Thèse

présentée pour obtenir le grade de docteur

de l'Ecole Nationale Supérieure des Télécommunications

Spécialité : Informatique et Réseaux

Paolo VEGLIA

On the network awareness of P2P-TV Applications

Applications TV pair-à-pair conscientes du réseau

Soutenue le 23 septembre 2011 devant le jury composé de

Président - Examinateur	Serge FDIDA	LIP6
Rapporteurs	Fabien MATHIEU Kurt TUTSCHKU	INRIA Universität Wien
Examinateurs	Olivier FOURMAUX Beatrice Pesquet Popescu	LIP6 Télécom ParisTech
Directeur de thèse	Dario ROSSI	Télécom ParisTech

"Es irrt der Mensch, so lang er strebt."
Johann Wolfgang von Goethe

Acknowledgements

Last three years have been a period full of achievements, knowledge, pleasure, pain, lot of fun and emotions. The PhD iter is the first real, big accomplishment of a person career and it would not have been possible without the invaluable help of a great number of great people.

The first person I really would like to thank is Dario, who advised me along all this process. His involvement in my work has been enormous and he always managed to find the time (even if I knocked in his office right in the middle of his epic battles with the e-mail inbox) to listen to my problems and to always give me smart and pertinent advise. His human side revealed qualities as astounding as his scientific skills: what surprised me is that I never felt a bit of haughtiness from him, he always listened to my ideas and comments, and he always thought about them seriously and without being prejudiced. Thank you very much, I learned so many things from you!

Also, during these years, I had the occasion to collaborate with other brilliant people whose opinions and knowledge made a big contribution for my scientific growth, thank to Marco, Emilio, Luca; a special thank goes to Fulvio and Olivier who introduced and pushed my in the world of research.

I kindly thank the reviewers for their time and for their precious help to the improvement of the manuscript and other members of the jury for attending the defense.

I would like to give a big hug to all the pizza-eaters colleagues of Télécom-ParisTech and Parisian friends who shared all the daily emotions with me. It would have been really hard without you guys. Thanks Claudio, Antonio, Stefano, Dorice, Xavier, Giuseppe, Thomas, Matteo, Davide, Nico, Maciej (is it spelled correctly at last? ;)), Amy, Sameh, Salma, Gege, Max e Jessica, Marianna, Mayssa, Mathilde, Maurice e Federico, Daniele. Then those who left earlier for other places: Anand, Aruna, Paola e Federico, Mattia: I hope to see all of you soon. A big thank also to the Via Italia's pizza which provided a big part of the carbohydrates and proteins required for a PhD. Thanks to all my teammates of CSPTT for the Wednesday night relief (and hard knocks). Another special thank is for Silvio, my flatmate, who shared with me not only an apartment but many wonderful and funny years of study since the poli.

Then all my Italian friends, who always warmly welcomed me back in Italy. I miss you. In sparse order, mixing Turin and Cuneo people: thanks to Dema e Laura, Bergi, Ema, Albi e Silvia, Maffe e Laura, Skiddo, Marty, Mirko e la George, Toni, Zalla, Obbe e Sara, Monky. And to the others that I forgot here.

A huge thank you to Erika, who has been at my side for almost all this way, helped me anytime and made everything magical. I think I was really lucky the day I found you. Really. Thank you so much.

Final thanks to my family that, even remotely, allowed all this happen, I would not be here without you. Grazie per avermi appoggiato in questo viaggio lontano ed essermi sempre stati cosí vicini. A big thought goes to my aunt who unfortunately could not live this moment with us. Ciao.

Abstract

This work originates in the context of the "NapaWine" European project and its main goal is to make P2P applications aware of the underlying network topology. This information (e.g., path capacity, latencies, etc.) helps to optimize P2P traffic and its benefits are twofolds: on one hand operators can limit traffic on peering links by confining traffic in their AS. On the other hand, users can experience higher quality of service due the proximity of neighbors.

To tackle the problem we first analyze existent applications and we especially gauge their level of network awareness: since many systems are closed source, we study them as black boxes by means of purely passive analysis, then we setup a controlled testbed and finally we make use of parallel active probing to gather dynamic neighbors properties.

As a second step we test state of the art chunk diffusion algorithms in a realistic simulator where we model latencies and access link capacities according to academic results. Within such a scenario we study how chunk diffusion performance suffers in presence of measurement errors and out-of-date system state knowledge.

Finally we develop an emulation scenario in which we are able to test real applications. Our testbed can scale up to 200 application instances and emulates the topology of the existing research network Abilene; it also allows to perform traffic engineering which we exploit to analyze coupling phenomenon between IP and overlay routing.

Résumé en français

Introduction

Au début de cette thèse le paradigme pair-à-pair était en train de devenir de plus en plus populaire et des nouveaux services étaient en train d'être déployés dans le réseau globale. Selon une étude de Hendrik Schulze [43], en 2008, environ le 70% du trafic des données européen était généré par des applications P2P. Cependant, la révolution promise par le P2P n'est pas encore avérée : cela est essentiellement dû aux limitations de l'ADSL (le système P2P n'a pas la capacité de soutenir le service) mais cela poudrait changer avec le déploiement de la fibre optique jusqu'aux utilisateurs finaux. Ceci confirme donc l'importance de notre travail.

Les motivations derrière la croissance initiale des systèmes P2P-TV sont multiples : d'abord, une grande partie de l'intelligence exigée par un système P2P est au bord du réseau, ainsi que ce ne soit pas nécessaire d'améliorer son infrastructure pour déployer un nouveau service. Le deuxième aspect important c'est qu'il n'y a pas un seul point d'échec : les graphes logiques de niveau 7 sont construits d'une façon distribuée et n'exigent pas d'équipement centralisé pour coordonner l'échange des informations. Troisième point, c'est difficile d'avoir un contrôle du trafic de données et cela a permis aux systèmes de partage de fichiers de gagner de plus en plus de popularité.

Aujourd'hui le paradigme P2P s'étend sur une large gamme de services de réseau, de la computation distribuée jusqu'aux systèmes de fichiers distribués; il est aussi en train de modifier la façon dont le contenu vidéo est distribué. Les opérateurs qui offrent ainsi la dénommée "triple-play" [1] sont en train d'utiliser la technologie multicast, laquelle, si d'un coté garantit une performance optimale, de l'autre ne permet la distribution du contenu que dans le même autonomous system. Les technologies vidéo P2P, tout en gardant une haute qualité d'efficience, pourraient remédier à cette limitation en permettant aux distributeurs de construire des marchés de distribution d'échelle globale.

Les systèmes de vidéo en direct comme `PPLive` [89], `TVAnts` [112], `SopCast` [107] sont déjà en train d'attirer un grand nombre d'utilisateurs. `PPLive` soutient d'avoir eu 200 millions d'installations et 104 millions d'utilisateurs actifs par mois en 2011 [109]. Des nombres si grands que ça, combinés avec les caractéristiques intrinsèques des overlays modernes [2], sont en train de préoccuper les opérateurs et fournisseurs de services Internet qui sont en train d'adopter des stratégies pour contrer le phénomène. Comcast, un provider américain [23], est le cas le plus célèbre et récemment a admis de gérer différemment le trafic P2P par rapport a l'autre trafic; en outre, des études ont démontré que la forme du trafic P2P est inversé par rapport au trafic quotidien. Cela semblerait dû à l'effet de la manipulation du trafic par les opérateurs.

Le défi est donc de faire si que les systèmes P2P, et spécialement vidéo en direct, se comportent synergiquement avec la couche réseau et aussi qu'ils coopèrent avec les fournisseurs de services. Cela peut porter des avantages pour les utilisateurs comme pour les fournisseurs d'accès:

---

[1] ADSL, télévision et téléphone

[2] nombre élevé de connections, flux avec large débit, trafic pas optimisé, etc.,

un trafic P2P mieux conçu peut améliorer drastiquement l'expérience des utilisateurs (e.g., bas délais, qualitée augmenté) tout en limitant les coûts opérationnels des opérateurs. En fait, si un noeud P2P est capable de choisir ses voisins dans le même système autonome, le trafic généré ne passera pas à travers des liens coûteux. On appelle cette habilité "Network Awareness" (NA) ou conscience du réseau.

Conscience du réseau

La définition de conscience du réseau est très simple: une application P2P est consciente du réseau ou "network aware" si elle a connaissance de la couche qui est en dessous d'elle même et utilise cette information pour ces algorithmes internes (i.e., programmation d'envoi des morceaux vidéo, gestion topologique). Pour raisons d'espace, on utilisera NA (Network Awareness) comme synonyme de conscience du réseau.

