Rapport de stage de fin d'études

Analyse de systèmes de streaming vidéo en Pair-à-Pair sur l'internet

Auteur :
Pierre Vigreux

Tuteur UPMC :
Olivier Fourmaux

25 août 2012

Table des matières

Mes remerciements s'adressent à :

Olivier Fourmaux pour avoir accepté de me prendre en stage ainsi que pour le suivi rigoureux qu'il a effectué tout au long de cette période

Sebastien Tixeuil, responsable de l'équipe NPA, pour ses conseils et son soutien à ma candidature en thèse

Kei Delhomme pour l'aide précieuse apportée lors de la mise en place de la plateforme d'expérimentation, ainsi que les échanges autour de nos thématiques respectives

Mes parents ainsi qu'Estelle Morin pour les heures de relecture qu'ils ont bien voulu accorder à ce rapport pour tenter de le débarrasser de ses nombreuses fautes

Aux enseignants de Polytech Nantes qui sont intervenus sur les thématiques liées au réseau et à la distribution de contenus et qui m'ont apporté de solides bases au cours de mes trois années d'études

Introduction

Désirant poursuivre mes études après le diplôme d'ingénieur vers un cursus de doctoral, j'ai axé mes recherches de stage vers les laboratoires français d'informatique. Je me suis naturellement orienté vers le laboratoire d'informatique Paris 6 en raison de son grand rayonnement notamment dans le domaine du réseau. J'ai également été intéressé par les multiples opportunités qu'il représentait en terme de domaine d'études. Ayant déjà travaillé dans le cadre de la distribution de contenus vidéo au cours de mon dernier stage, j'ai postulé au sein de l'équipe « Networks and Performance Analysis » dont les thématiques incluent les réseaux de distribution de contenu. J'ai ainsi obtenu une proposition de stage de la part d'Olivier Fourmaux, enseignant chercheur à l'université Pierre et Marie Curie, sur l'analyse des systèmes de distribution de contenus en direct en pair-à-pair sur l'internet.

Ce stage s'est déroulé du 1er février au 31 juillet 2012.

Ce rapport présente tout d'abord l'établissement dans lequel j'ai effectué mon stage, puis détaille la planification suivie. Il se concentre par la suite sur une étude bibliographique des techniques de distribution de contenus en pair-à-pair et sur les méthodes d'analyse de ces systèmes. Le chapitre suivant détaille la plateforme d'expérimentation que j'ai mis en place en collaboration avec un autre stagiaire pour effectuer une nouvelle campagne de mesures. Puis je présente dans les grandes lignes les fonctions d'analyse offertes par mon application. Enfin j'expose une partie des résultats obtenus suite à l'analyse des mesures effectuées par une stagiaire précédente, ainsi que celles que j'ai relevées durant mois de juillet.

Chapitre 1

Cadre du stage

1.1 Université Pierre et Marie Curie

L'université Pierre et Marie Curie (UPMC) est une université spécialisée dans les sciences et la médecine. Elle est présente dans quatre régions de France et comprend 32 000 étudiants et 10 000 personnels. J'ai pour ma part été intégré au campus de Jussieu situé dans le 5$^{\text{ème}}$ arrondissement de Paris. Le campus comprend sept unités de formation et de recherche à savoir :

- La faculté de médecine Pierre et Marie Curie ;
- La faculté de chimie ;
- La faculté d'ingénierie ;
- La faculté de mathématiques Pierre et Marie Curie ;
- La faculté de physique Pierre et Marie Curie ;
- La faculté de biologie ;
- L'unité de formation et de recherche « Terre, environnement, biodiversité ».

L'université comprend également une école d'ingénieur rattachée au réseau Polytech, Polytech Paris-UPMC. La recherche à l'UPMC s'articule autour de quatre pôles majeurs :

- Le pôle modélisation et ingénierie ;
- Le pôle énergie matière et univers ;
- Le pôle terre vivante et environnement ;
- Le pôle vie et santé.

1.2 Laboratoire d'Informatique Paris 6

Le laboratoire d'informatique Paris 6 (LIP6) est un laboratoire de recherche sous tutelle de l'UPMC et du CNRS. Il s'agit de l'un des principaux laboratoires d'informatique de France et du plus important de la région parisienne, comprenant 189 chercheurs permanents et 248 doctorants. Le LIP6 couvre un vaste ensemble de domaines regroupés au sein de cinq départements :

- Calcul Scientifique ;
- Décision, Systèmes Intelligents et Recherche Opérationnelle ;
- Données et Apprentissage Artificiel ;
- Réseaux et Systèmes Répartis ;
- Système Embarqué sur Puce.

J'ai travaillé au sein du département réseaux et systèmes répartis et plus précisément au sein de l'équipe « Networks and Performance Analysis » (NPA) sous la tutelle d'Olivier Fourmaux, maître de conférence à l'UPMC. Les thématiques de l'équipe NPA sont le développement d'une vision pour l'internet du futur et la conception de solutions pour le représenter et le contrôler.

1.3 Planification

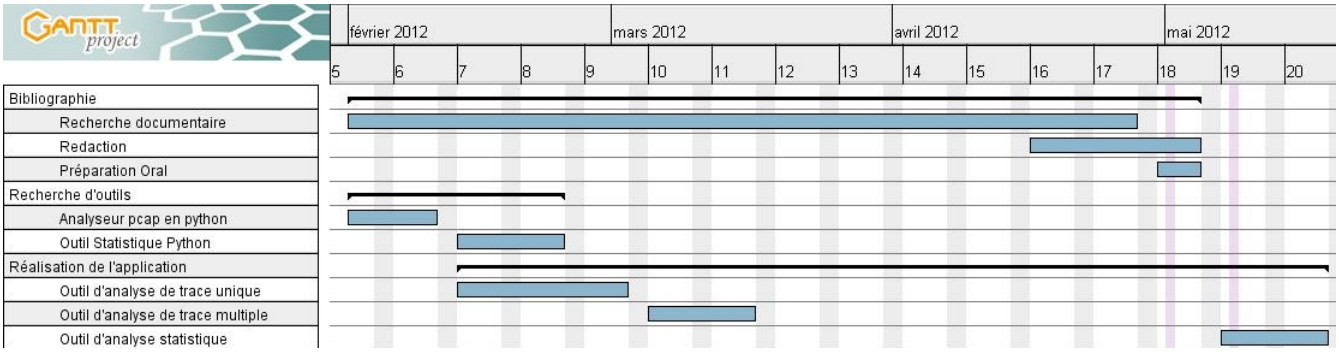


FIGURE 1.1 – Planning effectif de la première partie du stage

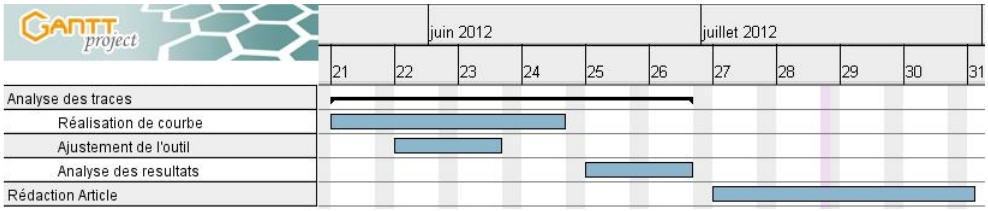


FIGURE 1.2 – Planning prévisionnel de la deuxième partie du stage

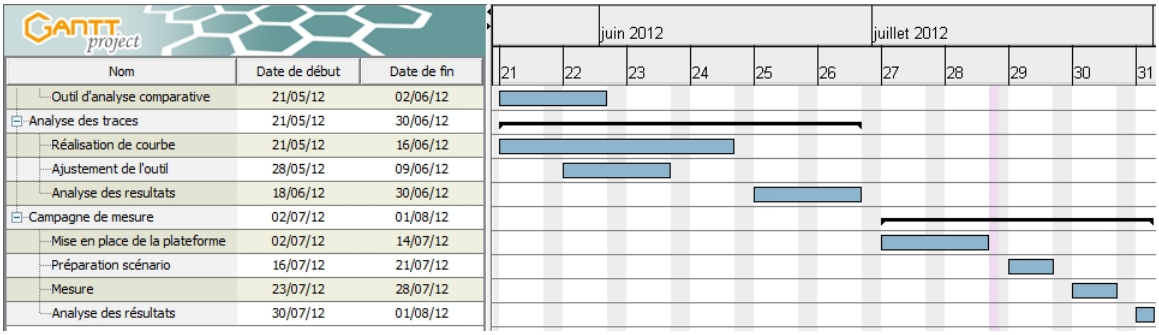


FIGURE 1.3 – Planning effectif de la deuxième partie du stage

Le suivi du projet par mon tuteur s'est fait au moyen de réunions ou d'échanges de mail environ toutes les unes à deux semaines. J'ai effectué toute la première partie de mon stage seul. La figure 1.1 présente le planning suivi au cours de la première partie de mon stage. Le planning prévisionnel de la seconde partie est détaillé dans la figure 1.2. La figure 1.3 représente le planning effectif de la deuxième partie de mon stage, il inclut notamment la réalisation d'une deuxième campagne de mesures à l'occasion des jeux olympiques de Londres.

Au cours de la deuxième partie de mon stage j'ai eu à travailler avec un autre stagiaire qui était également sous la responsabilité de mon tuteur sur une problématique de suivi d'utilisateurs sur des

réseaux de distribution de contenus P2P. Nous avons notamment travaillé ensemble sur la plateforme d'expérimentation mise en place pour les jeux olympiques. Pour une plus grande efficacité nous avons décidé de mettre en place un wiki. Celui-ci détaille notamment la configuration des machines virtuelles, l'architecture, la configuration des accès aux machines, ainsi que les commandes utilisées pour la capture des paquets.

Chapitre 2

Étude bibliographique

Dans ce chapitre nous essayerons de faire une synthèse des différents travaux déjà effectués dans le cadre de l'analyse des systèmes applicatifs pair-à-pair (P2P). Nous détaillerons notamment les différents types de services disponibles, les approches d'études utilisées, et nous nous attarderons spécialement sur les travaux relatifs aux applications de type diffusion de contenus vidéo en direct.

2.1 Les applicatifs P2P

Les applications P2P couvrent un large spectre d'activités. Si initialement il s'agissait de logiciels dédiés aux transferts de fichiers tels que BitTorrent[3] ou eMule[4], par la suite de nombreux autres services dédiés à la connexion de terminaux distants ont été séduits par les protocoles P2P. On peut ainsi parler de la téléphonie IP avec Skype[5], mais également, et ce sera le sujet principal de notre étude, des outils de distribution de contenus vidéos qu'il s'agisse de vidéo à la demande avec Joost[6], ou de chaîne de télévision en direct avec SopCast[7], PPLive[8], TV Ants[9]. Ainsi de nombreux articles posent la question de savoir si le P2P finira par proposer la « killer application ».

2.2 Axes d'études

Il existe deux axes principaux pour étudier une application. Nous pouvons faire une étude de rétro-ingénierie dans laquelle nous chercherons à comprendre et potentiellement reproduire le fonctionnement interne du sujet d'étude ou alors nous pouvons tenter une approche de type boîte noire. On ne cherche pas dans ce cas à comprendre les mécanismes mis en œuvre, mais plutôt à observer les conséquences sur l'environnement.

2.2.1 Rétro-ingénierie

Nous voulons ici décomposer le fonctionnement d'une ou plusieurs applications. Ainsi on tente de décompiler le programme et de reproduire de la façon la plus fidèle possible le fonctionnement du logiciel. Ce type d'étude peut être long et fastidieux et présente l'inconvénient de n'être valide que pour une période limitée. En effet une mise à jour pourrait changer le fonctionnement du logiciel rendant l'étude caduque. Par ailleurs la rétro-ingénierie ne peut être appliquée qu'à un panel de logiciels préalablement définis. Cependant elle peut aussi mettre en évidence certains systèmes qui peuvent alors être précisément analysés. On peut ainsi décortiquer les trames émises et comprendre le mécanisme d'encodage et de sélection des pairs. Cette méthode est employée dans divers travaux[10].

