). Dans le cas d'une image ISO (comme un live CD d'Ubuntu / Debian) ou un fichier vidéo la taille la plus courante oscille autour de 700 Mo / 1 Go. Ce type de fichier permet donc d'avoir un nombre de trames échangées très important mais ne permet pas de renouveler l'expérience un grand nombre de fois. Dans le cas d'un fichier texte maintenant, la taille est trop faible (quelques octets ou kilo-octets) pour avoir un échantillon de trames assez important.

³⁵ <https://www.wireshark.org/>

Nous n'avons pas choisi de prendre le fichier que nous avons créé précédemment car nous étions la seule source du partage et il aurait été difficile de créer un swarm suffisamment intéressant.

Le fichier audio nous semblait donc une bonne alternative pour pouvoir répéter l'opération dans diverses conditions, mais aussi pour avoir un échantillon de trames suffisant.

Les caractéristiques du fichier sont les suivantes :

- Hash SHA1 : 7a:63:e3:2b:3e:8b:65:16:66:0c:14:1e:89:ae:1b:91:f2:6c:b5:af
- Taille : 5.19 Mo
- Longueur : 3 minutes 46 secondes
- Taux d'échantillonnage : 192 Kbits/s (qualité faible / très moyenne)
- Format : MP3

3-3. Téléchargement d'un fichier audio et analyse des trames :

Dans cette première partie, nous avons téléchargé le fichier décrit ci-dessus sans y ajouter de perturbations volontaires. C'est-à-dire que nous n'avons pas utilisé l'ordinateur pendant toute la durée du téléchargement pour éviter l'envoi requête vers des machines distantes autre que celles envoyées par µTorrent. Bien entendu, des services utiles à Windows (tel que Windows Update) tournent en parallèle.

Durant la réalisation de cette expérimentation, Wireshark a capturé un total de 35 009 paquets mais, après ajout d'un filtre pour isoler les paquets propres au protocole BitTorrent, seuls 47 paquets étaient intéressants. Ceci nous a permis de voir que les tâches de fonds qui utilisent le réseau et les services Windows échangent beaucoup d'informations sans que l'utilisateur ne s'en aperçoive.

Parmi ces 47 paquets, nous avons pu relever 10 trames BitTorrent différentes. Certaines trames étant peu fréquentes, nous ne nous attarderons pas sur celles-ci.

Nous avons décidé de nous pencher sur 3 trames essentielles :

- La trame handshake
- La trame request
- La trame piece

Les 7 autres trames que nous avons relevées durant ce transfert n'apparaissent que peu de fois. Sur les autres captures que nous avons pu réaliser, nous avons remarqué que parfois elles n'étaient même pas présentes.

3.3.1. Handshake : serrer la main

Le Handshake³⁶ est la trame initiale échangée entre les clients via le tracker. C'est la mise en relation entre les pairs ; il y a une trame Handshake obligatoire à chaque fois que l'on veut se connecter à un pair supplémentaire. Cette trame est structurée de la manière suivante :

pstrlen (1 octet)	pstr (pstrlen octets)	reserved (8 octets)	info_hash (20 octets)	peer_id (20 octets)
-------------------	-----------------------	---------------------	-----------------------	---------------------

³⁶ <http://www-igm.univ-mlv.fr/~dr/XPOSE2004/bittorrent/inter.html>

Expliquons chaque maillon de cette trame :

- Pstrlen représente la longueur de la chaîne
- Pstr est l'identifiant du protocole
- Reserved sont 8 octets réservés (généralement à 0)
- Info_hash est le hash (empreinte) du fichier que l'on télécharge, cette partie de trame est très importante car elle permet de faire la distinction entre les fichiers que l'on télécharge. Ceci permet donc de télécharger plusieurs fichiers à la fois.
- Peer_id est l'identifiant unique du pair. Ce dernier est donné par le tracker

Dans notre expérience nous avons pu isoler une trame handshake à l'aide de Wireshark ; elle se présente de la manière suivante :