Intuitivement cette information peut répondre aux questions: est-ce que il y a un noeud que je peux contacter dans mon système autonome? Combien mesure-t-il le chemin vers ce noeud? Combien de sauts faut-il faire pour le rejoindre? Un rôle partiel de cette thèse est de déterminer quelles propriétés du réseau puissent être prises en compte pour améliorer les systèmes P2P.

| Approche	Travail	Année	Navigation de la topologie
P2P Mesures	[57]	2009	Latence
	[104]	2008	Débit
	[93]	2008	$\frac{Dbit}{Latence}$
	[12]	2006	Localité
Coopération avec ISP	[37]		Oracle (IETF ALTO WG)
	[3]	2007	Oracle
	[124]	2008	iTracker

Table 1: Une vue d'ensemble des approches pour atteindre la "network awareness

Les méthodes dont les données sont collectionnés peuvent changer d'un système à l'autre et la Table 1 résume les majeures efforts au présent. Les travaux sont divisés en deux grandes familles, mesures pair-à-pair et coopération avec les ISPs. La première consiste en des mesures exécutées par les applications (e.g., en utilisant des outils comme `ping` pour mesurer la latence ou `CapProbe` [46] pour l'estimation de la capacité du chemin) sans besoin d'aide d'une infrastructure externe. Toutefois, si la mesure de capacité, nombre de sauts, taux de perte ou mesure du débit de l'application est relativement facile à effectuer, autres métriques comme la capacité d'un chemin sont achevées en injectant du trafic additionnel dans le réseau qui peut biaiser les mesures lui même. Finalement ces techniques sont généralement conçues pour être utilisés seules et leur utilisation multiple en parallèle pourrait générer une mauvaisee conscience du réseau. Le deuxième groupe de techniques utilise l'aide de ISPs qui peuvent déployer des noeuds ainsi dénommé "oracles" dans des points stratégiques du réseau. Les oracles exposent des interfaces d'interrogation pour permettre l'échange d'informations avec l'ensemble des noeuds pour leur permettre de choisir le meilleur voisin. Un aspect positif de cette solution est la parfaite connaissance de la topologie du réseau par les ISP qui pourraient aussi potentiellement exploiter les oracles pour faire du trafic engineering basique. Dans cette thèse on se focalisera sur la classe de mesures P2P.

The Big Picture

En faisant référence à la figure 1, nous décrivons maintenant le contenu et l'organisation de cette thèse.
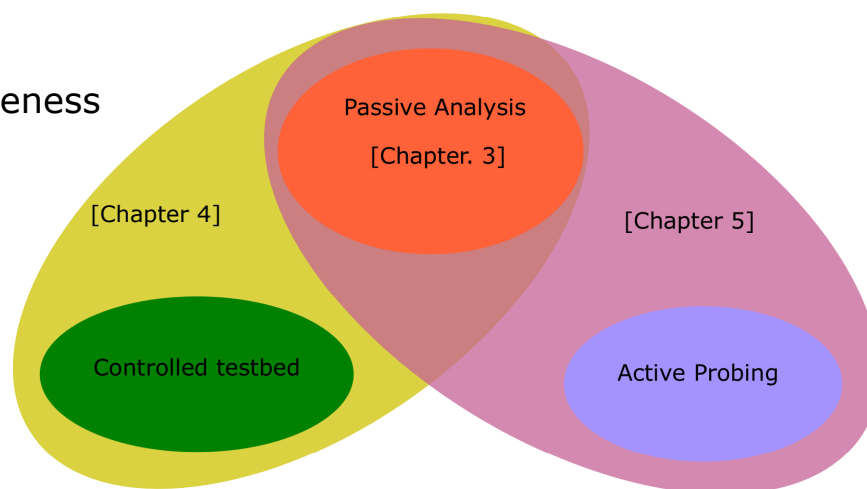
Figure 1: La vision d'ensemble

Mesure de la conscience du réseau

Un pas préliminaire dans le contexte du projet européen *NapaWine* c'est d'avoir une vue d'ensemble de l'état de l'art actuel des plus importants systèmes P2P déployés (PPLive, TVAnts, SopCast and autres). Notamment, notre but est d'évaluer le niveau de NA embarquée dans ces logiciels. L'obstacle majeur c'est la fermeture du code de ces systèmes et nous n'avons pas d'informations concernantes leur algorithmes ou protocoles. Donc notre première défi est de trouver une méthodologie qui puisse permettre l'évaluation; nous n'avons pas pris en compte l'approche reverse-engineering car il a un coût trop élevé et il n'est pas applicable à plusieurs logiciels. Par conséquence nous avons développé une méthodologie à "boite noire" qui permet d'examiner n'importe quel logiciel actuel, ou futur, suivant le paradigme P2P. La prochaine section expliquera en détail nos résultats.

Approche Passive Le premier travail en cette direction est présenté dans le Chapitre 3 dans lequel nous avons préparé une campagne de mesures pan européenne entre les partenaires du projet NapaWine pendant lequel le nous avons capturé le trafic généré par des application P2P depuis de points privilégiés. Nous définissons aussi une structure pour quantifier quels sont les paramètres de réseau qui modifient les décision des applications; ensuite nous appliquons notre méthodologie aux données collectées.

L'idée principale est d'utiliser une *Partition Préférentielle* (PP) : chacun des noeuds dans notre expérience prend contact avec autres peers dans le monde. Avec $\mathcal{P}(p)$ nous indiquons l'ensemble de peers contacté par le noeud p. Après, en considérant une propriété du réseau $X(\cdot)$, nous divisons $\mathcal{P}(p)$ en deux sous-ensembles utilisant la valeur de $X(\cdot)$ d'une façon telle qu'une classe devrait intuitivement être choisi par l'application (e.g., si nous utilisons la propriété RTT nous pouvons diviser les noeuds qui sont loin de ces qui sont voisins). À ce point nous computons le nombre d'octet qui sont échangés avec ce noeuds qui sont dans le sous-ensemble "proche" sur le totale d'octets envoyés. Intuitivement le plus ce nombre est grande, le plus l'application est biasée par la propriété examinée.

Selon l'analyse des données, nous observons que TVAnts et PPLive préfèrent légèrement échanger les données dans leur même autonomous system. Pour SopCast, par contre, cet effet d'agglomération est moins évident. Toutefois, dans tous les cas, les applications ne présentent aucune préférence vers le pays, sous-réseau ou nombre de sauts.

Approche Active Le problème principal de l'analyse purement passive c'est que seulement les métriques concernantes les noeuds (système autonome, pays, adresse IP) peuvent être étudiés. Au contraire, le métriques concernant les chemins sont difficiles à analyser a partir seulement de l'analyse passive des traces car nous ne connaissons pas ni les détail d'implémentation, ni les details des protocoles utilisés. Pour surmonter cette limitation est necessaire d'utiliser un approche active comme celui descrit dans le Chapitre 4, oú nous utilisons des conditions contrôlés pour tester le comportement des applications.

Nous obligeons les applications à télécharger le flux vidéo depuis une source contrôlé, puis nous commençons à changer les propriété du chemin (capacité, latence, taux de perte, nombre de sauts, etc) et nous observons comment l'application réagit à ces altérations.

Nous appliquons notre méthodologie à PPLive et nous investiguons quelles sont le propriétés qui influencent ses préférences d'échange. Le resultat principal est que PPLive semble plutot chercher les noeuds avec un haut débit mais il ne préfère pas explicitement les noeuds qui sont voisins (RTT ou nombre de sauts).

Avec un approche purement passive nous pouvons evaluer la NA d'une application vers quelques propriétés mais nous ne sommes pas encore capable de comprendre la situation générale. Nous avons besoin de la contribution de deux techniques. En corrélant les dernières découvertes avec l'analyse passive des traces, nous déduisons que la préférence vers les grands débits peut induire un niveau non négligeable et bénéfique d'agglomération géographique

Analyse passive augmenté avec sondage actif Finalement, comment nous venons de voire, méthodes actives ou passives seules ne peuvent pas contribuer à avoir une vue globale de la NA des applications: pour cette raison nous exploitons les deux méthodologies au même temps. Dans le Chapitre 5 nous présentons un cadre complet nommé Sherlock pour (i) analyser les applications P2P d'une façon "boite-noire" et (ii) décrire le trafic généré de manière compacte. Sherlock peut collectionner informations sur une application au moyen d'une analyse au même temps passive et active et présenter d'un seul coup ses empreintes digitales en forme de graphiques de Kiviat.

Nous implémentons la structure de Sherlock dans le logiciel démonstratif *P2PGauge* que nous avons présenté à *SIGCOMM09* [94] et que publions en open-source in [76]. P2PGauge est capable de mesurer, en fait, le niveau de NA embarqué dans les applications P2P

courantes en corrélant techniques actives et passives. Les résultats de notre étude sont appliqués à l'analyse du système de P2P-TV `SopCast`.

Mise en oeuvre du NA

Dans la deuxième partie de la thèse nous transférons notre intérêt de l'analyse des systèmes existants aux designs et algorithmes originaux et nous développons des méthodologie et outils pour les tester en conditions réelles.

Étude de simulation Avec les autres partenaires du projet Napa-Wine nous avons développé un simulateur nommé P2PTV-Sim qui est spécialement conçu pour la télévision P2P. Ses principales objectif de conception sont (i) être facilement customisé et (ii) les plus réaliste possible. Notre but est de comprendre si les algorithmes de NA sont robustes aux erreurs de mesure et aux mauvais connaissances de l'état du système ou si, par contre, des informations imprécises ont un impact négative sur les performances de l'overlay.