2.2.2 Analyse boîte noire

Ici l'idée n'est pas de mettre en avant les principes déployés, mais plutôt d'analyser les impacts sur un système complet. Ainsi on se place à un niveau d'abstraction élevé dans lequel on effectue des mesures à l'aide d'outils tels que tcpdump. Par la suite ces traces sont analysées via des outils

spécifiques. De cette façon on peut travailler sur un plus large champ de logiciels. Ainsi on peut citer les travaux de D. Rossi, E. Sottile et P. Veglia qui via le framework Sherlock disposent d'un outil permettant l'analyse de nombreuses applications P2P[2].

2.3 La diffusion de contenus en temps réel

Nous présenterons ici les contraintes liées à la diffusion de contenus en temps réel et les différences avec un service de distribution de fichiers classiques en P2P. Puis nous exposerons une approche générale sur la conception des applications de lectures de flux P2P.

2.3.1 Les contraintes

L'un des principaux problèmes de la diffusion de vidéo est que, contrairement à la distribution de fichiers, on ne connaît pas la taille du contenu en cours de partage. Ainsi dans le cas d'un client BitTorent on sait quel espace sera occupé in fine par les fichiers qui sont en cours de téléchargement. Tandis que lors du visionnage d'une chaîne de télévision le fichier vidéo est virtuellement infini, puisqu'il ne prendra fin que lorsque l'utilisateur fermera le programme. Par ailleurs un système de streaming en temps réel est par définition très sensible au temps. En effet il faut absolument que les données arrivent avant leur lecture. Dans le cas contraire, le lecteur serais face à un cas de famine. Une autre contrainte qui découle de l'absence de fin du fichier vidéo est le fait que la vidéo n'est pas stockée sur le disque. À la place on dispose d'une zone de mémoire tampon qui stockera de manière temporaire les données. Cependant cet espace est limité. Ce fonctionnement apporte deux problèmes :

- On ne peut pas stocker des données n'étant pas utiles à court terme ;
- On ne peut également pas stocker trop longtemps des données déjà visionnées, même si elles peuvent éventuellement être utiles à d'autres pairs.

2.3.2 Conceptions générales

Ici nous nous attacherons à décrire un modèle général. Il peut exister des différences entre les implémentations faites au sein des divers logiciels existants, mais le principe général restera le même. Nous verrons tout d'abord la vision du buffer telle que présentée par H. Liu et G. Riley[1]. Puis nous présenterons les architectures possibles et résumerons la comparaison faite par N. Magharei et R. Rejaie[11].

Le Buffer

Tout d'abord nous considérons que la vidéo qui sera diffusée est découpée en segments de tailles fixes. Le buffer disposera de deux pointeurs ; l'un sera dédié à la lecture du contenu et l'autre servira à l'écriture des segments téléchargés. L'objectif est bien évidemment qu'à aucun moment le pointeur de lecture ne rattrape celui d'écriture ce qui reviendrait à un cas de famine.

Chaque pair s'inscrit dans un ensemble que nous nommerons son voisinage avec qui il échangera les segments qu'il possède et ceux qu'il souhaite obtenir. Pour ce faire on peut utiliser la vision conceptuelle du buffer étendu (fig 2.1) qui permet au pair de savoir à l'avance combien de segments il peut demander. De plus chaque pair tient en permanence un index des segments dont il dispose pour répondre au plus vite aux requêtes provenant de son voisinage.

Architecture Réseau

Dans la mise en place de réseaux de streaming en P2P, on distingue principalement deux architectures :

- Le réseau maillé : les liaisons sont formées à l'aide d'algorithmes semi-aléatoires,
- Le réseau basé sur des arbres : le système d'architecture est relativement monolithique et dispose d'une hiérarchisation forte.

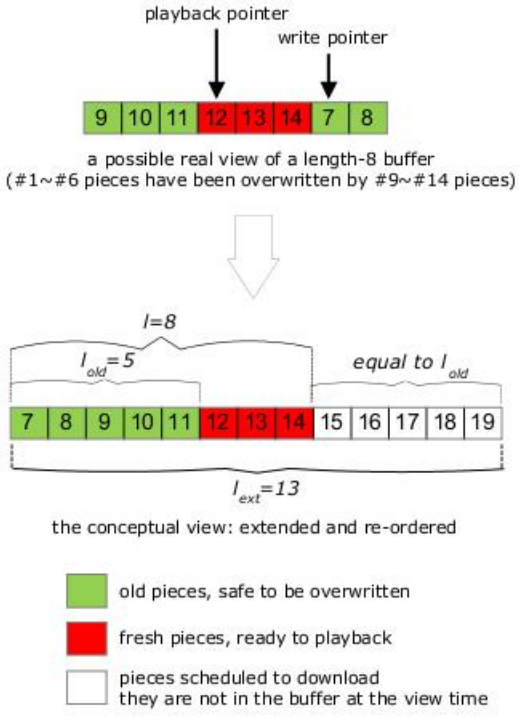


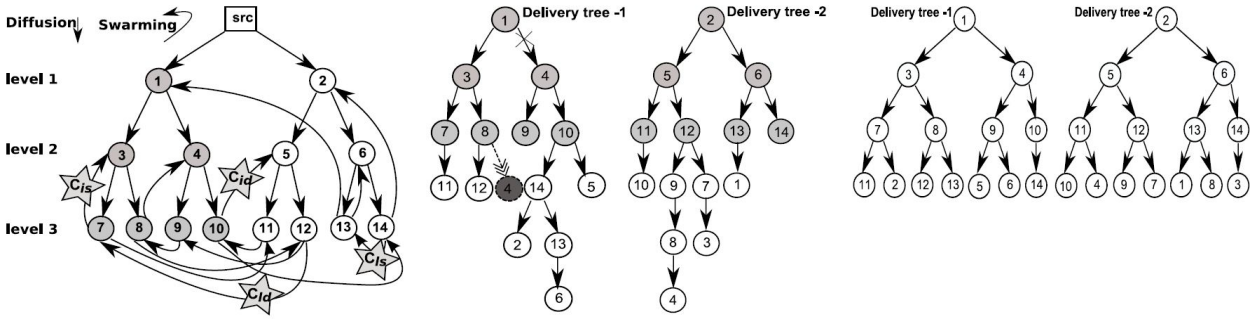
FIGURE 2.1 – Vue Conceptuelle d'un buffer[1]

Dans leur article[11] N. Magharei et R. Rejaie font un comparatif de ces deux architectures que nous allons résumer ici.

Tout d'abord nous allons étudier les similarités entre les deux méthodes. En effet, dans un cas comme dans l'autre on veut favoriser l'échange et la participation la plus importante possible de tous les pairs. Ainsi chaque pair peut recevoir des données de plusieurs parents et envoyer des données à plusieurs enfants. Par la suite on peut former dans chaque cas un chemin emprunté par les paquets pour aller de la source (le serveur d'origine) à chaque nœud. L'ensemble de ces chemins est appelé arbre de livraison « delivery tree ». Enfin dans les deux cas, l'objectif est de maintenir un délai de synchronisation permettant l'acheminement des paquets avant l'apparition d'une famine. On peut ainsi définir le temps de synchronisation sans perte τ , qui influera sur le processus de mise en mémoire tampon. Ce temps sera lui même influencé par le nombre maximum de sauts pour relier un pair à la source, la diversité des chemins possibles et la possible arrivée désordonnée des paquets.

La principale différence entre les deux approches est la manière dont chaque sorte de paquet est acheminé. En effet dans le cas d'une approche basée sur des arbres, on construit un arbre par type de paquet ; de cette façon si un pair ne dispose pas de la bande passante suffisante pour acheminer ces paquets alors tous ses fils sont impactés. Dans le cas du réseau maillé, l'arbre d'acheminement lié à un type de paquet peut se reconstruire dynamiquement ce qui permet de s'affranchir d'un pair ne disposant pas des capacités de transmission suffisantes.

La figure 2.2 présente les deux architectures : le réseau maillé (figure 2.2(a) et (b)) et l'architecture basée sur des arbres (figure 2.2(c)). Les deux représentations, organisées sous la forme d'un arbre, montrent un réseau unidirectionnel pour lequel chaque lien constitue une relation parent-enfant. Ces graphiques permettent de définir pour chaque pair un niveau qui correspond au nombre de sauts



(a) Organized view of a random mesh (b) Delivery trees in Mesh-based (c) Delivery trees in Tree-based

FIGURE 2.2 – Exemple d'architecture, avec deux arbres de diffusion[11]

minimums nécessaires pour aller de la source au pair. Ainsi avec une représentation sous forme d'arbre le niveau du pair correspond à la profondeur du nœud dans l'arbre.

Nous allons maintenant détailler le modèle utilisé pour acheminer un paquet. Il se décompose en deux phases pour maximiser l'utilisation de la bande passante sortante[12].

1) La phase de diffusion : dès qu'un nouveau paquet devient disponible à la source, il est envoyé à un pair de niveau 1 (p) durant l'intervalle δ. Puis les pairs enfants de p de niveau 2 récupèrent le paquet durant l'intervalle suivant. Durant cette phase tous les liens entre les pairs de niveau i et ceux de niveau i+1 (où $i <$ profondeur de l'arbre) sont utilisés. Ces liens sont appelés « diffusion connection » et sont représentés à l'aide de flèches rectilignes sur la Figure2.2(a). De cette manière chaque paquet est délivré une seule fois à tous les pairs. Le chemin de diffusion est appelé sous-arbre de diffusion (« diffusion subtree »), et il est composé d'un pair de niveau 1 (la racine) et de tous ses fils de niveaux inférieur. Dans la Figure 2.2(a), un sous-arbre de diffusion est représenté par les pairs assombris (la racine est 1). Ainsi l'architecture dispose d'un sous-arbre de diffusion par pair de niveau 1. À la fin de la phase de diffusion, seul un sous-ensemble de pairs (*i.e* un sous-arbre de diffusion) dispose du nouveau paquet.

2) La phase de « swarming » : au cours de cette phase, les pairs vont échanger leurs nouveaux paquets entre les sous-arbres de diffusion. Dans cette phase toutes les connections entre les niveaux i et j sont utilisées ($j \leq i$), elles sont appelées « swarming connections ». Elles sont représentées à l'aide de flèches courbées sur la Figure 2.2(a).

Les « swarming connections » peuvent se classer en quatre catégories en fonction des pairs qui sont ainsi connectés :

C_{ld} qui joint deux feuilles appartenant à deux sous-arbres ;
C_{ls} qui joint deux feuilles appartenant au même sous-arbre ;
C_{id} qui joint une feuille et un pair intermédiaire appartenant à deux sous-arbres ;
C_{is} qui joint une feuille et un pair intermédiaire appartenant au même sous-arbre.

Un exemple de chaque connexion est marqué par une étoile sur la figure 2.2(a). Les Figures2.2(b) et 2.2(c) présentent les arbres d'acheminement dans le cas d'un réseau maillé et d'un réseau en arbre. Ces figures montrent bien les risques auxquels sont soumis les réseaux en arbre en cas de baisse de la bande passante. En effet la Figure 2.2(b) présente le cas d'une chute de débit entre 1 et 4. Dans ce cas le pair 4 peut, grâce à la phase de « swarming », passer par le pair 8 pour recevoir le paquet.

2.4 Les métriques d'analyse de réseau P2P

Dans cette partie nous présenterons les différentes métriques utilisées pour évaluer les performances d'un réseau de diffusion de contenus en direct en P2P.

2.4.1 Nombre de pairs dans le voisinage

L'une des métriques les plus évidentes est le nombre de pairs qu'un client peut compter dans son voisinage. Ainsi on peut évaluer le nombre de pairs participant sur un intervalle de temps donné ou faire une mesure cumulative des pairs qui donnera le nombre de clients avec lesquels ont eu lieu les échanges. La donnée cumulative va notamment être fortement dépendante du profil de l'utilisateur. Ainsi si celui-ci passe son temps à changer de chaîne, à chaque changement il devra contacter de nouveaux utilisateurs visionnant eux aussi cette chaîne. Dans un cas de visionnage stable (rester sur la même chaîne) l'utilisateur contactera un groupe de pairs et la courbe cumulative tendra vers un maximum qui sera le nombre maximum de voisins.