No.	Time	Source	Destination	Protocol	Length	Info
11650	385.906477000	192.168.1.59	46.239.239.20	BitTorrent	122	Handshake
11666	386.096592000	192.168.1.59	70.65.184.113	BitTorrent	122	Handshake
11691	386.426973000	70.65.184.113	192.168.1.59	BitTorrent	172	Handshake

Ici on voit que nous nous sommes mis en relation avec deux pairs via deux trackers différents (un d'adresse 70.65.184.113 et l'autre 40.239.239.20), on va donc pouvoir échanger des trames pour récupérer le fichier. On voit également à la dernière ligne que le tracker d'adresse 70.65.184.113 a également une demande de Handshake avec nous (on voit que notre adresse est destinataire de la trame). Cela signifie que nous allons également envoyer les morceaux de fichiers que l'on a récupérés.

Nous nous sommes ensuite intéressés à la trame proprement dite et en particulier à ce qu'elle contient, nous avons choisi la seconde trame de notre tableau ci-dessus :

Remarque : la trame entière contient beaucoup plus d'informations (information sur le protocole Ethernet, sur le protocole TCP/IP...) on ne s'intéressera qu'à la partie concernant le BitTorrent.

```

Frame 11666: 122 bytes on wire (976 bits), 122 bytes captured (976 bits) on interface 0
Ethernet II, Src: IntelCor_4b:b1:c6 (5c:51:4f:4b:b1:c6), Dst: Sagemcom_21:f0:4e (2c:39:96:21:f0:4e)
Internet Protocol Version 4, Src: 192.168.1.59 (192.168.1.59), Dst: 70.65.184.113 (70.65.184.113)
Transmission Control Protocol, Src Port: 7280 (7280), Dst Port: 6881 (6881), Seq: 1, Ack: 1, Len: 68
BitTorrent
  Protocol Name Length: 19
  Protocol Name: BitTorrent protocol
  Reserved Extension Bytes: 0000000000100005
  SHA1 Hash of info dictionary: 7a63e32b3e8b6516660c141e89ae1b91f26cb5af
  Peer ID: 2d5554333432302d7a93f3e52004ab8458fd42e3

```

On voit que l'on retrouve bien les 5 parties de la trame étudiée plus haut. On remarque également que les bits réservés sont presque tous à zéro. On retrouve le hash SH1 de notre fichier qui est exactement le même que celui donné dans la description du fichier.³⁷

On note que le tracker d'adresse 70.65.184.113 a son pair_id égal à : 2d:55:54:33:34:32:30:2d:7a:93:f3:e5:20:04:ab:84:58:fd:42:e3. La pair_id permet de créer plus d'anonymat car pendant le handshake, on ne connaît pas encore l'adresse du pair avec qui on va échanger mais simplement son ID sur le tracker.³⁸

³⁷ Rappelons que le hash donné dans la description du fichier était :

7a:63:e3:2b:3e:8b:65:16:66:0c:14:1e:89:ae:1b:91:f2:6c:b5:af

³⁸ Chaque pair dispose d'un pair ID différent

Dans la suite du fichier de capture suivent 6 demandes (de notre part ou de la part du tracker) de handshake. Le transfert du fichier n'a pas encore commencé.

3.3.2. Les trames Request et Piece

Viens ensuite le transfert du fichier qui s'accorde autour de deux trames : ³⁹

- La trame request : utilisée pour demander une partie des données
- La trame piece : la trame contenant le morceau de fichier demandé avec la trame request

Sous Wireshark on peut observer ces trames :

No.	Time	Source	Destination	Protocol	Length	Info
14072	442.88215600	192.168.1.59	70.65.184.113	BitTorrent	122	Request, Piece (Idx:0x4c,Begin:0xc000,Len:0x4000)
14722	444.49896200	70.65.184.113	192.168.1.59	BitTorrent	1506	Piece, Idx:0x4c,Begin:0xc000,Len:0x4000

En réalité, en une seule fois, on peut demander jusqu'à trois pieces qui seront ensuite envoyées l'une après l'autre, ce qui limite le nombre de trames envoyées.