Le chapitre 6 expose une analyse simulative qui prend en considération différents facteurs: L7 overlay (e.g., chunk scheduling, management de la topologie, etc), réseau niveau 3 (modèles de latence, conditions dynamiques vs. statiques, etc) et l'interaction de deux niveaux (erreurs de mesure, perte de messages de signalisation, etc). Pour avoir une représentation complète des systèmes, les résultats sont exprimés depuis un point de vue soi des utilisateurs soi du réseau. En résumé, notre résultat principale est que les systemes P2P-TV sont généralement robustes aux erreurs de measure (latence ou estimation de la capacité), mais il sont, au contraire, profondément affecté par les erreurs de signalisation (e.g., perte de paquet ou vieux état du système), qui sont souvent négligés sans justification.

Malheureusement, malgré le précieux aperçu qu'un simulateur peut offrir du mechanisme de fonctionnement d'un overlay, il pourrait pas suffire pour capturer toutes les failles potentielles de l'architecture d'un système P2P: en effet, beaucoup de détails d'implémentation ne peuvent pas être analysé par simulation.

Étude d'émulation Pour dépasser cette limitation et approfondir notre analyse, nous utilisons une technique de émulation réseau niveau paquet. En détail nous considérons *modelnet* [113] qui est un environnement d'émulation qui permet l'expérimentation d'applications réseau réelles en émulant topologies réseau arbitraires avec latences, capacités des chemins et pertes. A partir de Modelnet, nous développons Modelnet-TE [70], qui est capable de performer de l'ingénierie du trafic en temps réel.

Dans le Chapitre 7 nous réalisons une campagne expérimentale d'interaction entre les niveaux du routage 7 et 3. Nous considérons un algorithme classique du répartition de charge, que nous comparons avec le routage IP standard. Comme applications P2P d'exemple, nous prenons BitTorrent, un des plus connu système de partage de ficher aujourd'hui, et WineStreamer, une application open-source de streaming en direct développée au sein du projet NapaWine et disponible à [119]. Nous émulons BitTorrent et WineStreamer avec des topologies soi réalistes (e.g., Abilene) soi simpliste qui sont habituellement utilisées aujourd'hui.

Les résultats de notre campagne expérimentale démontrent que le performance niveau utilisateurs peuvent être significativement affecté par, soi les mécanismes utilisés au niveau 3 (e.g., à cause des interactions avec le contrôle de cogestion TCP ou la logique d'échange de morceaux vidéo de l'application.), soi par des paramètres qui sont difficiles à contrôler dans l'internet, ce qui confirme l'intérêt pour les outils comme ModelNet-TE.

Table 2: Résumé des hôtes, sites, pays (CC), systèmes autonomes et type d'accès des noeuds prenant part aux expériences

Host	Site	CC	AS	Access	NAT	FW
1-4	BME	HU	AS1	high-bw	-	-
5			ASx	DSL 6/0.512	-	-
1-9	PoliTO	IT	AS2	high-bw	-	-
10			ASx	DSL 4/0.384	-	-
11-12			ASx	DSL 8/0.384	Y	-
1-4	MT	HU	AS3	high-bw	-	-
1-3	FFT	FR	AS5	high-bw	-	-
1-4	ENST	FR	AS4	high-bw	-	Y
5			ASx	DSL 22/1.8	Y	-
1-5	UniTN	IT	AS2	high-bw	-	-
6-7				high-bw	Y	-
8			ASx	DSL 2.5/0.384	Y	Y
1-8	WUT	PL	AS6	high-bw	-	-
9			ASx	CATV 6/0.512	-	-

Analyse Passive

Dans cette section nous présentons notre analyse du NA a partir d'une base de données purement passive.

DataSet Passive

Les partenaires du projet NapaWine ont pris part aux expériences en exécutant des logiciels P2P sur des PCs connectés dans les sous-réseaux institutionnels ou ADSL. En détail, la configuration comprenait un total de 44 noeuds, dont 37 machines connectées aux sites académiques/industriels et 7 machines connectées à des passerelles ADSL domestiques. Par conséquence, la configuration représente un nombre non négligeable d'environnements réseaux. Par la suite nous appellerons l'ensemble des machines qui ont pris part à l'expérience "NAPA-WINE peers".

Dans les systèmes P2P, les hôtes qui exécutent l'application (peers) forment un topologie logique en créant des liens virtuels sur lesquels ils transmettent et reçoivent les informations. Une source est responsable d'injecter dans le système le flux vidéo découpé en morceaux (que nous appelons chunks) de quelques Ko, qui sont après envoyés vers un sous-ensemble de ses voisins. Chaque peer peut contribuer à la diffusion des chunks, en les retransmettant vers ses voisins comme dans le système BitTorrent. Les différences principales entre P2P-TV et le partage de fichiers sont: (i) la source génère le flux vidéo en temps réel, (ii) les données doivent être livrées aux peers à débit presque constant et (iii) chunks doivent arriver en séquence pour être rapidement joués chez le récepteur.

Nous avons considéré trois différentes applications, `PPLive`, `SopCast` et `TVAnts`, et nous avons performé plusieurs expériences d'une heure pendant le mois d'avril 2008, pendant lesquelles nos noeuds regardaient le même canal à la même heure. Des traces au niveau paquet ont été collectées et analysées. Comme les applications P2P-TV sont très populaires dans les pays asiatiques, nous avons réglé chaque application sur la chaine CCTV-1 pendant les heures de pic [41]. Dans toutes les expériences, le débit du flux vidéo était de 384kbps, l'encodeur était Windows Media 9

Table 3: Moyenne et coefficient de variation de quelques statistiques collectionnées pendant les expériences

	PPLive		SopCast		TVAnts	
	Mean	cv	Mean	cv	Mean	cv
Avg RX rate [kbps]	549	0.22	482	0.05	415	0.04
Avg TX rate [kbps]	3150	1.02	262	1.04	396	0.65
N. contacted peers	22263	0.53	740	0.29	222	0.19
N. RX contrib peers	382	0.50	139	0.43	54	0.31
N. TX contrib peers	958	0.73	137	0.54	66	0.54
% non-resp. peers	27	0.87	25	0.73	30	0.31

Encoder et la qualité vidéo perçue par les utilisateurs était très similaire entre les systèmes. Les résultats reportés dans cette thèse concernent plus de 120 heures d'expériences, qui correspondent à environ 140 millions de paquets. Les traces collectées sont disponibles pour la communauté scientifique.

Une structure pour l'analyse de la sélection des peers

Notre but est de développer une structure rigoureuse pour découvrir la NA exposée par les applications, i.e., quels sont les paramètres de réseau qui sont pris en considération en distribuant le flux vidéo. Nous définissons un cadre flexible qui nous permet de ne pas seulement inspecter le niveau de NA d'un système par rapport à la couche L3, mais aussi pour comprendre si les peers se comportent de façon "amicale" entre eux, i.e. si les noeuds sont poussés vers un échange mutuel des données. En particulier nous considérons :

- **AS**(p): l'Autonomous System ou le peer p est localisé

- **CC**(p): le pays duquel le noeud p fait partie

- **NET**(p): le sous-réseau duquel le noeud p fai partie

- **HOP**(p, e): le nombre de sauts entre le p et e

- **SYM**(p, e): la symétrie de l'échange d'octets entre le noeud p et e

Définition de la structure

Définissons $p \in \mathcal{W}$ un peer qui appartient à l'ensemble NAPA-WINE $\mathcal{W}$. $\mathcal{P}(p)$ indique l'ensemble de peers qui *contribuent* et avec qui p échange des données. $\mathcal{P}(p)$ est composé de peers auxquels p a envoyé/reçu des informations. $\mathcal{U}(p)$ décrit le sous-ensemble de peers auxquels p envoie les données et $\mathcal{D}(p)$ le sous-ensemble depuis lesquels télécharge des données. $\mathcal{U}(p)$ et $\mathcal{D}(p)$ sont deux (non disjoints) sous-ensembles de $\mathcal{P}(p)$, et $\mathcal{U}(p) \cup \mathcal{D}(p) = \mathcal{P}(p)$.

$e \in \mathcal{P}(p)$ est un noeud arbitraire qui échange du trafic avec p. $B(p, e)$ est le nombre d'octets transmis par p vers e, tel que $B(e, p)$ est le nombre d'octets reçus par p depuis e.

En considérant maintenant un paramètre réseau $X(\cdot)$, $X(p, e) \in X$ indique la valeur observée de $X(\cdot)$ pour le paire (p, e). Nous répartissons $\mathcal{P}(p)$ dans deux classes en se basant sur $X(p, e)$, telle que une classe devrait intuitivement être préférée par l'application (e.g., bons peers contre

mauvais peers). Plus formellement, nous répartissons le support X dans deux ensembles disjoints : l'ensemble préfère X_p et son complément $X_{\overline{P}}$, tel que $X_P \cup X_{\overline{P}} = X$ et $X_P \cap X_{\overline{P}} = \emptyset$.