2.4.2 Quantité de données échangées

Une autre métrique triviale est la quantité de données échangées entre un client et les pairs de son voisinage. De cette manière on peut évaluer son implication au sein du réseau. Ainsi si le pair n'envoie que très peu de données, cela peut être dû soit à une asymétrie du débit ascendant, soit à un manque de synchronisation avec la majorité des pairs. En effet si le pair est en retard de plusieurs segments par rapport à la source il ne possède alors que des segments pouvant intéresser les pairs ayant le même retard ou un retard encore plus grand. À contrario si le pair suit au plus près la diffusion des nouveaux segments par la source il sera énormément sollicité par ses voisins.

2.4.3 Efficacité du réseau[1]

Dans leur article H. Liu et G. Riley proposent une formule permettant d'estimer les performances d'un système de diffusion de contenus temps réel en P2P. Leur étude se base sur un article[13] étudiant les performances d'un système de transfert de fichiers P2P type BitTorrent. L'objectif est de fournir une évaluation de η qui est la probabilité qu'un pair ait au moins une pièce à envoyer à ses voisins. On obtient alors la probabilité suivante :

$$\eta = 1 - \sum_{n_i=0}^{N-1} \frac{1}{N} \left(\frac{N - n_i}{N(n_i + 1)} \right)^k$$

N est le nombre de pièces du fichier
k le nombre de voisins de chaque pair
n_i le nombre de pièces déjà téléchargées par le i^{eme} pair

Cependant en raison du système de buffer mis en place pour le streaming on ne peut pas appliquer cette formule directement. En effet dans le cas du partage de fichier, le nombre de pièces total est fixe pour chaque fichier, tandis que dans le cadre de la diffusion vidéo il s'agit d'une valeur fluctuante. Nous utiliserons à la place la diversité des pièces, c'est à dire le nombre de pièces disponibles à un moment donné. Nous obtenons alors la formule suivante :

$$\eta = 1 - \left(1 - \frac{2\alpha - 1}{2\alpha^2} . \frac{l_{ext} - 1}{l_{ext}} \right)$$

Avec

$$\alpha = \frac{N - l + 1}{l_{ext}}$$

N le nombre de pièces disponibles
l_{ext} la taille du buffer étendue, cf Figure 2.1
l la taille du buffer

Ainsi pour obtenir les meilleurs performances (maximiser la valeur de η) il faut minimiser α. Nous pouvons pour cela minimiser la diversité des pièces (N), en forçant les pairs à se synchroniser au mieux avec la source.

Dans la pratique, l'estimation de l'efficacité du réseau est incompatible avec une analyse de type boîte noire car elle nécessite de connaître des paramètres internes à l'application (i.e taille du buffer, diversité des pièces) qui ne peuvent être récupérés qu'à l'aide d'une démarche de rétro-ingénierie.

2.4.4 Métrique utilisée par Sherlock[2]

Sherlock, pour « Sketch Hallmark Elements to Recap and Look-into Overlays with Charts of Kiviat » est un outil d'analyse permettant la mesure de divers paramètres pour des applications P2P à l'aide d'une approche boîte noire. De part son approche il apparaît comme intéressant et proche des objectifs de notre projet bien qu'il n'ait pas été développé prioritairement pour les logiciels de diffusion de contenus en P2P, mais pour tous types de système P2P. Nous allons ici présenter les métriques utilisées qui pourraient présenter le plus grand intérêt pour notre étude.

Symétrie des échanges

Il existe deux façons d'évaluer la symétrie des échanges. Nous pouvons effectuer les mesures soit en terme de nombre de paquets échangés, soit en terme de quantité de données échangées. Ici contrairement à la section 2.4.2 on ne mesure pas la quantité de données globales échangées avec le voisinage, mais bien le rapport entre les données reçues et envoyées à chaque pair. Nous notons $P(X,Y)$ (respectivement $B(X,Y)$) le nombre de paquets (respectivement la quantité de données) circulant de X vers Y, et $P(Y,X)$ ($B(Y,X)$) les quantités circulant dans l'autre sens. Nous pouvons alors définir la symétrie pour les paquets Sym_P et celle pour les données Sym_B comme suit :

$$Sym_P = \frac{P(X,Y)}{P(X,Y) + P(Y,X)}$$

$$Sym_B = \frac{B(X,Y)}{B(X,Y) + B(Y,X)}$$

Ces variables devraient être égales à 0.5 si les flux étaient parfaitement symétriques en terme de paquets ou de données. Par opposition les valeurs tendront vers 1 (ou 0) si tout le trafic est émis (ou reçu) par le pair X.

Mouvement des pairs

Nous avons vu dans la section 2.4.1 que l'on pouvait mesurer le nombre de pairs actifs dans le voisinage sur un intervalle ΔT ainsi que le nombre de pairs ayant fait parti du voisinage de manière cumulative. Sherlock cherche à quantifier les mouvements des pairs dans le voisinage ; pour ce faire il utilise le nombre de pairs présents à un instant donné, mais il évalue aussi les nouveaux pairs et les pairs restant dans le voisinage entre l'instant ΔT et l'instant $\Delta T - 1$. On notera $\mathcal{P}_k$ l'ensemble des pairs avec lesquels X échange durant le k$^{\text{ème}}$ intervalle de temps, c'est à dire durant l'intervalle $[(k-1)\Delta T, k\Delta T]$. De la même façon on notera $\mathcal{N}_k$ l'ensemble des pairs découverts entre le moment 0 et le moment $k\Delta T$. Nous pouvons alors noter :

$$\mathcal{P}_k = \{p : P(X,p) + P(p,X) > 0\}$$

$$\mathcal{N}_k = \cup_{i=1}^{k} \mathcal{P}_i$$

On peut alors définir les variables $P_{\Delta T}$ pour le nombre de pairs instantanés, P_{new} pour le nombre de pairs découverts sur le dernier intervalle de temps, et P_{same} le nombre de pairs intervenant sur l'intervalle et étant déjà intervenu. Nous pouvons alors noter :

$$P_{\Delta T} = card(\mathcal{P}_k)$$

$$P_{new} = card(\mathcal{P}_k \setminus \mathcal{N}_{k-1})$$

$$P_{same} = card(\mathcal{P}_k \cap \mathcal{P}_{k-1})$$

Ces trois paramètres peuvent être observés soit simplement avec leur moyenne et déviation, soit de manière continue dans le temps. Ainsi on peut s'attendre à un grand nombre de découvertes de pairs lors du lancement de l'application qui tendra ensuite progressivement vers 0, tandis que le nombre de pairs ré-intervenants va, lui, progressivement augmenter jusqu'à ce que le pair client ait construit un voisinage optimal.

2.5 Cadre de l'étude

Dans cette partie nous présenterons les objectifs de l'actuel projet, ainsi que le cadre dans lequel il va s'effectuer.

L'objectif du projet est la réalisation d'un outil d'analyse de captures réseaux pcap relevées au cours du fonctionnement d'un logiciel client de télévision en pair-à-pair. Les logiciels sont notamment Sopcast, PPLive, PPStream, UUSee. Ces captures ont été réalisées au cours des phases précédentes du projet par d'autres stagiaires. Elles comprennent divers comportements dont des phases de lecture de 200 secondes d'une même chaîne, ou des changements de chaîne toutes les 25 secondes terminés par une phase de lecture prolongée ; ces changements ont fait parcourir entre deux et cinq chaînes. Ces scénarios ont été reproduits plusieurs fois. L'idée est d'analyser ces traces suivant divers critères (voir section 2.4) et d'essayer de construire des modèles permettant de reconnaître les états par lesquels le client passe (état transitive, état stable).

Ces études seront utilisées pour établir des corrélations entre un comportement utilisateur et un comportement du réseau. Dans le cadre du projet nous définissons un comportement utilisateur et observons la réaction du réseau. La finalité est, à terme, de pouvoir uniquement observer le réseau et en déduire des comportements utilisateurs. Ces analyses permettront de fournir des clés aux opérateurs de transit et aux sociétés de distribution de contenus pour mieux maîtriser le trafic circulant sur leurs infrastructures.

L'application qui sera ainsi développée utilisera donc une approche boîte noire et se voudra généralisable à toutes applications clientes de télévision en pair-à-pair. L'une des évolutions possibles serait l'ajout d'une fonction pour effectuer des mesures en direct, c'est à dire d'analyser les trames au cours du fonctionnement du logiciel client.

Chapitre 3

Expérience

Au cours de mon stage j'ai été amené à mettre en place une nouvelle plateforme d'expérimentation. L'idée était de pouvoir disposer d'un panel de résultats plus important. De plus nous souhaitions profiter d'un évènement particulier qui était l'ouverture des jeux olympiques de Londres pour effectuer des mesures au cours d'une période disposant d'un grand nombre d'utilisateurs simultanés pour faire des comparaisons avec un jour « classique ». Dans un premier temps je décrirai l'architecture de la plateforme d'expérimentation puis je détaillerai les scénarios réalisés ainsi que les méthodes employées.

3.1 Description de la plateforme

Pour les expérimentations nous disposions de quatre machines de type GPU [1] ainsi qu'un boîtier à lames Dell comportant une lame PowerEdge 1850 et sept lames PowerEdge 1950. Les GPU dispose d'une connexion directe à internet tandis que seule la première lame (PowerEdge 1850) est connectée directement à internet ; les autres sont en NAT derrière. La figure 3.1 représente l'architecture de la plateforme.

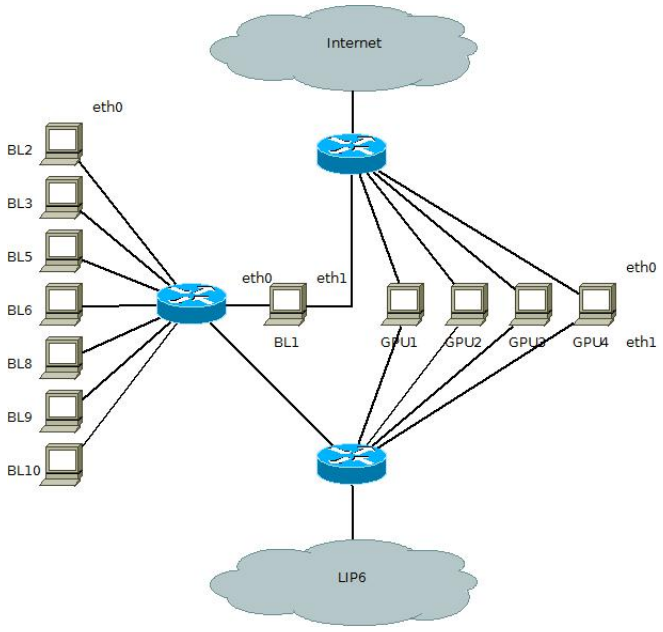


FIGURE 3.1 – Architecture de la plateforme d'expérimentation

1. Graphic Process Unit

	eth0	eth1
GPU1	132.227.62.226/28	10.0.0.11/24
GPU2	132.227.62.227/28	10.0.0.12/24
GPU3	132.227.62.228/28	10.0.0.13/24
GPU4	132.227.62.229/28	10.0.0.14/24

	eth0	eth1
BL1	10.0.0.1/24	132.227.62.230/28
BL2	10.0.0.2/24	
BL3	10.0.0.3/24	
BL5	10.0.0.5/24	
BL6	10.0.0.6/24	
BL8	10.0.0.8/24	
BL9	10.0.0.9/24	
BL10	10.0.0.10/24	

TABLE 3.1 – Plan d'adressage

Ainsi dans cette architecture nous disposions également d'un sous réseau privé utilisé uniquement pour l'administration, le plan d'adressage est résumé dans les tableaux 3.1. Durant cette campagne de mesures nous souhaitions analyser en priorité l'application PPLive en raison de sa position dominante sur le marché. Cette application ne disposant que d'une version Windows, nous avons mis en place un système de virtualisation basé sur le module noyau kvm. Nous avons ainsi pu installer des machines virtuelles sur chacun des GPU et une à deux machines virtuelles sur les lames en fonction de leur puissance. L'accès au réseau pour les machines virtuelles se faisait via une translation d'adresse et des interfaces réseaux TUN/TAP.