Analysons la construction de ces trames :

length prefix (4 octets)	message ID (1 octet)	payload ((length prefix + 1) octets)
--------------------------	----------------------	--------------------------------------

- Length prefix est un indicateur donc la valeur varie en fonction de la trame échangée pour une trame de type request, le length prefix est 00013 et pour un trame de type piece il sera 0009+X (X étant le nombre d'octet de donnée transmis).
- Message_ID est l'ID de la trame échangée. L'ID de la trame request est 6 et celui de la trame piece est 7.
- Payload est un champ qui varie en fonction de la trame échangée. Cependant, il est presque identique pour la trame request et piece :

- Trame request :

index (4 octets)	begin (4 octets)	length (4 octets)
------------------	------------------	-------------------

- Trame piece :

index (4 octets)	begin (4 octets)	block (X octets)
------------------	------------------	------------------

Le champ index contient l'indice du morceau (puisque l'on transfère uniquement des parties réduites du fichier), il permet de se repérer sur ce que l'on a déjà transmis ou non. Le champ begin donne une information sur le point de départ des données à transférer à l'intérieur de l'index. Le champ length donne une indication sur le nombre de données à transmettre. Le champ block est en fait la donnée transmise.

Sur Wireshark, on peut observer la trame request :

³⁹ <http://www-igm.univ-mlv.fr/~dr/XPOSE2004/bit torrent/inter.html>

```

Frame 14072: 122 bytes on wire (976 bits), 122 bytes captured (976 bits) on interface 0
Ethernet II, Src: IntelCor_4b:b1:c6 (5c:51:4f:4b:b1:c6), Dst: Sagemcom_21:f0:4e (2c:39:96:21:f0:4e)
Internet Protocol Version 4, Src: 192.168.1.59 (192.168.1.59), Dst: 70.65.184.113 (70.65.184.113)
Transmission Control Protocol, Src Port: 7295 (7295), Dst Port: 6881 (6881), Seq: 540, Ack: 34941, Len: 68
BitTorrent
  Message: Len:13, Request, Piece (Idx:0x4c,Begin:0xc000,Len:0x4000)
    Message Length: 13
    Message Type: Request (6)
    Piece index: 0x0000004c
    Begin offset of piece: 0x0000c000
    Piece Length: 0x00004000
  
```

Ici on voit que la fonction utilisée est bien la fonction Request (ID : 6), que l'index de la pièce à transmettre est 0x0000004c, sur cet index le point de départ du transfert est 0x0000c000 et on souhaite effectuer le transfert d'une pièce de longueur 0x00004000 à partir de ce point.

À la suite de la trame request, on reçoit la trame piece :

```

Frame 14722: 1506 bytes on wire (12048 bits), 1506 bytes captured (12048 bits) on interface 0
Ethernet II, Src: Sagemcom_21:f0:4f (2c:39:96:21:f0:4f), Dst: IntelCor_4b:b1:c6 (5c:51:4f:4b:b1:c6)
Internet Protocol Version 4, Src: 70.65.184.113 (70.65.184.113), Dst: 192.168.1.59 (192.168.1.59)
Transmission Control Protocol, Src Port: 6881 (6881), Dst Port: 7295 (7295), Seq: 74063, Ack: 608, Len: 1452
[12 Reassembled TCP Segments (16397 bytes): #14470(1452), #14473(1452), #14490(1452), #14493(1452), #14495(1452),
BitTorrent
  Message: Len:16393, Piece, Idx:0x4c,Begin:0xc000,Len:0x4000
    Message Length: 16393
    Message Type: Piece (7)
    Piece index: 0x0000004c
    Begin offset of piece: 0x0000c000
    Data in a piece: e1bad3e829a68d2c0daf47cd226cead5369d11d179c5d830...
  
```

On voit ici que la trame piece (ID : 7) que l'on reçoit correspond exactement à celle demandée soit la pièce d'index 0x0000004c, d'offset de départ 0x0000c000. La donnée transmise peut être lue dans la partie « Data in piece ».