Pour faciliter la notation, nous indiquons avec $\mathbf{1}_P(p,e)$ la fonction identité qui prend la valeur 1 si $X(p,e) \in X_P$ et 0 autrement ; de la même façon $\mathbf{1}_{\overline{P}}(p,e) = 1 - \mathbf{1}_P(p,e)$. Pour rester général, nous nous concentrons sur le trafic sortant d'un noeud NAPA-WINE $p \in \mathcal{W}$, et nous définissons :

$$Peer_{U|P}(p) = \sum_{e \in \mathcal{U}(p)} \mathbf{1}_P(p,e) \tag{1}$$

$$Byte_{U|P}(p) = \sum_{e \in \mathcal{U}(p)} \mathbf{1}_P(p,e) \cdot B(p,e) \tag{2}$$

$$Peer_{U|\overline{P}}(p) = \sum_{e \in \mathcal{U}(p)} (1 - \mathbf{1}_P(p,e)) \tag{3}$$

$$Byte_{U|\overline{P}}(p) = \sum_{e \in \mathcal{U}(p)} (1 - \mathbf{1}_P(p,e)) \cdot B(p,e) \tag{4}$$

ou les indices U et D sont utilisés pour indiquer respectivement le trafic sortant et entrant. $Peer_{U|P}(p)$ conte le nombre de noeuds desquels p est un donateur et qui appartient à la partition préférentielle X_p. Pareillement, $Byte_{U|P}(p)$ représente le nombre total d'octets envoyés par p aux peers de la partition X_p. Au contraire, $Peer_{U|\overline{P}}(p)$ et $Byte_{U|\overline{P}}(p)$ représentent le nombre de peers e d'octets auxquels p envoie malgré leur appartenance à la partition non préférentielle $X_{\overline{P}}$.

En considérant maintenant l'ensemble complet $\mathcal{W}$ des peers NAPA-WINE, nous définissons le nombre total de noeuds et d'octets comme :

$$P_U = 100 \frac{Peer_{U|P}}{Peer_{U|P} + Peer_{U|\overline{P}}} \tag{5}$$

$$B_U = 100 \frac{Byte_{U|P}}{Byte_{U|P} + Byte_{U|\overline{P}}} \tag{6}$$

Intuitivement, P_U exprime les probabilités que le mécanisme de sélection des peers favorise la découverte et l'échange entre noeuds appartenant à la partition préférentielle X_P. Pareillement, B_U exprime la probabilité que n'importe quel octet soit envoyé vers des noeuds appartenant à la classe X_P. Clairement, plus grands P_U et B_U sont, plus grand est le biais vers la partition préférentielle concernant la métrique X. L'avantage d'utiliser ces simple métriques c'est qu'elles permettent une comparaison *directe* et *compacte* des différentes propriété du réseau et des systèmes P2P-TV car elles ne sont sensitives ni à l'unité de mesure, ni à la valeur actuelle de X.

Les métriques P_D et B_D peuvent être définie en considérant $e \in \mathcal{D}(p)$ dans la dérivation précédente.

Partitions préférentielles

Avec le terme partitions préférentielles, nous considérons :

- **AS**: $\mathbf{1}_P(p,e) = 1$ si et seulement si $AS(p) = AS(e)$,
  i.e., les deux noeuds sont localisés dans le même système autonome [3] ;

- **CC**: $\mathbf{1}_P(p,e) = 1$ si et seulement si $CC(p) = CC(e)$,
  i.e., les deux noeud sont localisés dans le même pays;

---

[3]CC et AS ont été déterminés en utilisant la base de données "whois".

- **NET**: $\mathbf{1}_P(p,e) = 1$ si et seulement si $\text{HOP}(e,p) = 0$,
  i.e., les noeuds appartiennent au même sous-réseau;

- **HOP**: $\mathbf{1}_P(p,e) = 1$ si et seulement si $\text{HOP}(e,p) < median[\text{HOP}]$,
  i.e., le nombre de sauts entre e et p est plus petit que la distance moyenne calculée entre tous les noeuds;

- **SYM**: $\mathbf{1}_P(p,e) = 1$ si et seulement si $1/2 < B(e,p)/B(p,e) < 2$,
  i.e., le nombre d'octets reçus (envoyés) est au moins le double des octets envoyés (reçus).

Si par AS, CC et NET le choix de la classe préférentielle est simple, les cas de HOP et SYM exigent une discussion supplémentaire. Si nous considérons la métrique HOP d'abord, le décompte de sauts $HOP(e,p)$ a été évalué à 128 moins le TTL des paquets reçus, étant 128 la valeur par défaut sur des hôtes avec système d'exploitation Windows. Nous utilisons la médiane de la distribution comme un seuil pour définir deux sous-ensembles. Comme la valeur de la médiane de nombre de sauts varie entre 18 et 20 selon l'application, nous utilisons un seuil fixe de 19 sauts pour toutes les applications. Cela signifie qu'environ 50% des noeuds tombent dans la classe préférentielle.

Dans le cas des mécanismes des avantages (incentive mechanisms), nous classifions un échange comme "symétrique" quand le nombre d'octets reçu est au maximum le double d'octets envoyés et vice versa. Nous soulignons que même si ce mécanisme définit une relation de symétrie non exacte, nous avons vérifié que les résultats ne sont pas trop sensibles au choix du seuil (voir Sec. 3.3.4).

Résultats expérimentaux

L'évaluation expérimentale de la NA de `PPLive`, `SopCast` et `TVAnts` est rapportée en table 4. Spécifiquement nous rapportons, pour les directions de upload (U) et dowload (D), le décompte de noeuds (P) et d'octets (B) pour chacune des différentes métriques considérées avant. Table 4 détaille les résultats concernant tout l'ensemble des apporteurs (P_U, P_D, B_U, B_D) et aussi l'ensemble des apporteurs en excluant les noeuds NAPA-WINE (P'_U, P'_D, B'_U, B'_D).

Conclusions

Dans cette section nous avons proposé une méthodologie pour comprendre quelles sont les métriques qui sont exploitées par les applications P2P-TV pour optimiser la diffusion vidéo. En considérant trois applications populaires (`PPLive`, `SopCast` et `TVAnts`) nous avons démontré que seulement `TVAnts` et `PPLive` exposent une légère préférence à échanger les données avec des noeuds qui sont dans le même système autonome. Cependant, il n'y a pas de preuve d'une préférence vers les noeuds dans le même sous-réseau, avec un chemin plus court et il n'apparaît pas de mécanisme d'avantage (incentive) en aucun des logiciels observés.

Nous croyons qu'un meilleur niveau de NA doit être embarqué dans les systèmes P2P-TV pour mieux exploiter et optimiser l'allocation des ressources des fournisseurs de service. Dans le contexte du projet NAPA-WINE nous sommes en train d'examiner comment atteindre ce but (améliorer la localisation du trafic, emprunter des chemins plus courts, exploiter les connaissances topologiques, etc.).

Table 4: La conscience du réseau comme "peer-wise" et "byte-wise" biais

| | | Download | | | | Upload | | | |
| | | Non-Napa | | All Contributors | | Non-Napa | | All Contributors | |
Net	App	B_D' %	P_D' %	B_D %	P_D %	B_U' %	P_U' %	B_U %	P_U %
AS	PPLive	6.5	0.6	12.8	1.3	0.8	0.2	1.8	0.5
	SopCast	0.6	0.7	3.5	3.9	1.7	0.7	6.4	3.9
	TVAnts	7.3	3.3	32.0	13.5	11.6	1.8	30.1	9.6
CC	PPLive	6.5	0.6	13.1	1.4	1.1	0.3	2.1	0.6
	SopCast	0.6	0.8	4.0	4.4	1.7	0.8	7.2	4.4
	TVAnts	7.6	4.0	37.9	16.3	14.3	3.1	37.7	12.5
NET	PPLive	-	-	9.9	0.8	-	-	1.4	0.3
	SopCast	-	-	2.0	2.6	-	-	3.5	2.6
	TVAnts	-	-	18.1	6.7	-	-	18.1	5.4
HOP	PPLive	42.2	41.1	51.4	42.4	30.4	40.4	31.7	41.0
	SopCast	29.0	40.7	37.9	48.0	45.9	43.0	56.9	49.8
	TVAnts	62.1	55.0	81.1	71.9	57.8	53.0	78.9	67.2
SYM	PPLive	3.3	4.8	4.3	5.0	-	-	-	-
	SopCast	6.7	13.0	7.8	14.2	-	-	-	-
	TVAnts	12.4	10.9	20.0	14.3	-	-	-	-

Analyse hybride

En cette section nous utilisons deux différents ensembles d'expériences afin de mesurer la NA d'un système P2P-TV par rapport aux métriques peer-wise et path-wise discutées plus tôt :

Méthodologie

- D'un coté, nous exploitons une *plate-forme d'essai active* pour imposer des conditions artificielles (longueur du chemin, délai, pertes, étranglement du débit du chemin) sur des chemins particuliers.

- D'un autre coté nous adoptons une approche basée sur des *mesures passives*, avec lesquelles nous performons des mesures en direct et au même moment depuis des points spéciaux dans l'Internet pour étudier les propriétés (comme le système autonome ou la localisation géographique) qui appartiennent aux noeuds de l'overlay.