Difficulté rencontrée La plus grande difficulté dans la mise en place de la plateforme d'expérimentation fut la prise en main du boîtier à lames Dell. Il s'agissait d'un système qui avait été donné au laboratoire. Le boîtier disposait donc d'une pré-configuration inconnue et souffrait d'un manque complet de documentations. Nous avons donc passé de nombreux jours pour pouvoir le remettre en conformité avec le réseau du lip6 et pour y installer les systèmes d'exploitations.

3.2 Scénarios

Durant la précédente campagne de mesures, de nombreux scénarios avaient été mis en place alternant entre visionnage d'une unique chaîne et zapping sur deux ou cinq canaux. Pour ma part je cherchais à recueillir un maximum de captures pour permettre une analyse statistique la plus pertinente possible. Par ailleurs mon stage se concentrant en priorité sur les phases de transition j'ai donc opté pour des scénarios parcourant cinq chaînes.

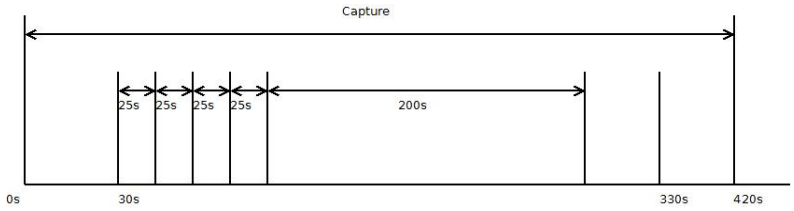


FIGURE 3.2 – Déroulement temporel du scénario

Pour la nouvelle campagne de mesures j'ai réalisé trois scénarios, dans chaque scénario on parcourt les cinq chaînes les plus visionnées à cet instant, le scénario détaillé dans la figure 3.2, se décompose comme suit :

- Ouverture de l'application sans visionner de chaîne pendant 30 secondes,
- Visionnage successif de quatre chaînes pendant 25 secondes sur chacune,
- Visionnage d'une cinquième chaîne pendant 200 secondes,
- Arrêt de la lecture pendant 30 secondes,

– Arrêt de l'application pendant 30 secondes.

La durée de 25 secondes choisie comme durée de visionnage des chaînes transitoires a été estimée comme relativement proche d'un comportement humain permettant ainsi à l'utilisateur d'évaluer l'intérêt du programme. La différence entre les scénarios est la façon dont sont parcourues les chaînes, ainsi je les ai visionnées soit de manière descendante c'est à dire de la plus populaire à la moins visionnée, soit de manière ascendante, à savoir en sens inverse. Enfin j'ai aussi fait un scénario permettant un parcours cyclique, l'idée étant de repasser par des chaînes déjà visionnées précédemment. Ainsi on peut observer si lors du retour sur une chaîne, l'application nous réintroduit dans le nuage précédent ou nous introduit dans un nuage aléatoire et donc potentiellement nouveau.

L'un des objectifs était aussi l'automatisation des expériences. Cependant la manipulation de l'application ne pouvant pas se faire en lignes de commande, il a donc fallu utiliser une solution de script pour interface utilisateur. Je me suis tourné vers Sikuli Script[14]. Il s'agit d'un système de script permettant de contrôler le clavier et la souris à l'aide de captures d'écran. Le système repose sur un interpréteur Python pour Java. Ainsi la syntaxe utilisée pour la création d'un script est celle de Python de même que l'on peut importer n'importe quelles bibliothèques Python. Cependant l'exécution et notamment le système de reconnaissance de l'affichage s'effectue en Java. Le script comprend également un système de génération de log pour faire état de toutes erreurs survenues au cours de l'exécution et notamment si une erreur survient au moment du changement de chaîne. L'image 3.3 représente un exemple de script réalisé avec Sikuli, ce script se contente d'ouvrir PPLive puis dans lancer l'une des chaines diffusé en direct pour une durée de 25 secondes, durant ce temps le script va également relever le nombre de phénomène de mise en mémoire tampon qui va survenir.

FIGURE 3.3 – Exemple de script Sikuli

Cette méthode présente néanmoins des limites. Ainsi, la session devait être ouverte pour que le système fonctionne. L'accès aux machines virtuelles pouvait se faire via RDP [2], cependant la fermeture de la session RDP entraînait aussi la fermeture de la session Windows. J'ai donc utilisé le système de contrôle à distance Log Me In[15] qui permet de conserver la session ouverte et donc à Sikuli de s'exécuter.

2. Remote Desktop Protocol

L'utilisation d'interfaces virtuelles permet d'effectuer la capture directement sur cette interface, l'enregistrement s'effectue avec tcpdump et filtre notamment les paquets rdp et ssh. La commande utilisée 3.1 capture les paquets de l'interface tap0 pour les enregistrer dans un fichier en précisant la date et le début de l'enregistrement. Un nouveau fichier est recréé toutes les 420 secondes, permettant la synchronisation avec le scénario qui redémarre aussi toutes les 420 secondes.

```
tcpdump -ni tap0 -w recordTap0%F-%T -G 420 port not 13389 and port not 22
```

Listing 3.1 – Commande tcpdump

Les mesures ont été effectuées durant la première heure de la cérémonie d'ouverture des jeux olympiques le vendredi 27 juillet 2012 entre 22h et 23h. D'après les programmes de télévision que nous avons pu consulter, la chaîne BeijingTv diffusait successivement les cérémonies d'ouverture des jeux olympiques de Pékin et de Londres. De plus des enregistrements ont aussi été effectués la veille et quelques jours plus tard pour servir de base comparative.

Difficulté rencontrée Ici la principale difficulté à été la prise en main du logiciel PPLive. En effet celui-ci ne dispose que d'une version en chinois et la plupart des informations que nous avons pu trouver en anglais sur internet faisaient référence à une ancienne version du logiciel bilingue. Nous avons donc eu à progresser par essais et je tiens à remercier une stagiaire qui était présente dans la même salle que nous et qui nous a traduit partiellement l'interface et a ainsi pu confirmer nos suppositions.

Chapitre 4

Application

Dans cette section je présenterai les fonctionnalités d'analyse offertes par mon application. Je m'attarderai tout d'abord sur les technologies utilisées au sein de l'application, puis sur les critères de filtrages applicables, par la suite je détaillerai les métriques retenues et implémentées. Enfin je présenterai les différents types d'analyses possibles.

4.1 Technologies utilisées

Suite à une demande de mon tuteur, l'ensemble de l'application a été développée en Python. Mon tuteur étant particulièrement familier avec ce langage, il a ainsi pu me guider sur ses fonctionnalités, notamment pour l'analyse de traces pcap. Je vais maintenant détailler les bibliothèques principales présentes dans l'application ainsi que les points dans lesquelles elles interviennent.

4.1.1 Analyse de paquets

Il existe plusieurs solutions pour l'analyse de paquets en Python, chacune présentant des avantages et des inconvénients. Je me suis documenté sur quelques unes avant de me tourner vers dpkt[16]. Cette bibliothèque présente l'avantage d'être facilement compilable, à jour sur les derniers protocoles et de ne pas nécessiter de privilège utilisateur particulier. Cependant elle souffre d'une absence complète de documentation.

L'intérêt de cette solution est qu'elle permet une désencapsulation progressive des paquets et permet également la reconstruction d'une structure de données de capture. Je l'ai utilisée dès le début pour la partie de filtrage à l'aide de divers critères. Je retiens les paquets pertinents et supprime les paquets inutiles de la capture. Je reconstruis ensuite une nouvelle structure de données qui représente ma capture filtrée qui est ensuite passée en paramètre pour les diverses analyses.

4.1.2 Analyse statistique

Pour la réalisation des graphes, la précédente stagiaire avait travaillé avec gnuplot, mais je souhaitais combiner la partie dessin de graphes avec l'analyse statistique. Au cours de la première année à l'école nous avions eu un cours de statistique au cours duquel nous avions utilisé le logiciel d'analyse statistique R. J'ai donc utilisé la bibliothèque rpy2[17] qui propose une interface de bas niveau avec R et permet également l'utilisation des fonctions, structures de données et fonctions graphiques de R. Rpy2 intervient donc dans toutes les parties dédiées au dessin de graphes avec représentations des écarts types.

4.1.3 Interface graphique

Bien que l'application ne soit pas destinée à une utilisation grand public, j'ai décidé d'inclure une interface graphique basique pour faciliter la prise en main. Pour cette partie j'ai utilisé la bibliothèque Tkinter[18] qui est une forme de standard pour la réalisation d'interfaces graphiques en Python. Elle intervient pour toutes les interactions avec l'utilisateur telles que les choix d'analyses à effectuer, les métriques retenues, ou le chargement des fichiers. Un aperçu de l'interface graphique est présenté dans la figure 4.1.

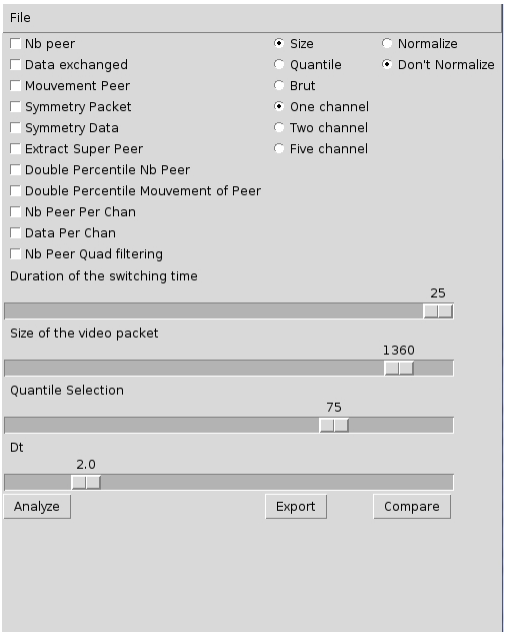


FIGURE 4.1 – Aperçu de l'interface graphique

4.2 Filtrage des traces

L'application peut effectuer trois types de filtrage. Dans chaque cas le principe est d'appliquer un critère qui permettra de déterminer si un pair peut être considéré comme actif au sein des échanges avec notre client. Cependant il existe des critères de filtrage communs. Ainsi nous ne retiendrons que les paquets utilisant les protocoles TCP [1] et UDP [2] car il s'agit des protocoles utilisés par PPLive. Par la suite on peut fournir une liste de ports que l'on souhaite filtrer tels que 22 (ssh), 80 (http), 443 (ssl), 3389 (RDP). Et enfin on ne retient que les paquets ayant soit comme source, soit comme destination notre adresse IP.

Par la suite il faut différencier les trois critères applicables pour identifier un pair ayant une activité significative au sein de notre voisinage :

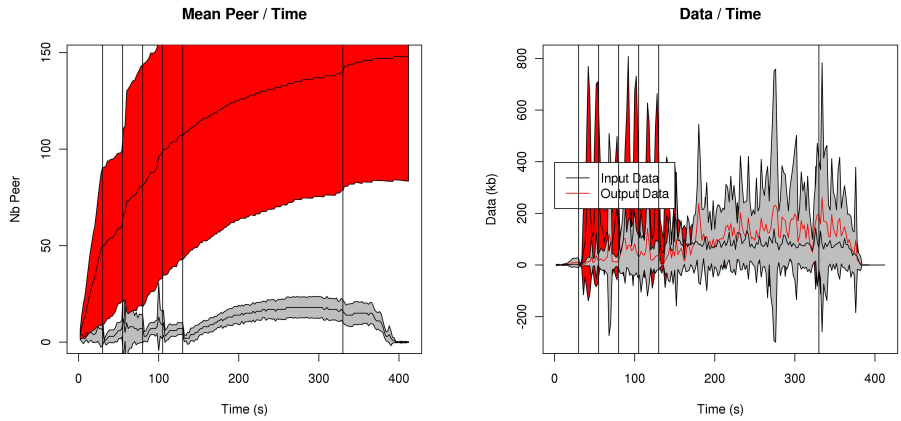
- Critère Brut : on ne définit pas de réels critères de filtrage. On considère que tous les pairs appartiennent au voisinage. Ce critère permet d'évaluer le nombre total de pairs qui ont été contactés lors d'une session en incluant également les pairs qui n'ont échangé que du trafic de signalisation.
- Critère de taille : on définit une taille de paquet que l'on juge comme étant une taille correspondant à un paquet vidéo et non à un simple paquet de signalisation. Avec ce critère on considère

1. Transmission Control Protocol
2. User Datagram Protocol

qu'un pair appartient au voisinage s'il a envoyé ou reçu au moins un paquet de taille supérieure ou égale. Le réglage de la taille du paquet se fait à l'aide d'un curseur réglé par défaut pour une taille de 1360 bytes.