En cas d'erreur lors de la réception de la trame piece, on peut être amené à demander une retransmission de la trame. Pour se faire, il n'y a pas de trame prévue pour la retransmission, on utilise simplement une nouvelle fois la fonction request.

Une fois que le transfert est terminé, on remarque que l'on obtient plus de trames de Handshake. En effet, le tracker sait que nous avons terminé notre téléchargement (nous l'avons notifié d'une trame particulière) et nous devenons un seeder capable de fournir le fichier dans son intégralité :

23713 495.70553700	70.65.184.113	192.168.1.59	BitTorrent	134 Handshake
24496 514.35590000	46.239.239.20	192.168.1.59	BitTorrent	134 Handshake
25532 563.72032600	70.65.184.113	192.168.1.59	BitTorrent	134 Handshake
26831 588.42610900	46.239.239.20	192.168.1.59	BitTorrent	134 Handshake
27309 621.93050000	89.135.134.182	192.168.1.59	BitTorrent	122 Handshake
27680 631.68859000	70.65.184.113	192.168.1.59	BitTorrent	134 Handshake
34950 704.89742000	89.135.134.182	192.168.1.59	BitTorrent	122 Handshake

4 - Avenir, danger et dérive : le P2P est-il légal ?

Nous ne pouvons pas parler du peer-to-peer sans parler du problème d'éthique qu'il engendre. En effet, à l'apparition de ce protocole, il s'est développé de nombreuses dérives. La première a été directement liée au point que nous avons majoritairement développé dans ce TIPE : le transfert de fichier décentralisé. L'anonymat de ce type de réseau a amené le développement d'un important trafic de fichiers piratés, dont en majorité des films, musiques, logiciels ou encore des jeux vidéo mis à dispositions sur différentes plateformes illégalement.

De nombreuses lois ont été créées un peu partout dans le monde (comme Hadopi par exemple) pour limiter ces échanges.

Cependant, il existe beaucoup d'autres applications comme par exemple, l'utilisation du P2P afin de créer un réseau décentralisé et anonyme de machines. En effet, il s'est développé un réseau parallèle au Web classique via le logiciel TOR. Ce logiciel une fois installé permet d'accéder au réseau TOR et aux sites avec l'extension « .onion ». Les adresses de ces sites sont souvent de simples séries de lettres et de chiffres. Le logiciel crée une surcouche (un peu comme les différentes couches de l'oignon) qui permet de gagner en anonymat. Notre adresse IP est cachée et toutes nos requêtes passent par plusieurs serveurs avant d'arriver au destinataire afin de brouiller les pistes. Il est donc très difficile de localiser quelqu'un sur ce type de réseau surtout si la personne ajoute d'autres sécurités (comme un VPN⁴⁰ par exemple). Cependant, cette technologie limite énormément les débits.

Cette utilisation du P2P a donné naissance au "deep-web". Le deep-web représente la totalité des sites n'étant pas indexés par Google ou les autres moteurs de recherches classiques (appelé aussi Web non visible ou Web profond). Contrairement à ce que l'on pourrait penser, cette portion du web représente une part très importante d'internet : selon les études, on retrouve des chiffres allant de 75% du web à 90%⁴¹. L'entreprise BrightPlanet a réalisé une enquête en 2001 dans laquelle elle estime que le deep-web contiendrait 500 fois plus de ressources de celles répertoriées par les moteurs de recherche. Une grande part de ces sites non indexés sont d'excellentes ressources : des blogs, des articles en grande quantité et parfois très intéressants, ou encore simplement des sites classiques que les développeurs n'ont pas souhaité indexer afin de les garder confidentiels ou privés et qui sont souvent protégés par un mot de passe. Le problème majeur causé par TOR est qu'il est très difficile de filtrer son contenu. En effet, il est très facile d'entrer sur le réseau (aucune inscription préalable) et d'y publier le contenu de notre choix tout en bénéficiant d'un anonymat total. Ceci a permis pour certaines personnes d'outre passer les lois restrictives concernant la liberté d'expression. Par exemple lors des révolutions arabes TOR a été un lien indispensable entre les militants et le reste du monde.⁴²