Propriétés concernant le chemin: plate-forme d'essai contrôlée

Pour les préférences liées aux métriques concernant les chemins, nous avons monté une plate-forme d'essai qui impose des conditions de réseaux artificielles comme dans [5], duquel notre approche diffère par deux raisons particulières. Premièrement, nous décidons de contrôler complètement les métriques. Cela veut dire que, au contraire de [5] oú les restrictions sont appliquées en outre aux conditions du réseau actuel, nous connaissons les conditions des différents noeuds concernés par l'expérience. Deuxièmement, nous ne testons pas seulement l'impact des métriques en isolement mais nous étudions aussi leur effet combiné.
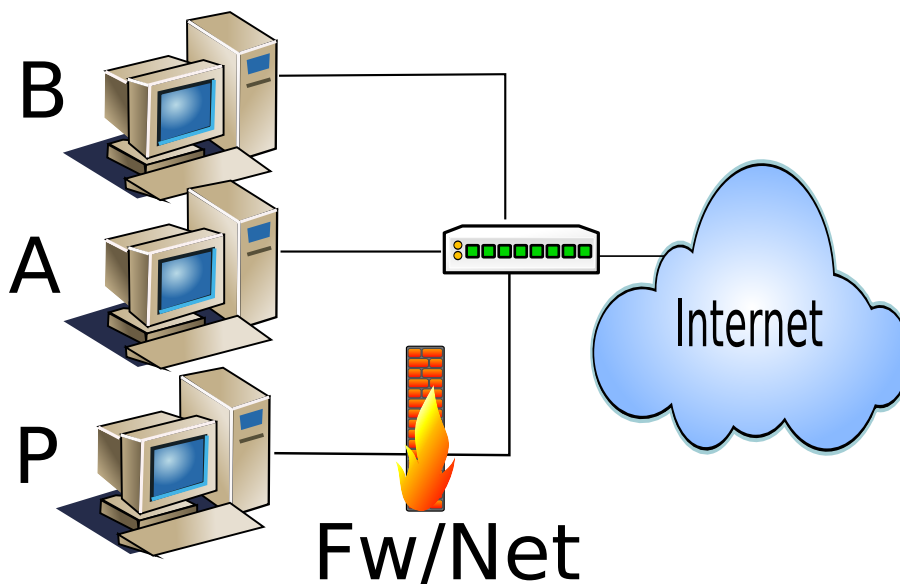
Figure 2: Métriques concernant les chemins : configuration du banc d'essai

La configuration utilisée pour toutes les expériences actives est reportée en figure 2. Nous utilisons trois ordinateurs modernes équipés avec processeurs dual-core qui exécutent un logiciel de P2P-TV (`PPLive` 2.4) sur windows XP. Deux machines A et B sont connectées à un commutateur de paquets a travers des interfaces ethernet à 100 Mbps. Le trafic est observé à la sonde P qui est connectée au switch par une autre machine, nommée Fw/Net dans la figure, qui joue le rôle de bridge, firewall et émulateur réseau. Remarquez qu'une large partie d'utilisateurs, et la source aussi, sont accessibles par l'internet.

Pendant la phase de démarrage, toutes les machines A, B et P exécutent l'application pendant 5 minutes. Pendant ce temps initial (oú nous vérifions que la vidéo est jouée correctement et synchronisée entre les machine), P naturellement reçoit la plupart du trafic par des noeuds dans Internet. Après, au moment $F_{on} = 5$ minutes, des règles du firewall sont établies sur la machine Fw/Net afin de bloquer le trafic provenant depuis internet vers P, qui, à partir de ce moment, peut recevoir seulement le trafic depuis A ou B. Dans ce cas, les machines A et B recevront encore les données par les noeuds distants en Internet, mais notre sonde ne pourra pas recevoir la totalité du trafic par A et B.

Nous introduisons après, à partir de $R_{on} = 10$ min, des règles d'émulation du réseau (comme la perte de paquets, RTT délai, réduction du débit, etc) sur les chemins qui relient la sonde P aux hôtes A et B. Nous soulignons que, comme notre but est de comprendre comment la "peer selection" marche pendant les opérations normales, nous vérifions que la sonde P est en train de recevoir et de jouer correctement la vidéo. Quand une métrique X est considérée en isolation, nous aggravons volontairement les conditions de réseau seulement pour le chemin qui relie la machine A à P, en configurant correctement la discipline de queue (queuing discipline) sur la machine Fw/Net. Les règles pour le chemin entre B et P, au contraire, sont appliquées seulement pour l'étude de l'importance relative des différentes métriques (e.g., delai sur $A \rightarrow P$ et pertes sur $B \rightarrow P$). Finalement, les conditions de réseau artificielles sont enlevées à $R_{off} = 20$ m et les limitations du firewall sont enlevées à $F_{off} = 25$ m.

Cette configuration nous permet de nous focaliser sur la décomposition du bit-rate reçu par P parmi ses apporteurs (i.e. A, B et hôtes internet), et d'exprimer d'une façon simple et intuitive
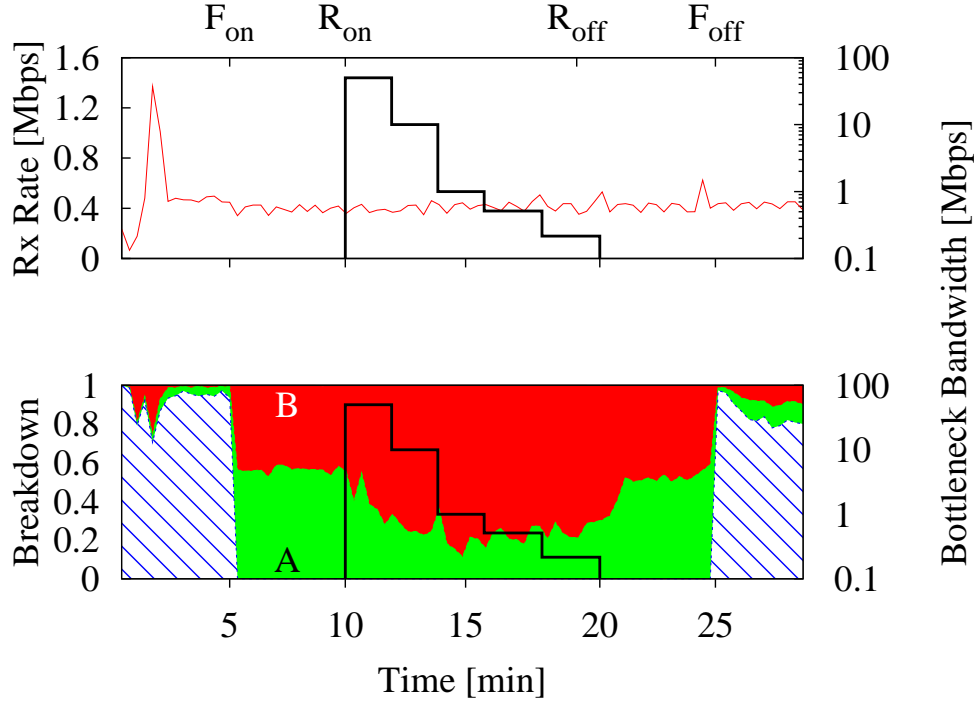
Figure 3: Métriques concernant les chemins : le débit agrégé de `PPLive` est représenté en haut et la décomposition par rapport à A, B et les hôtes internet en bas pour différentes capacités (a) et taux de perte (b). Les profils de la limitation de la capacité et de l'imposition des pertes sont reportés avec les lignes noires et font référence à l'axe de droite. `PPLive` montre une grande sensibilité vers le changement de capacité alors qu'il réagit seulement à des grands taux de pertes.

la NA des applications P2P-TV. En fait, en considérant la période pendant laquelle les règles d'émulation sont activées sur le chemin entre A et P, nous pouvons imaginer intuitivement que s'il n'y a pas de biais par rapport à une métrique déterminée X, la décomposition du trafic est insensible aux variations de X. Au contraire, une décomposition variante est le réflexe de la NA vers X, oú l'amplitude de la variation est un vague indicateur de la sensibilité du système vers X.

Résultats expérimentaux: métriques concernant les chemins

Nous reportons dans ce résumé à titre d'exemple seulement les résultats obtenus par la plate-forme d'essai active et en particulier ceux concernant la capacité des chemins.

Les résultats de cette expérience sont reportés en figure 3. Le temps de l'expérience est affiché sur l'axe x, alors que les temps du début et fin du firewall sont reportés sur l'axe en haut pour référence. Un profil décroissant limitant le débit est appliqué à partir du R_{on} au moyen d'un filtre "token bucket", qui suit des marches de $C = \{50, 10, 1, 0.5, 0.25\}$ Mbps chaque 2 minutes comme montré par la ligne noire épaisse. Les valeurs du débit du bottleneck sont montrés sur l'axe y de droite et le bottleneck est enlevé à R_{off}. L'évolution temporelle du débit agrégé à la machine P est montré dans la portion haute du graphique (moyenne calculé toutes les 20 secondes). Il est possible de voir, après une phase de démarrage $t < F_{on}$ pendant laquelle le débit entrant arrive à 1.2Mbps, le débit agrégé vers P est stable autour du 400 Kbps, qui compte pour les données vidéo et les données de signalisation. De plus, nous pouvons remarquer que le débit est stable pendant tout l'expérience, ce qui suggère que le "shaping" du trafic ne perturbe pas la qualité du service

perçu.