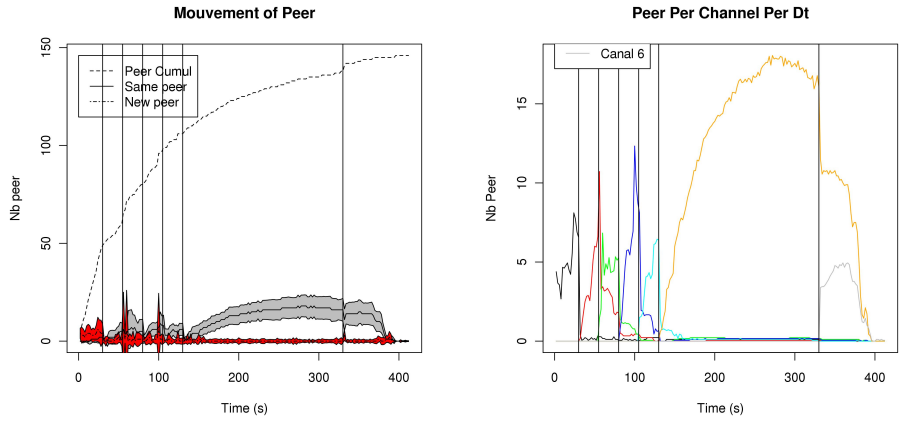
– Critère de percentile : on définit un certain pourcentage de paquets que l'on souhaite retenir. Ce critère fonctionne conjointement avec le choix d'un scénario et risque de présenter des résultats incorrects si le scénario choisi ne correspond pas à celui de la capture. Avec ce critère on découpe la capture en fonction des étapes traversées (visionnage de chaque chaîne). Pour chaque étape on classe les pairs du voisinage en fonction de la quantité de données qu'ils ont échangées. Puis on retient les pairs ayant participé à X % du trafic. Le choix du pourcentage se fait au moyen d'un curseur, tandis que le scénario est choisi à l'aide d'un bouton d'option.

4.3 Métriques d'analyse



(a) Exemple de graphe du nombre de pairs instantanés

(b) Exemple de graphe de quantité de données échangées

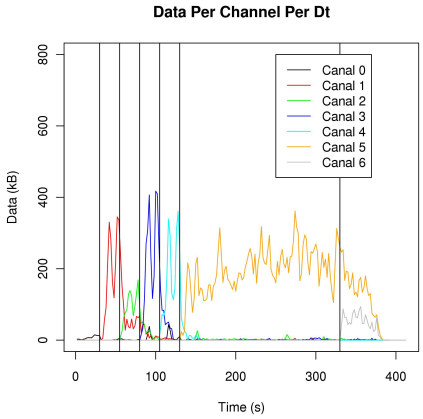


(c) Exemple de graphe du mouvement des pairs

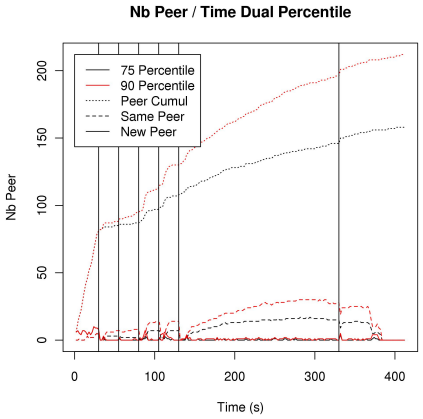
(d) Exemple de graphe du nombre de pairs par chaîne

FIGURE 4.2 – Exemples de graphe 1/2

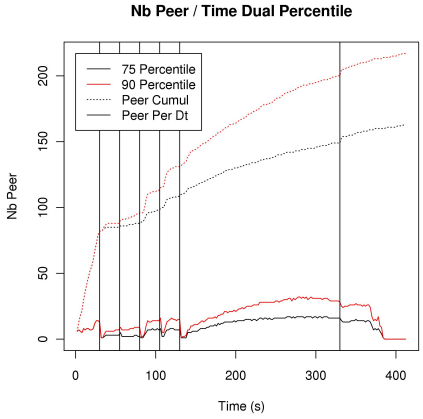
Les métriques retenues font partie des métriques qui ont été présentées dans la section 2.4. Par ailleurs l'idée était de pouvoir analyser simultanément un ensemble de traces, ainsi le programme



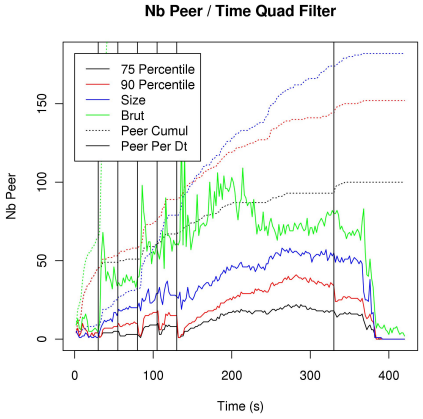
(a) Exemple de graphe de quantité de données par chaîne



(b) Exemple de graphe du mouvement des pairs pour 90 et 75 percentile



(c) Exemple de graphe du nombre de pairs pour 90 et 75 percentile



(d) Exemple de graphe du nombre de pairs avec les quatre filtrages

FIGURE 4.3 – Exemples de graphe 2/2

génère des courbes moyennes pouvant également inclure l'écart type à la moyenne. L'application peut donc réaliser les graphes suivants :

- Nombre de pairs instantanés : il s'agit du nombres de pairs observés dans le voisinage durant une fenêtre temporelle. Par défaut cet intervalle de temps est réglé sur 2 secondes. Le graphe affiche également le nombre de pairs cumulés, c'est à dire le nombre de pairs contactés depuis le début de la capture (exemple figure 4.2(a)).
- Quantité de données échangées : le graphe affiche deux courbes correspondantes à l'envoi et à la réception des données (exemple figure 4.2(b)).
- Mouvement des pairs : il s'agit du mouvement des pairs entre deux intervalles de temps, le graphe affiche là aussi le nombre de pairs cumulés (exemple figure 4.2(c)).
- Nombre de pairs par canal : un pair est attribué à un canal s'il n'a pas encore échangé de données avec le client précédemment. Cette analyse permet notamment de détecter les pairs restant dans le voisinage après un changement de chaîne (exemple figure 4.2(d)).
- Quantité de données par canal : à partir de l'extraction des pairs associés à chaque chaîne, on peut déterminer les données échangées par ces pairs et donc les données associées à chaque chaîne (exemple figure 4.3(a)).

Par ailleurs il est aussi possible de faire certaines analyses avec plusieurs critères de filtrage. Cela permet notamment d'évaluer le critère le plus approprié et d'observer la perte engendrée par un critère plus restrictif. Ainsi on peut effectuer les graphes comparatifs suivants :

- Nombre de pairs pour un critère de 75 percentile et 90 percentile (exemple figure 4.3(c)) ;
- Mouvement des pairs pour un critère de 75 percentile et 90 percentile (exemple figure 4.3(b)) ;
- Nombre de pairs avec quatre critères de filtrage, capture brute, taille, 75 et 90 percentile (exemple figure 4.3(d)).

L'application dispose également de fonctions d'analyse ne produisant aucun graphe, mais uniquement des retours sur le terminal :

- Symétrie des paquets : cette fonction évalue le rapport moyen entre le nombre de paquets envoyés et le nombre de paquets reçus par le client avec chacun des pairs de son voisinage. De cette manière nous pouvons évaluer l'importance que nous avons au sein de notre voisinage.
- Symétrie des échanges de données : cette fonction marche comme la précédente mais elle évalue cette fois ci le rapport entre la quantité de données envoyées et la quantité de données reçues. Cela nous permet de savoir si nous sommes uniquement consommateurs des données de notre voisinage ou si au contraire nous sommes un nœud source majeur.
- Extraction des super pairs : cette fonction analyse tous les pairs participant au voisinage et relève ceux qui apparaissent dans plusieurs fichiers de capture. En effet dans tous systèmes de diffusion de contenus, il y a nécessairement une ou plusieurs sources. Cette fonction couplée à un système de whois permet de voir si nous avons été ou non en contact avec l'un des serveurs sources.

4.4 Types d'analyse

L'application propose trois types « d'analyse ». Il s'agit en fait de trois types de sorties différentes. Chacune de ces analyses est représentée par un des boutons de l'interface graphique (figure 4.1). Le premier bouton (« Analyze ») correspond à une analyse classique ; elle va alors générer les différents graphiques dans des fenêtres séparées et produire diverses sorties sur le terminal telles que le nombre de pairs participant au nuage pour chacun des fichiers, les valeurs minimales et maximales des écarts types, elle permet également d'appliquer les fonctions produisant uniquement des sorties sur le terminal.

L'application propose également un mode d'export. Il va être demandé à l'utilisateur de choisir un répertoire où seront enregistrés les différents graphes. Par ailleurs pour éviter de perdre les informations affichées sur le terminal dans le cadre du fonctionnement classique, l'application opère une redirection temporaire de la sortie standard pour sauvegarder dans un fichier texte toutes les informations habituellement visibles dans la ligne de commande.

Enfin il est également possible de charger deux jeux de captures pour en réaliser ensuite une comparaison. L'application va alors effectuer les analyses en fonction des critères retenus par l'utilisateur. L'application utilisera alors une classe permettant de rédiger un document LaTeX opposant chacun des graphes en précisant le critère de filtrage utilisé et le nombre de pairs moyen.

Chapitre 5

Résultats

Dans cette section je présenterai les résultats obtenus au cours de mon stage. Je commencerai tout d'abord par les résultats obtenus de l'analyse des captures faites par la précédente stagiaire. Ainsi je détaillerai notamment les différences de fonctionnement entre les logiciels PPLive et Sopcast qui étaient alors les deux acteurs majeurs du marché. Ces principes de fonctionnement se baseront sur les observations faites sur les graphiques obtenus. Par la suite je présenterai les résultats de la campagne de mesures menée à l'occasion des jeux olympiques de Londres. Le but était de savoir si on observe une explosion de trafic liée à l'affluence d'un grand nombre de téléspectateurs. Je ferai également un comparatif avec le cas d'un jour « classique », la question sera alors de savoir si l'allure des courbes et donc les logiques de fonctionnement sont modifiées.

5.1 Résultats préliminaires

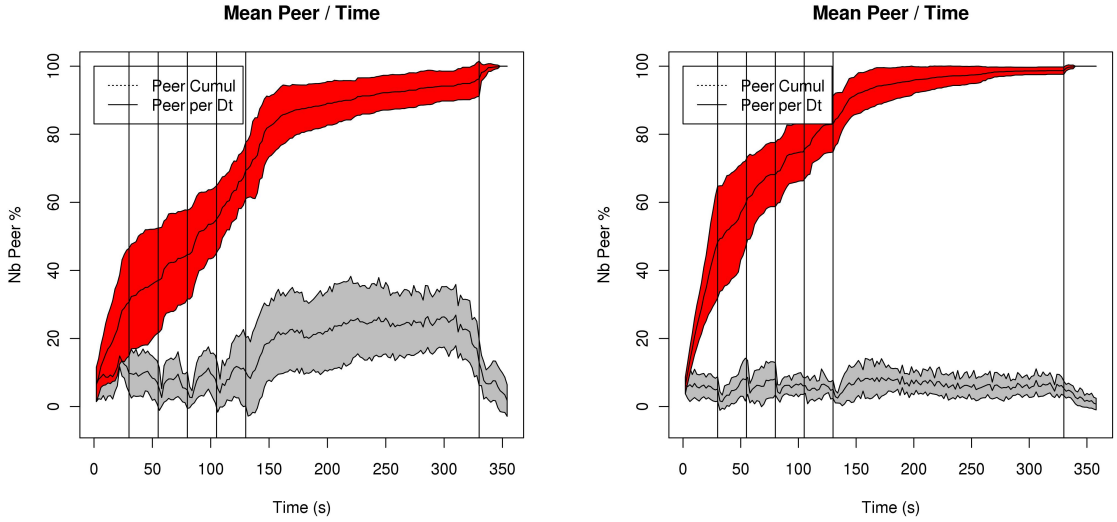
Mes premières analyses ont été faites à partir des captures réalisées par la précédente stagiaire. J'ai ainsi pu effectuer mes premiers tests sur ces fichiers et observer le comportement des diverses applications. Dans le cadre de cette présentation de résultats je me concentrerai uniquement sur une comparaison entre les logiciels PPLive et Sopcast pour le scénario de parcours de cinq chaînes. Pour ces résultats j'ai choisi un critère de filtrage retenant les pairs participant à 75% du trafic.