Cependant, derrière cet aspect du deep-web, s'est développé un véritable réseau beaucoup plus inquiétant sur lequel l'anonymat amené par TOR a créé une zone qui transgresse de nombreuses lois internationales. On trouve de nombreux sites parmi lesquels certains proposant du trafic d'armes, de drogues, ou encore d'autres dérives notamment sexuelles bannies par le web classique. Un des sites les plus célèbres du deep-web est "Hidden Wiki" (une sorte de Wikipedia caché) qui contient de nombreuses explications sur le deep-web, mais surtout des liens vers des sites, plus ou moins fréquentables, du deep-web.

Ainsi, il est important de toujours soumettre à notre réflexion l'utilisation de tels outils de partage et donner les moyens à un organisme indépendant de la fermeture de ou la mise hors service des sites malveillants sur le deep-web, tout en gardant à l'esprit l'importance de la liberté d'expression et de l'accès à la connaissance ?

⁴⁰ VPN : Virtual Private Network permet de créer un tunnel entre deux ordinateurs (réseau local virtuel). Ceci permet de passer outre les restrictions du réseau (pare-feu / proxy)

⁴¹ <http://spintank.fr/2013/09/on-a-explore-le-deep-web/>

⁴² <http://www.donneesprivees.com/2013/09/19/tor-le-routeur-oignon/>

5 - Conclusion

Ce projet TIPE nous a permis de comprendre les enjeux actuels en matière de transfert de fichier. À l'heure où le nombre d'internautes dans le monde ne cesse de croître et que les échanges ne cessent de se multiplier, il est important de se demander si les avancées technologiques en matière de serveur informatique et les coûts qu'elles représentent arriveront à compenser la montée du nombre d'internautes. Les internautes veulent que leurs fichiers soient disponibles partout et en toutes circonstances, ils veulent également pouvoir visualiser leurs fichiers en même temps que ces derniers se téléchargent. Le peer-to-peer pour le transfert de fichier paraît donc être une alternative viable et est en train de prendre une place significative dans les échanges de fichiers notamment au travers du protocole BitTorrent qui est le meilleur protocole pour le transfert de fichiers volumineux. Beaucoup de sociétés comme Microsoft, EA, Ubisoft, Blizzard et beaucoup d'autres utilisent cette technologie pour déployer les mises à jour ou de nouveaux logiciels chez leurs clients limitant ainsi l'utilisation de leurs serveurs. Mais nous avons aussi été forcés de constater que beaucoup de fichiers disponibles sur les plateformes de téléchargement utilisant le peer-to-peer ou un bon nombre de torrent sur internet étaient des fichiers partagés sans le consentement des ayants droits. C'est ainsi que nombreuses personnes ont beaucoup de difficultés à distinguer la frontière entre peer-to-peer, BitTorrent et illégalité.

Plus largement, le peer to peer a permis de créer un large éventail d'applications allant de la crypto-monnaie au deep-web en passant par le calcul collaboratif (projet SETI@Home pour la recherche de population extraterrestre). Certaines de ces applications comme les crypto-monnaies ont permis de révolutionner les échanges d'argent entre les personnes sur internet mais d'autres comme le deep-web ont créé une zone de liberté sans limite dans laquelle on trouve de tout. L'anonymat et la sécurité que confèrent les échanges en peer-to-peer doivent nous faire réfléchir sur l'éthique et les dérives que cela entraîne et toujours faire une utilisation raisonnée, à la fois pour les utilisateurs et les sociétés, des technologies que l'on emploie.

Le peer-to-peer a cependant de bonnes perspectives d'évolutions et d'améliorations y compris dans le domaine du partage de fichier. En effet, il serait intéressant de mettre en place des systèmes permettant de corriger les erreurs (comme Read-Solomon) et de reconstituer les fichiers corrompus sans demander de retransmission. Ainsi cela permettrait d'optimiser les temps de transfert.