Le graphique en bas montre la répartition du trafic entrant vers P par rapport aux différents hôtes qui envoient le trafic vers P : l'hôte A est montré en bas en vert, l'hôte B en rouge et le reste du trafic internet en pointillé. C'est facile de voir que, avant que les règles du firewall soient mise en place à $t < F_{on}$, plus du 80% du trafic entrant est reçu depuis les hôtes internet extérieurs. Des que les règles du firewall démarrent a $t = F_{on}$, P est obligé de recevoir la totalité du trafic exclusivement depuis A et B : comme pendant $F_{on} < t < R_{on}$, la capacité du chemin n'est pas encore forcée, le trafic est divisé également entre A et B parce que les conditions du réseau et d'état de l'application (play-out time) sont pareilles. Puis, dès que l'émulation limite la capacité à 50 Mpbs à R_{on}, `PPLive` commence tout de suite à préférer l'hôte libre B, qui ensuite apporte la plupart du trafic vers P.

Cette observation est importante car elle signifie que (i) `PPLive` est *extrêmement* sensible au débit et (ii) pourrait trop réagir ou estimer incorrectement le débit.

Finalement les limitations de la capacité sont enlevées à R_{off}, ce qui enlève l'inégalité dans la répartition du trafic en faveur de B ; cette situation reste stable jusqu'au moment où les limitations du firewall sont enlevées a F_{off}, après lesquelles les hôtes internet redeviennent les majeurs apporteurs.

Sherlock

Dans cette section nous appliquons l'outil de visualisation Sherlock pour analyser la NA des applications. Nous suivons une méthodologie hybride qui mélange [5,22] dans un logiciel démonstratif, nommé P2PGauge, qui exploite les techniques actives et passives pour déduire résultats complets. Notre logiciel est disponible open-source a l'adresse [76]. En détail, nous trouvons toujours les deux différentes catégories de métriques : celles qui concernent les noeuds et celles qui concernent les chemins.

Il est nécessaire de souligner que `P2PGauge` exploite une exploration active des noeuds contactés par l'application P2P sous analyse : même si l'outil est capable d'analyser des traces passives, les résultats sont beaucoup plus fiables si les mesures sont exécutées simultanément à l'application. Donc, en utilisant P2PGauge, nous avons collecté un nouveau ensemble d'expériences pendant lesquelles nous avons étudié l'application SopCast. Dans ces expériences une sonde unique située en France est utilisée pour suivre différentes chaînes vidéo à différentes heures, explorant ainsi un large spectre de localisation du contenu et de popularité des chaînes. De plus, nous avons été particulièrement attentifs au choix d'un contenu *local* (ligue des champions) et aussi étranger (informations et films en langue étrangère). Encore une fois, chaque expérience est conduite indépendamment pour éviter de biaiser les résultats.

Processus d'analyse

Nous décrivons maintenant le processus d'analyse à l'aide de la figure 4. Dans nos expériences, un client SopCast non modifié tourne sur la machine sonde, le trafic de laquelle est capturé par l'outil `P2PGauge` qui est exécuté sur la machine à l'écoute. `P2PGauge` analyse le trafic généré par SopCast et recueille des statistiques sur (i) les caractéristiques concernant les noeuds au moyen de l'analyse passive et (ii) des caractéristiques concernant les chemins en envoyant des paquets sondes vers les noeuds contactés par SopCast.

Avant de commencer l'explication des métriques et des caractéristiques choisies, nous voulons souligner une implication importante. Concernant les méthodologies passives, P2PGauge recueille les caractéristiques concernant les noeuds au moyen d'une base de données locale [65] (e.g., geo
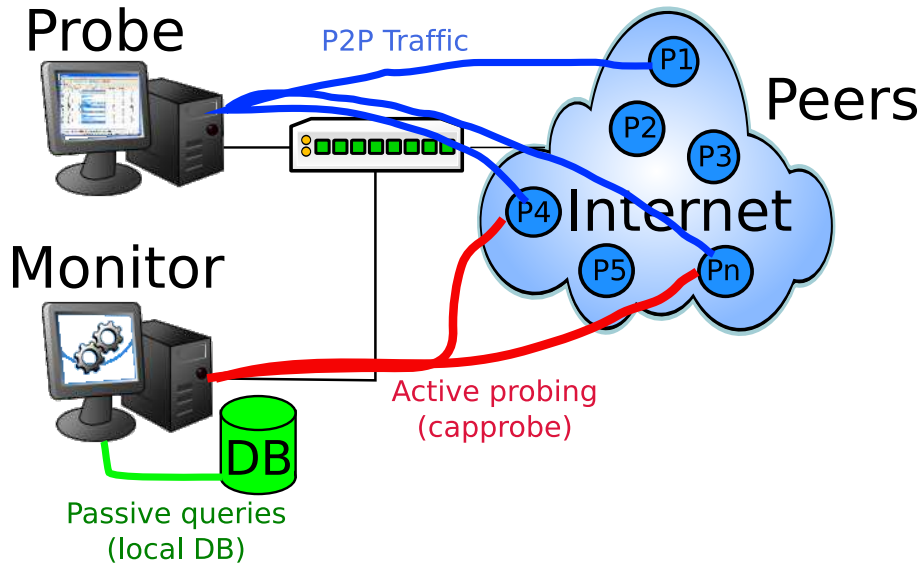
Figure 4: Processus d'analyse de P2PGauge

localisation et numéro d'AS, etc.) ou à travers des simples analyses et inférences (e.g., longueur du préfixe IP, débit de l'application, nombre de sauts, etc.). L'analyse passive ne peut pas interférer avec le trafic observé, mais elle pourrait être limitée pas la vitesse d'accès de la base de données : comme les API de la base de données supportent plus de 40,000 questions par seconde, cela ne constitue pas un problème de vitesse.

Pourtant, l'outil performe aussi des mesures *actives* pour collecter des propriétés concernant les chemins, possiblement interférant avec le trafic observé : pour cette raison, les mesures actives doivent être utilisées le moins possible. Notez en fait que, bien que les mesure sont exécutées par une machine dédiée, le monitor et la sonde partagent la même liaison d'accès. Considérez par exemple le problème d'estimation de la capacité : les coûteuses techniques du sondage actif des liens (comme les mesures de bande par trains de paquets) ne sont pas appropriées à nos objectifs, et nous avons besoin de techniques de mesure plutôt légères (comme celles basées sur la dispersion des paires de paquets). En raison de cette observation, nous nous résoudrons à utiliser CapProbe [46] pour estimer activement la capacité du chemin, le RTT et aussi le TTL IP (a partir duquel nous pouvons connaître le nombre de sauts). Pour chaque noeud, nous performons $N = 100$ mesures en envoyant des paires de ICMP paquets l'une après l'autre (back-to-back), et chaque paire est espacé par $\Delta T = 0.5$ seconds.

Pour limiter le nombre de paquets sondes pendant les phases intenses de découverte du réseau, nous limitons le nombre de procès de sondage active à $C = 50$. Bien que le volume de paquets sondes soit limité à $R = 2C/\Delta T = 200$ paquets par seconde, mesurer activement tout le voisinage serait une tache prohibitive. Nous soulignons que seulement une partie des noeuds prend effectivement part à l'échange du contenu alors que les autres n'échangent pas de données. Si ces noeuds peuvent constituer un pourcentage significatif de la population des noeuds, il n'ont quand même pas de poids pour ce qui concerne le volume du trafic. Comme nous sommes intéressés par la majeure partie du volume du trafic, nous limitons donc les mesures actives seulement aux noeuds qui apportent activement à la distribution du flux. Spécifiquement, nous considérons seulement ces peers qui envoient au moins deux paquets dans une fenêtre temporelle ΔT. Cette simple heuristique nous permet de nous concentrer sur la plupart du trafic (e.g., au dessus du 95% dans

le pire des cas de PPLive), tout en limitant le biais causé par le sondage actif. Notez que cette heuristique est très robuste et peut être appliquée à d'autres classes du trafic comme le partage de fichiers.

Définition des métriques

Partition préférentielle La métrique plus simple et plus intuitive que nous utilisons est la partition préférentielle déjà présentée dans le chapitre 3. Pour chaque caractéristique F, l'ensemble $\mathcal{N}_k$ des noeuds contactés jusqu'à $T = k\Delta T$ est divisé en deux sous-ensembles disjoints $\mathcal{N}_k = \mathcal{N}_k^{close(F)} \cup \mathcal{N}_k^{far(F)}$, afin que les noeuds qui sont "proches" au peer analysé X en terme de caractéristique F soient regroupés ensemble.

Spécifiquement nous utilisons les règles suivantes pour partager l'ensemble. Nous considérons les peers qui tombent dans le même AS ou pays, dans le même sous-ensemble de noeuds proches. Concernant le NET nous utilisons un seuil fixe de 16 bits, au dessus duquel nous considérons les noeuds comme voisins. Finalement, pour RTT, HOP et CAP, nous utilisons un seuil variable basé sur la médiane calculée sur tous le noeuds.

En nous basant sur ces simples partitions, nous quantifions le niveau de préférence en calculant la pourcentage d'octets que le peer X a échangé avec le noeud de la partition préférée.