Les mesures qui sont analysées dans cette section ont été effectuées par Wan Manxue entre fin juin et début septembre 2011.

5.1.1 Nombre de pairs instantanés

Premièrement je vais détailler les observations liées au nombre de pairs. Ces graphes peuvent être présentés de deux façons. Les valeurs peuvent être affichées de manière brute, entraînant la mise en évidence d'écarts entre une capture faite à un moment de forte affluence sur le réseau et une capture faite à un moment où le public est moins important. L'application propose pour certaines analyses de normer les résultats. Dans ce cas on exprime le nombre de pairs instantanés et le nombre de pairs cumulés en pourcentage du nombre de pairs maximum.

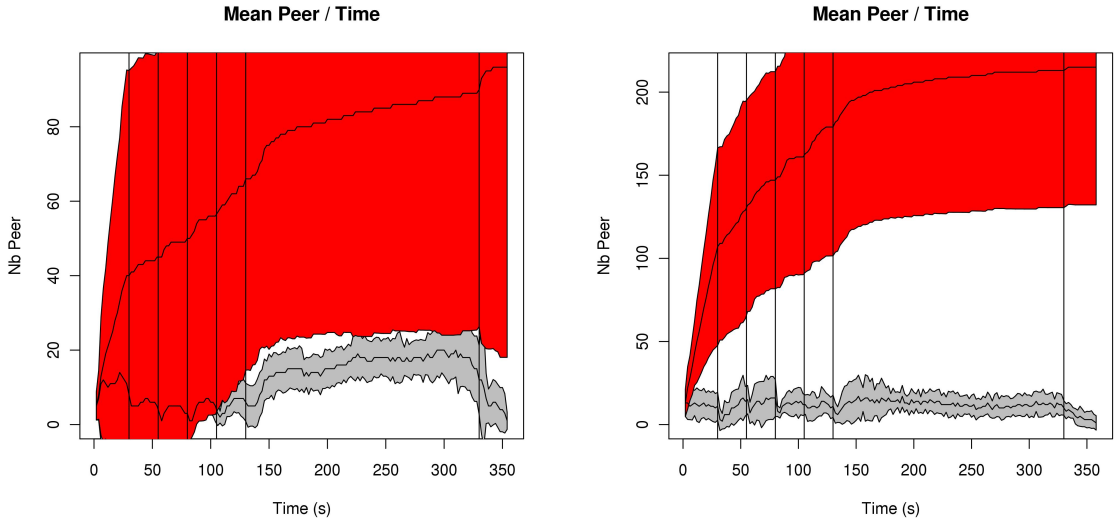
Les graphes de la figure 5.1 présentent ainsi l'évolution du nombre de pairs instantanés. La première observation que l'on peut faire est sur le nombre moyen de pairs contactés par chacun des logiciels. Ainsi Sopcast contacte en moyenne 97 pairs au cours du scénario, tandis que PPLive verra passer dans son voisinage 211 pairs. Ceci peut s'expliquer par des principes de fonctionnement différents dans la recherche du nombre de pairs. Ainsi Sopcast va rapidement constituer un voisinage mais il ne cherchera pas à l'étendre et effectuera un appel cyclique sur les pairs connus au sein de son entourage. De son coté PPLive va chercher à constituer un voisinage plus grand et répartir ses demandes de segments sur un plus grand nombre de pairs. De cette manière il est moins sensible au mouvement des pairs au sein de son voisinage.



(a) Sopcast 97 pairs en moyenne

(b) PPLive 211 pairs en moyenne

FIGURE 5.1 – Graphe du nombre de pairs normé



(a) Sopcast 97 pairs en moyenne

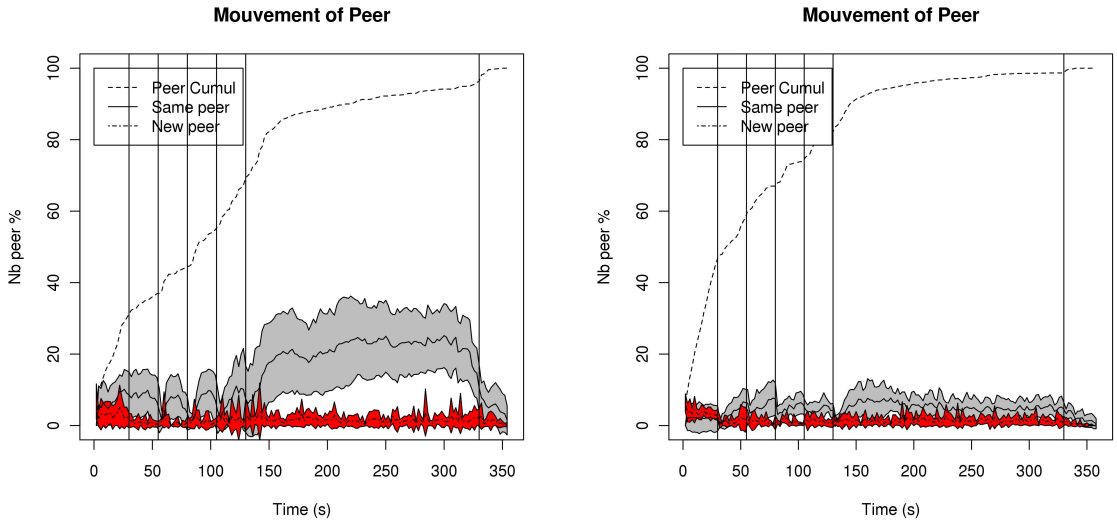
(b) PPLive 211 pairs en moyenne

FIGURE 5.2 – Graphe du nombre de pairs

Les graphes affichent également l'écart type à la moyenne sous la forme d'une surface colorée autour de la courbe ; dans le cadre du graphe normé (figure 5.1) la dispersion est assez faible. Cependant il ne faut pas oublier le caractère normé du graphe, cela veut donc dire que l'allure de progression dans la recherche des pairs est proche entre les différentes mesures. Mais il faut se poser la question du nombre de pairs contactés entre les multiples captures, celui-ci pouvant être très variable suivant les expériences. Pour cela on peut se référer au graphe du nombre de pairs non normé (figure 5.2) ; ainsi on voit notamment dans le cas de Sopcast que l'écart entre les différentes mesures est très important. Le voisinage aura compté entre 38 et 304 pairs. Pour PPLive cet écart est plus restreint car la taille du voisinage est compris entre 92 et 325 pairs.

5.1.2 Mouvement des pairs

Je vais maintenant présenter les résultats obtenus par l'étude du mouvement des pairs. Ici je ne présenterais que la version normée pour des raison de lisibilité (figure 5.3).



(a) Sopcast 97 pairs en moyenne

(b) PPLive 211 pairs en moyenne

FIGURE 5.3 – Graphe du mouvement des pairs normé

Dans cette analyse on travaille avec une fenêtre temporelle de deux secondes, on représente avec un trait plein au centre de la zone grisée les pairs identiques et l'écart type à l'extrémité de cette zone. Les nouveaux pairs sont modélisés par un trait discontinu au centre de la zone rouge dont les extrémités correspondent à l'écart type. Les pics d'apparitions les plus importants surviennent bien évidemment lors de la mise en route du logiciel ainsi que lors des changements de chaîne. On observe ici encore la différence entre les deux logiciels, Sopcast va activement utiliser le même panel de pairs durant la capture tandis que de son coté PPLive utilise en moyenne un ratio plus faible de pairs sur le long terme.

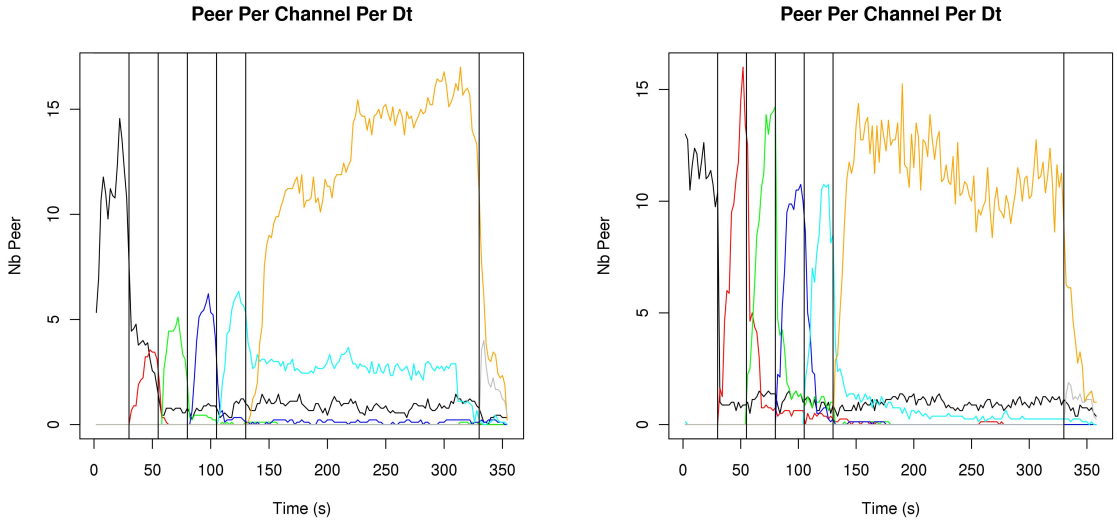
5.1.3 Pairs par canal

Dans cette analyse on observe le nombre de pairs qui sont associés à chaque canal ; pour qu'un pair soit associé à une chaîne il faut qu'il effectue des échanges avec le client durant la période de visionnage de la chaîne et qu'il ne soit pas déjà rattaché à un autre canal. Par exemple le pair X sera rattaché à la chaîne 2 s'il a échangé avec le client durant l'intervalle de temps compris entre 55 et 80 secondes et qu'il n'est pas rattaché à la période initiale ou à la chaîne 1. Cette analyse permet d'observer le

phénomène de résilience, c'est à dire le temps qu'il faut à l'application pour rompre le contact avec tous les pairs.

La figure 5.4 représente donc le nombre de pairs par canal observés en moyenne sur les divers enregistrements. Comme prévu, on observe des pics à l'arrivée sur chaque chaîne. On observe également un pic très important lors de la mise en route de l'application, qui s'explique par le critère de filtrage que nous utilisons. Lors de la mise en route, le logiciel se synchronise alors avec de nombreux autres clients. Le trafic engendré au total est relativement faible ce qui permet à notre critère de filtrage de retenir de nombreux pairs même si ceux-ci n'ont échangé que peu de données. On remarque d'ailleurs qu'une partie des pairs découverts lors de la mise en route du système vont être actifs tout au long du scénario.

Par ailleurs on observe là encore une différence. En effet il semblerait que Sopcast mette plus de temps à obtenir son voisinage maximal puisque le nombre de pairs augmente tout au long du visionnage de la cinquième chaîne. PPLive semble avoir atteint un maximum au cours des 30 premières secondes et se stabilise par la suite. De plus on observe une importante résilience pour Sopcast entre le passage de la chaîne 4 à la chaîne 5, cela peut être dû à l'utilisation de certains serveurs sources communs pour ces deux chaînes.



(a) Sopcast 97 pairs en moyenne

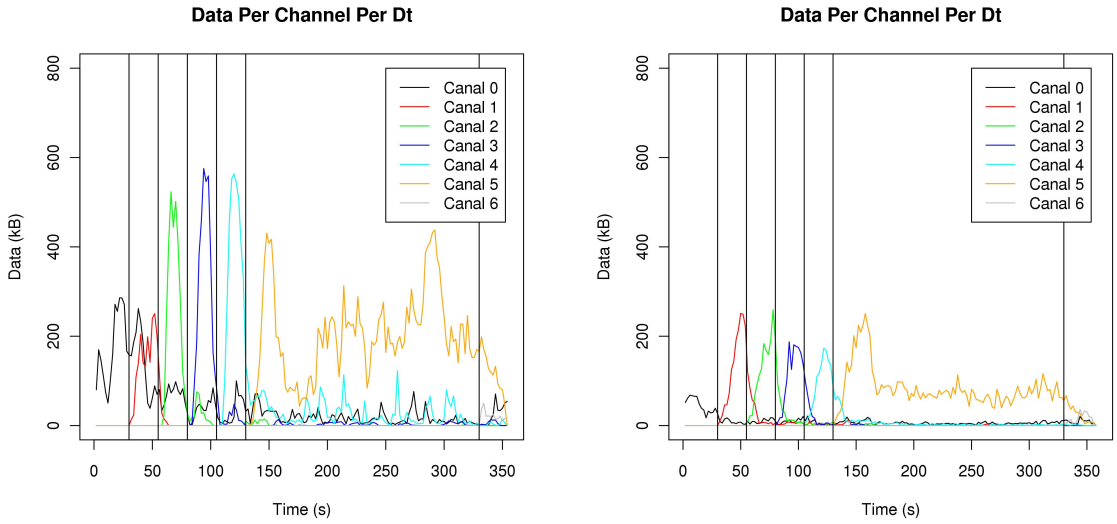
(b) PPLive 211 pairs en moyenne

FIGURE 5.4 – Graphe du nombre de pairs par canal

5.1.4 Quantité de données par canal

Dans le graphe 5.4 on présente les voisinages associés à chaque canal et on observe aussi l'activité persistante de pairs entre deux canaux. À partir des voisinages relevés précédemment nous pouvons calculer la quantité de données échangées avec ce voisinage et observer si la persistance d'activité est due uniquement à du trafic de signalisation ou si elle est bien liée à du trafic vidéo.

Le graphique 5.5 permet d'observer tout d'abord que, bien qu'une grande découverte de pairs soit observable lors de la mise en marche du logiciel, cela ne s'accompagne pas nécessairement d'un grand trafic. Par ailleurs, seul Sopcast continuera à échanger activement avec ces pairs par la suite. On observe également que le phénomène de résilience en contact de pairs ne s'accompagne que très faiblement d'une résilience dans l'échange de données. Ainsi les quelques pics de données ne correspondant pas à la chaîne



(a) Sopcast 97 pairs en moyenne

(b) PPLive 211 pairs en moyenne

FIGURE 5.5 – Graphe de quantité de données par canal

sont à associer à une certaine inertie du logiciel qui peut toujours échanger des données non nécessaires sur un intervalle très court. De plus on observe que globalement Sopcast échange d'avantage de données que PPLive, or dans les deux cas les canaux visionnés sont identiques et la qualité est théoriquement la même.

On remarque également que PPLive effectue un grand échange de données lors de son arrivée sur un canal qui correspond au remplissage du buffer, puis par la suite que les échanges vont se stabiliser. De son coté Sopcast semble régulièrement remplir son buffer et ne pas atteindre véritablement d'état stable ; cela s'explique à nouveau par son système de fonctionnement cyclique. Ainsi il va demander les mêmes segments à un cercle de pairs et relancer ses demandes périodiquement engendrant les multiples pics observables au cours du visionnage de la chaîne 5. Ce fonctionnement va alors engendrer une redondance dans l'échange des données mais va également offrir une plus grande robustesse en terme de réception de paquets.

5.1.5 Difficultés rencontrées

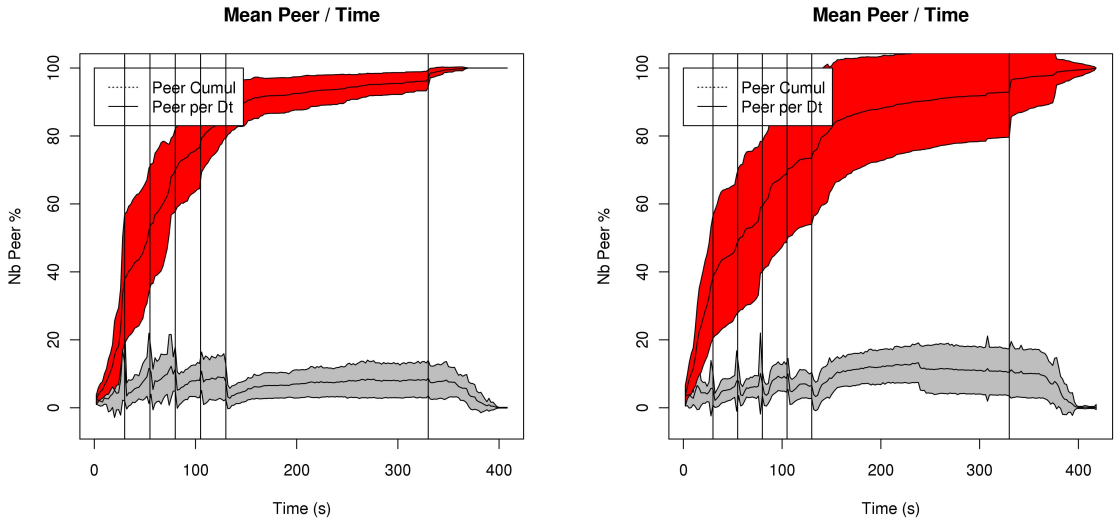
L'une des grandes difficultés au cours de cette partie de l'analyse a été la nécessité de faire des suppositions sur divers aspects. En effet je ne disposais pas de données qualitatives sur l'image au cours du scénario, et notamment sur la question de la fluidité de l'affichage vidéo. Par ailleurs je n'avais aucune assurance sur la façon dont avaient été exécutés les tests ; ont-ils été automatisés ou réalisés à la main et dans ce cas il me fallait prendre en compte une marge sur les phases de transition. Après discussion avec mon tuteur il a été décidé de considérer que certaines captures n'étaient pas fiables, car ne respectant pas correctement le scénario et pouvait donc être supprimé. J'ai ainsi été amené à supprimer des captures en raison d'un arrêt prématuré ou d'un comportement trop aberrant par rapport à la moyenne.

Un autre problème relevait de la partie sur l'étude statistique. En effet en raison du grand nombre de scénarios qui avaient été exécutés lors de la précédente campagne je ne disposais pas forcément d'un grand nombre de captures pour chaque scénario. Ainsi pour les analyses présentées dans cette section j'ai analysé 9 traces pour Sopcast et 8 pour PPLive. Ce facteur remet en question la pertinence des écarts types et moyennes ainsi calculés.

5.2 Résultats jeux olympiques

Dans cette section je présenterai les résultats obtenus suite à l'analyse des enregistrements effectués la veille et le jour de la cérémonie d'ouverture des jeux olympiques, soit les 26 et 27 juillet 2012 entre 22h et 23h. Je suivrai le même cheminement que lors de l'étude précédente, ici je chercherai non pas à faire un comparatif entre deux solutions de distributions de contenus en P2P mais à comparer un évènement à forte affluence et une journée « normale ». La question était de savoir si des circonstances particulières peuvent changer le comportement de l'application.

5.2.1 Nombre de pairs instantanés



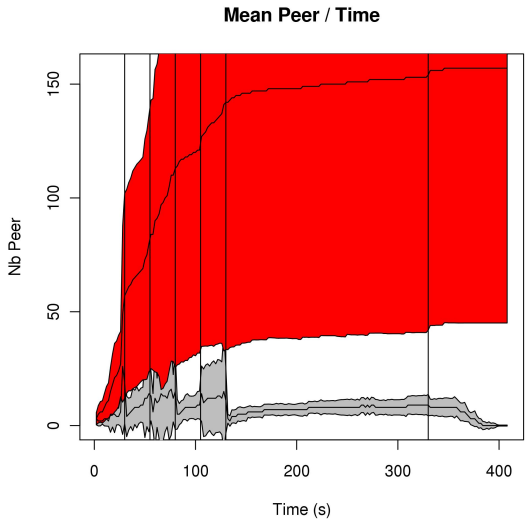
(a) Ouverture des jeux olympiques 158 pairs en moyenne

(b) Veille de l'ouverture 108 pairs en moyenne

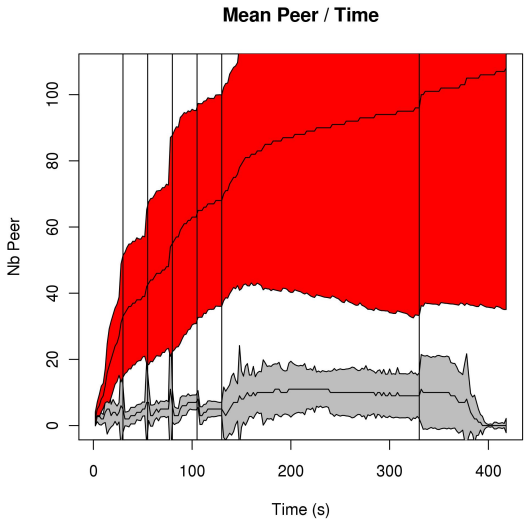
FIGURE 5.6 – Graphe du nombre de pairs instantanés normé

Observons tout d'abord le nombre de pairs instantanés via le graphe non normé (figure 5.6) il apparait bien évidemment un écart entre le nombre de pairs moyens contactés. Cela s'explique par un plus grand nombre de clients lors de la cérémonie d'ouverture. Selon les chiffres affichés par le site de PPLive on comptait un total de 11 000 personnes réparties sur les cinq chaînes les plus regardées, tandis que pour la soirée de la veille il y avait un total de 6 400 téléspectateurs. Cependant le rapport entre les nombres de pairs contactés et celui entre les nombres totaux de clients n'est pas le même ce qui pourrait nous amener à penser que dans le cas de la mesure lors de la cérémonie d'ouverture des jeux olympiques nous avons poussé le logiciel vers un maximum en terme de nombre de pairs dans le voisinage. Si l'on s'attarde sur la dispersion, on constate qu'elle est de moindre importance dans le cas du graphe 5.6(a) ce qui dénote une allure de courbe stable entre les différentes captures.

Concernant le graphe non normé, on observe là une dispersion bien plus importante lors de la cérémonie d'ouverture des jeux olympiques. Ainsi les mesures oscillent entre 36 et 412 pairs contactés alors que la veille on a contacté entre 51 et 295 pairs. On peut donc se demander si le logiciel ne subit pas quelque peu les effets d'une situation inhabituelle, ce qui se ressent sur la façon dont il réunit les pairs au sein d'un même voisinage.