Kullback-Leibler (KL) Comme seconde métrique nous considérons la mesure de divergence de Kullback-Leibler (KL), qui est une mesure connue de distance entre deux distributions de probabilité (pdf) p et b :

$$KL(p\|b) \quad = \quad \sum_{x \in X} p(x) \log \frac{p(x)}{b(x)} \tag{7}$$

Nous utilisons la divergence KL pour mesurer la différence entre les pdf concernant (i) les noeuds (peer-wise) et (ii) les octets (byte-wise) d'une caractéristique F. Autrement dit, nous évaluons le pdf de F, soit en décomptant chaque peer une fois, soit en prenant en compte le volume de trafic en octets que le peer loin a échangé avec X. La divergence KL nous indique si les deux distributions sont pareilles (KL$\simeq$0) ou s'il y a quelques disparités (KL>0). Notez que, au contraire d'avant, une large valeur de KL ne peut pas être lue comme un indicateur de préférence : plutôt, elle indique simplement que il y a un biais entre le nombre de peers exhibant une valeur donnée pour une caractéristique X et le nombre d'octets échangés avec eux. Par exemple, une large valeur de KL$_{AS}$ ne signifie pas qu'un large nombre d'octets est échangé avec des noeuds du même AS mais plutôt que *quelques* AS apportent plus des données que d'autres. En d'autre termes, les valeurs de KL élevées correspondent à un biais majeur, qui pourtant ne se traduit pas nécessairement en conscience majeure du réseau.

Résultats expérimentaux

Nous adoptons maintenant une représentation Kiviat de l'ensemble de caractéristiques, exprimées au moyen des métriques PP e KL pour SopCast. La figure 5 reporte les graphiques, arrangés d'une façon telle que les caractéristiques mesurées par inférence passive (AS, CC, NET) sont représentées sur les trois axes en haut, tandis que les caractéristiques qui impliquent des mesures actives sont dans la partie basse. Les deux métriques PP et KL sont reportées respectivement en 5-(a) et (b). Notez aussi que les axes s'étendent jusqu'à 1.0 (2.0) pour la métrique PP (KL).
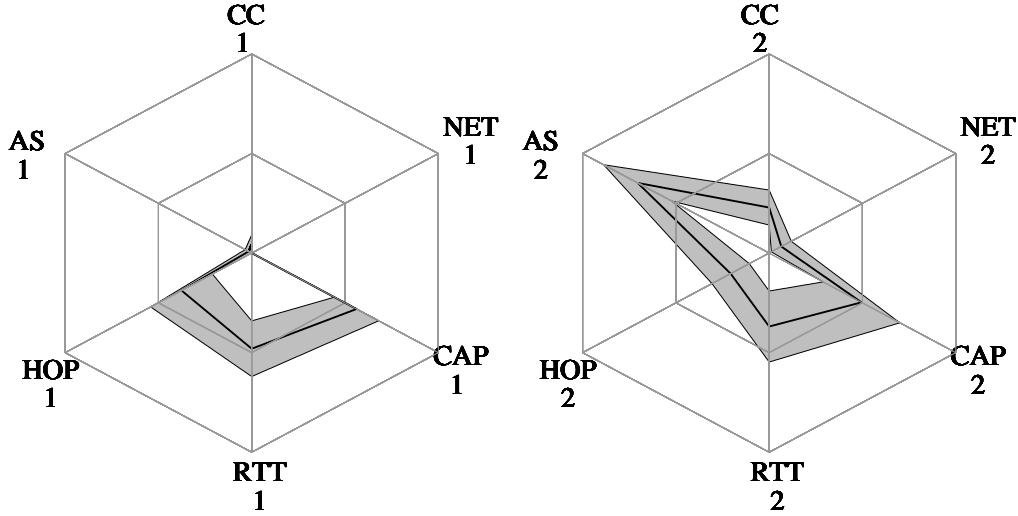
Figure 5: Représentation de la conscience du réseau : graphes Kiviat de la (a) Partition Préférentielle et (b) de la mesure de divergence de Kullback-Leibler pour SopCast. Les caractéristiques assemblées à partir de mesure passives sont montrées en haut (AS, CC, NET) alors que les mesures actives sont sur les axes en bas (HOP, RTT, CAP).

Kiviat reporte, comme d'habitude, la moyenne et la déviation standard sur tous les noeuds. Considérons d'abord la partition préférentielle. Il est facile de noter que, tandis que les expériences se réfèrent à un contenu qui est très populaire en EU (football match de ligue des champions), et aussi très localisé (équipes françaises), pourtant SopCast réussit à trouver seulement peu de noeuds qui sont dans le même sous-réseau ($PP_{NET} \simeq 0\%$) AS ou pays ($PP_{AS} \simeq 1.6\%$ and $PP_{CC} \simeq 4.5\%$).

L'analyse de graphiques kiviat nous permet d'arriver à la conclusion que la localisation géographique pourrait être causée par autre préférences : par exemple, la simple stratégie de choisir les noeuds a haute capacité, qui dans notre dataset sont aussi ceux appartenant au même AS. Et en fait, tout cela est corroboré par la caractéristique de capacité ($PP_{CAP} > 50\%$), qui montre une légère préférence vers les noeuds à haute capacité. Au contraire, aucune préférence n'est montrée pour les noeuds proche en terme de RTT, car juste la moitié du trafic global est échangé avec des noeuds proches ($PP_{RTT} \simeq 50\%$), fait qui suggère qu'il n'y a pas de préférence pout le RTT. Pareillement, le fait que $PP_{HOP} < 50\%$ confirme que des chemins légèrement plus longs peuvent être empruntés pour chercher ces noeuds à haute capacité.

Nous considérons d'après le graphique du KL (b). Dans ce cas [4], un large biais est manifesté pour la capacité KL_{CAP}, corroborant ainsi l'hypothèse de la sélection avide. Un biais encore plus large est manifesté pour KL_{AS}, qui dans ce cas, correspond à une distribution du trafic déséquilibré. Dans ce cas peu d'ASs sont les majeurs apporteurs : cependant, telles ASs diffèrent du AS du noeud analysé et leur occurrence pourrait être le résultat d'autres politiques de choix. Globalement nous pouvons conclure que les applications populaires P2P-TV comme SopCast n'ont pas encore considéré le problème de la NA.

---

[4] nous rappelons que valeurs du KL plus larges indiquent un biais plus large

L3 Host
• Multiple Classes
• Different Upload/
  Download Bandwidth

L3 Network
• Static
  - Overlay only (latency =0)
  - Constant delay
  - Meridian latency matrix
• Dynamic

L7

L3

L7 Peer and Overlay
• Chunk Schedulers
  - {ru/r, lu/r, lu/la, lu/ba, lu/pa}
• Topologies:
  - G(n, M), G(n, p)
  - Topology Manager
    + {static/dynamic}

L3/L7 Interaction
• Measurement Errors:
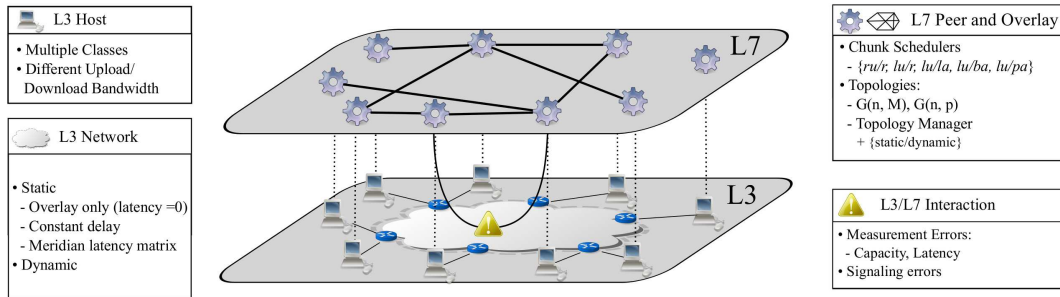 - Capacity, Latency
• Signaling errors

Figure 6: Aperçu du cadre d'évaluation : vue d'ensemble des composants L3 et L7 et de l'interaction des deux.

Analyse simulative

Dans cette section nous utilisons un simulateur pour tester des algorithmes [12, 17, 93, 104] en considérant les facteurs importants que nous modélisons avec un niveau croissant de réalisme. Un premier problème est que les systèmes NA P2P-TV prennent typiquement des décisions en se basant sur les caractéristiques des autres noeuds mesurés : pour cela, réussir à avoir de mesures correctes sur Internet est une tâche difficile et il est très important de comprendre l'implication des *erreurs de mesures* sur les performances du système. Un deuxième problème est que les algorithmes de "scheduling" ont été évalués en comptant sur une connaissance parfaite, mais pas réaliste, de l'état du système (buffer-maps des voisins) : pour cela, il est important d'évaluer l'impact de l'inconsistance du système (dû à la perte des messages de contrôle ou à de vieilles informations).

Description du système

Cette section donne une vue d'ensemble de la structure que nous avons conçu pour comparer les systèmes P2P-TV, laquelle est à disposition de la communauté scientifique à l'adresse [78]. Le simulateur guidé par les évènements (event-driven) customisé prend en considération différents éléments, qui sont visuellement présentés en 6. D'un point de vue de haut niveau, la structure consiste en deux couches : (i) la couche physique réseau L3 du dessous et une couche logique en haut nommé overlay, qui sont couplées par différents modelés d'interaction.