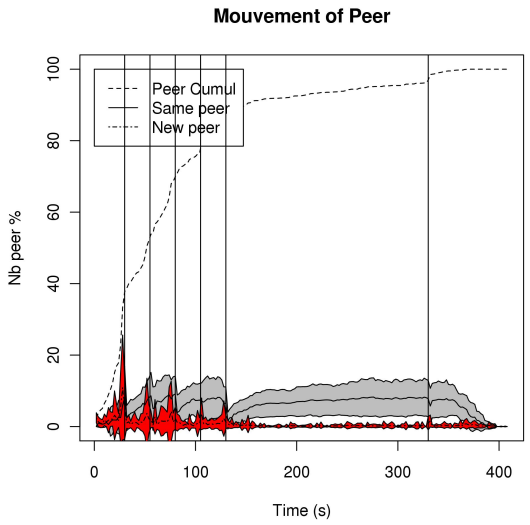


(a) Ouverture des jeux olympique 158 pairs en moyenne

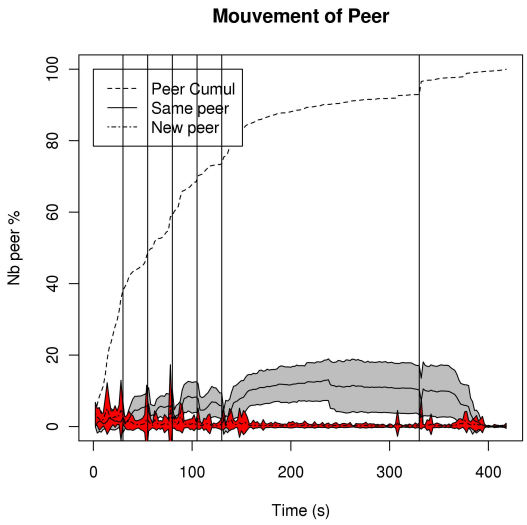


(b) Veille de l'ouverture 108 pairs en moyenne

FIGURE 5.7 – Graphe du nombre de pairs instantanés



(a) Ouverture des jeux olympique 158 pairs en moyenne

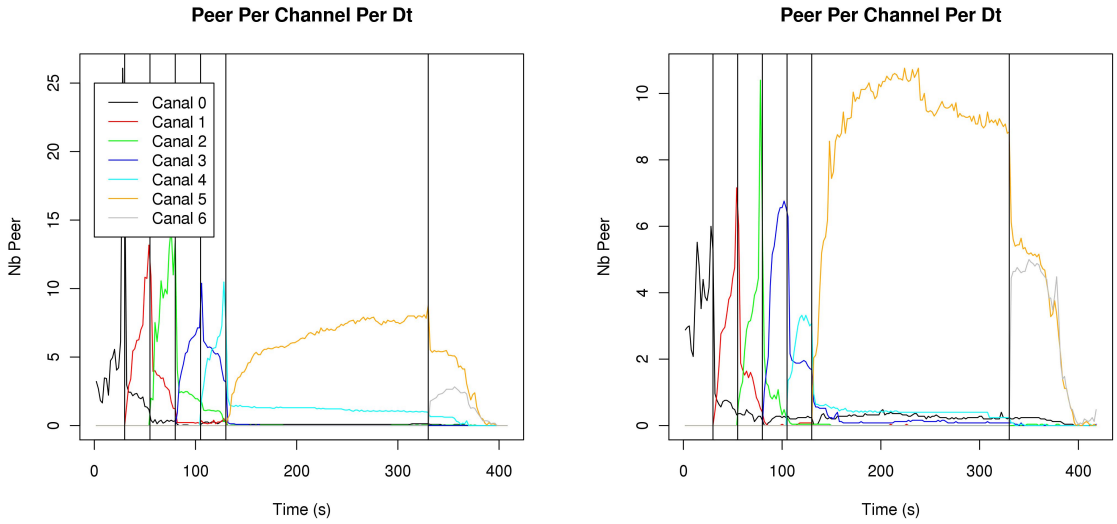


(b) Veille de l'ouverture 108 pairs en moyenne

FIGURE 5.8 – Graphe du mouvement des pairs normé

5.2.2 Mouvement des pairs

Dans les graphes 5.8 je fait une comparaison entre les dynamiques de mouvement des pairs. Le graphe est de nouveau représenté de manière normée. Ici on ne voit que très peu de différences entre les deux cas de figures si ce n'est un léger décrochages, autour de 250 secondes, dans le cas de la veille au cours du visionnage de la dernière chaîne.



(a) Ouverture des jeux olympiques 158 pairs en moyenne (b) Veille de l'ouverture 108 pairs en moyenne

FIGURE 5.9 – Graphe du nombre de pairs par canal

5.2.3 Pairs par canal

Dans le graphe 5.9 on représente les pairs rattachés à chaque canal et on constate à nouveau peu de différences à l'exception d'un pic important lors de la mise en route sur le graphe 5.9(a). Par ailleurs on constate dans les deux cas un trafic résilient lors du changement de chaîne qui reste jusqu'au nouveau changement de chaîne. Ce comportement pourrait être lié à un comportement prédictif du logiciel qui envisage la possibilité d'un retour en arrière.

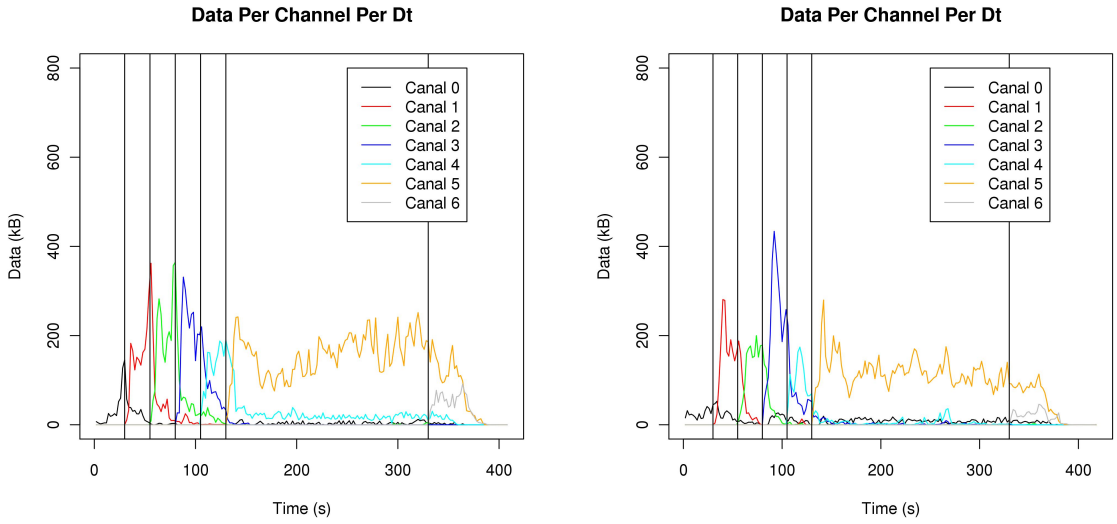
On retrouve également dans le graphe 5.9(b) le même décrochage au cours du visionnage de la chaîne 5 que celui observé précédemment dans le graphe 5.8(b).

5.2.4 Quantité de données par canal

Dans le graphe 5.10 détaillant les échanges de données associées aux voisinages constitués pour chaque chaîne on constate que dans le cas du graphe 5.10(a) le système n'atteint pas un état stable vraiment prononcé sur la chaîne 5 et que par ailleurs la résilience de contact avec les pairs du voisinage précédent s'accompagne également d'une résilience de trafic.

5.2.5 Limites de validité

Le problème de l'ensemble des captures que j'ai réalisé est qu'il se contente d'un enregistrement en un point du globe terrestre. En effet PPLive est un logiciel chinois à destination d'une population chinoise et par conséquent la plupart de ces utilisateurs sont situés en Asie. Ainsi il faut prendre en compte que notre situation géographique engendre une latence par rapport au système global qui peut entraîner des modifications quant au comportement de l'application. Par conséquent les mesures et analyses que j'ai faites devront être confirmées par des enregistrements faits sur place.



(a) Ouverture des jeux olympiques 158 pairs en moyenne

(b) Veille de l'ouverture 108 pairs en moyenne

FIGURE 5.10 – Graphe de la quantité de données par canal

Conclusion

L'objectif premier de mon stage était la réalisation d'un outil d'analyse de traces pour l'étude des systèmes de distribution de contenus pair-à-pair. Je devais par la suite tester mon application sur les captures réalisées au cours d'un stage précédemment encadré par mon tuteur. Ces objectifs se sont vus complétés en raison de l'opportunité offerte par les jeux olympiques de faire des enregistrements en situation de forte affluence. Leurs analyses ont permis de mettre en évidence un comportement du logiciel conforme à la normale, malgré un nombre de clients simultanément connectés plus important.

Ce stage a été pour moi l'occasion de découvrir de nouvelles technologies et notamment les possibilités du langage Python. Cela m'a aussi permis de mettre en pratique certaines des compétences acquises au cours de ma scolarité telles que la configuration d'un réseau et le paramétrage de matériel cisco. J'ai par ailleurs élargi mes connaissances sur les systèmes de distribution de contenus et les architectures pair-à-pair.

Au cours de ce stage j'ai pu de nouveau me confronter au fonctionnement d'un laboratoire comme je l'avais déjà vu lors de mon stage de $4^{\text{ème}}$ année aux Pays-Bas. Cette nouvelle expérience m'a conforté dans ma volonté de poursuivre mes études en thèse. Mon stage s'inscrivant dans le cadre d'un travail de doctorant, j'ai postulé pour une thèse sous la tutelle d'Olivier Fourmaux et Sébastien Tixeuil sur l'étude des interactions entre les utilisateurs et le réseau dans le cadre des applications de distributions de contenus audio/vidéo. Cependant je n'ai pas réussi à obtenir une bourse d'état et suis actuellement à la recherche d'une solution de financement.

Bibliographie

[1] H. Liu and G. Riley, "How efficient peer-to-peer video streaming could be ?," in *Proceedings of the 6th IEEE Conference on Consumer Communications and Networking Conference*, CCNC'09, (Piscataway, NJ, USA), pp. 994–998, IEEE Press, 2009.

[2] D. Rossi, E. Sottile, and P. Veglia, "Black-box analysis of internet p2p applications," *Peer-to-Peer Networking and Applications*, vol. 4, no. 2, pp. 146–164, 2011.

[3] "BitTorrent." Disponible à : `http://www.bittorrent.com/`, Accédé en 2012.

[4] "eMule." Disponible à : `http://www.emule-project.net/`, Accédé en 2012.

[5] "Skype." Disponible à : `http://www.skype.com/`, Accédé en 2012.

[6] "Joost." Disponible à : `http://www.joost.com/`, Accédé en 2012.

[7] "Sopcast." Disponible à : `http://www.sopcast.com/`, Accédé en 2012.

[8] "PPLive." Disponible à : `http://www.pplive.com/`, Accédé en 2012.

[9] "TV Ants." Disponible à : `http://www.tvants.com/`, Accédé en 2012.

[10] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale p2p iptv system," *IEEE Transactions on Multimedia*, vol. 9, 2007.

[11] N. Magharei and R. Rejaie, "Mesh or multiple-tree : A comparative study of live p2p streaming approaches," in *in Proceedings of IEEE INFOCOM*, pp. 1424–1432, 2007.

[12] N. Magharei and R. Rejaie, "Understanding mesh-based peer-to-peer streaming," in *Proceedings of the 2006 international workshop on Network and operating systems support for digital audio and video*, NOSSDAV '06, (New York, NY, USA), pp. 10 :1–10 :6, ACM, 2006.

[13] D. Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like peer-to-peer networks," *SIGCOMM Comput. Commun. Rev.*, vol. 34, pp. 367–378, Aug. 2004.

[14] "Sikuli Script." Disponible à : `http://www.sikuli.org`, Accédé en 2012.

[15] "Log Me In." Disponible à : `https://secure.logmein.com`, Accédé en 2012.

[16] "dpkt." Disponible à : `http://code.google.com/p/dpkt/`, Accédé en 2012.

[17] "rpy2." Disponible à : `http://rpy.sourceforge.net/rpy2.html`, Accédé en 2012.

[18] "tkinter." Disponible à : `http://wiki.python.org/moin/TkInter`, Accédé en 2012.

[19] C. Wu, B. Li, and S. Zhao, "Characterizing peer-to-peer streaming flows," *IEEE J. Select. Areas Commun*, 2007.

[20] M. Wang, L. Xu, and B. Ramamurthy, "Linear programming models for multi-channel p2p streaming systems," in *Proceedings of the 29th conference on Information communications*, INFOCOM'10, (Piscataway, NJ, USA), pp. 276–280, IEEE Press, 2010.

[21] D. Rossi and P. Veglia, "A hybrid approach to assess the network awareness of p2p-tv applications," *Int. J. Digital Multimedia Broadcasting*, vol. 2010, 2010.