Du point de vue de L3, au bord de l'architecture nous trouvons les hôtes, qui sont physiquement interconnectés au réseau L3 à travers des liens d'accès, qui se comportent comme des étranglements, et qui sont modelés comme des pairs capacité–délai. Les hôtes sont attachés aux routeurs de borde, qui constituent le point d'accès du trafic P2P dans le réseau, lequel nous modelons avec un niveau croissant du détail. De point de vue du L7, les hôtes exécutent des applications P2P-TV, que nous exprimons en terme d'algorithmes (chunk scheduling, selection des hôtes, management de la topologie) et du graphe logique en résultant. Finalement, nous modelons l'interaction L3/L7 en tenant compte que, dans le monde réel, différentes source d'erreurs peuvent apparaître dans n'importe quel point du procès (e.g., perte de paquets de signalisation, mesures biaisées, etc.).

Maintenant nous détaillons chacun des composants du système, et les motivations à la base des nos choix.

Table 5: Répartition en classes des propriétés des noeuds

Class	Ratio	BW_D	BW_U	t_{TX}
I	10%	∞	5.0 Mbps	20 ms
II	40%	∞	1.0 Mbps	100 ms
III	40%	∞	0.5 Mbps	200 ms
IV	10%	∞	0 Mbps	∞

Composants L3

Avec le terme "composants L3" nous indiquons des objets du monde physique, comme (i) hôtes et (ii) routers, qui sont interconnectés par un réseau (iii).

Hôtes Les hôtes sont des machines qui exécutent les instances des applications P2P et sont caractérisés par une interface physique vers le réseau L3. Les hôtes sont divisés en deux classes en accord avec leur capacité d'upload BW_U alors que nous considérons le débit entrant BW_D infini. Cela est une acceptation raisonnable en cas d'accès asymétrique, donné que nous supposons ultérieurement que l'étranglement est placé au bord du réseau (qui représente le cas commun aujourd'hui et il est généralement supposé par d'autres recherches sur la P2P-TV [93, 104] et sur le partage des fichiers [83]).

Dans nos simulations nous considérons N_H=2000 hôtes divisés en 4 classes, ou la moyenne $BW_U(i)$ pour la classe i-ème est répartie comme décrite en table 5, qui est en accord avec [83, 104]. La première colonne de table 5 reporte la répartition en classes : la plupart de la population est constituée par des peers avec une vitesse moyenne, avec une présence non marginale de peers très rapides et très lents. Dans la classe i, le débit sortant de chaque peer p est configuré à $\nu \cdot BW_U(i)$ où ν est une variable aléatoire uniformément distribuée en $[0.9, 1.1]$ (e.g., le vrai débit sortant de chaque peer p dévie au plus du 10% de la moyenne de la classe).

Router Chaque hôte est "single-homed", i.e., attaché à un single router d'accès, qui modèle le premier router IP du réseau d'agrégation (e.g., le BRAS pour le réseau ADSL). Dans nos simulations nous considérons un nombre de routers égal à $N_R = 100$ et nous utilisons un simple "mapping" entre hôtes et routers : chaque hôte est couplé par hasard à un router, il suit qu'en moyenne il y a $N_H/N_R = 20$ hôtes attachés a chaque router.

Comme montré en figure 6, les routers son placés au bord du réseau et agissent comme point d'accès en formant une "full-mesh" logique. Chaque router garde des statistiques concernant les paquets qui passent à travers ses interfaces et il différencie le trafic entre *remote* (i.e., trafic qui est réexpédié vers le réseau) et trafic *local* (i.e., trafic qui est reflété vers un autre lien d'accès attaché au même router). Notez que cette façon nous permet d'avoir une simple mesure de la localisation du trafic $P\% = local/(local + remote)$.

Réseau Le réseau niveau 3 modèle l'interconnexion des routers : dans ce travail, nous considérons différents modèles de réseau.

Si nous considérons le lien d'accès comme le "bottleneck", il n'y aura probablement pas de formation de queues dans le noyau du réseau : comme cela, le réseau modèle simplement le délai bout-a-bout. Dans ce cas, la topologie réseau, est bien représentée par une matrice statique des latences, ou la latence représente le délai de propagation. Nous considérons plusieurs modèles de réseaux statiques, à partir d'un overlay idéal (ou le délai est

donné seulement par la durée de transmission d'un morceau) vers des modèles plus réalistes comme Meridian [38] (où le délai bout-à-bout est dérivé à partir de mesures réelles effectuées par de multiples hôtes). Nous considérons aussi le cas où la congestion peut se produire dans le réseau et nous employons des matrices des latences *dynamiques*, où la latence entre deux hôtes peut être différente parmi plusieurs morceaux vidéo. Nous soulignons que le cas où le trafic P2P est (i) minoritaire ou (ii) prévalent sont à considérer séparément. Dans le premier cas, qui est typique aujourd'hui et que nous considérons dans ce travail, la congestion est due au trafic dit de "background" : nous modelons cet effet simplement en variant la latence entre deux morceaux consécutifs aléatoirement. Dans le deuxième cas, les liens du réseau devraient, eux aussi, être modélisés, pour que l'ingénierie du trafic (répartition de la charge, optimisation périodique, etc.) puisse être appliquée pour gérer la matrice de trafic générée par le P2P-TV. Comme le cas (i) est le plus commun, nous considérons le cas ou le trafic P2P est prévalent hors intérêt.

Composants L7

Avec le terme composants L7 nous indiquons les composantes de niveau plus élevé comme (i) les peers, qui sont des instances du niveau 7 des applications P2P-TV. En plus de ce détail, nous modelons les peers en définissant les algorithmes qu'ils implémentent : spécifiquement, chaque peer doit (ii) gérer la topologie overlay et (iii) programmer l'envoi des chunks.

Peer Chaque peer établit et maintient différentes connections logiques vers des autre peers : nous dénotons avec $N(p)$ l'ensemble de peers dans le voisinage de p. Comme, dans les systèmes mesh-push, les morceaux vidéo ne sont pas forcement reçus dans l'ordre, les peers doivent avoir une buffer-map B qui décrit les morceaux qui ont étés reçus et stockés dans la mémoire. Pour un peer p, nous indiquons avec $B(p)$ sa buffer-map et avec $c \in B(p)$ le fait que le peer p ait reçu le morceau c. La dimension de la buffer-map $B(p)$ détermine la performance de la P2P-TV comme décrit ci-après : des buffer-maps larges réduisent la probabilité de perte des morceaux mais ils augmentent le décalage temporel entre la source et les destinations ; au contraire, buffer-maps plus petites réduisent le décalage mais augmentent les pertes (les morceaux qui arrivent après le délai ne sont plus utiles et peuvent être considérés perdus).

Topologie overlay les liens logiques établis par les peers forment une topologie overlay. Pour optimiser leur performance, les peers peuvent modifier la topologie créée : i.e., ils arrangent leur voisinage pour exploiter l'hétérogénéité de la population, pour optimiser globalement la topologie en se basant sur des décisions locales.

Dans ce travail, nous nous focalisons sur le management de la topologie en le considérant soit comme (i) un "outil boîte" noire qui induit un type particulier de graphe ou (ii) comme un algorithme qui règle constamment la topologie. En détail, pour le point (i) nous considérons différentes topologie Erdos-Renyi $G(n, p)$ qui sont créées à $t = 0$ et ne sont jamais changées, et donc cela définit un voisinage logique fixé à $t = 0$. Pour (ii) nous considérons aussi un processus de management de la topologie qui tourne constamment et adapte la topologie initiale, en se basant sur les propriétés mesurées (e.g., latence, capacité, etc). En fait, comme les noeuds à plus haute capacité peuvent servir multiples voisins, les mettre près de la source permet de propager les nouveaux morceaux plus rapidement et vers un plus grand nombre de noeuds. Cela implique d'avoir des arbres de diffusion (qui peuvent changer pour différents morceaux) qui ont un majeur "fan-out" et une profondeur réduite.

Table 6: Chunk scheduler policies

Scheduler	Description		
ru/r [17]	Random useful chunk	/	Random peer
lu/r [17]	Latest useful chunk	/	Random peer
lu/la [12]	Latest useful chunk	/	Latency-aware peer
lu/ba [104]	Latest useful chunk	/	Bandwidth-aware peer
lu/pa [93]	Latest useful chunk	/	Power-aware peer

Nous remarquons que la gestion dynamique de la topologie est une approche raisonnable : en fait, en considérant une topologie $G(n, M)$ ou $G(n, p)$ à $t = 0$, elle modèle approximativement un système dans lequel les noeuds qui entrent dans un swarm reçoivent un petit nombre de noeuds de bootstrap (e.g., par moyen d'un tracker à la BitTorrent) qui constitue le voisinage initial, qui peut en suite être modifié continuellement (par exemple comme le mécanisme PEX de BitTorrent). Nous soulignons que, en accord avec le chapitre 3, les applications sur internet peuvent exhiber un comportement plus proche de (i) comme TVAnts et Joost, ou (ii) comme PPLive et SOPCast, ce qui rend les deux cas importants. Nous soulignons aussi que d'autres types de graphes peuvent être utilisés pour (i), comme Barabasi-Albert [11] scale-free et Watts-Strogatz [117] small-world, mais cela n'ajouterait pas de réalisme à notre campagne expérimentale car la dynamique des topologies est beaucoup plus importante que les conditions initiales au temps $t = 0$.

Chunk Scheduler Le but final de chaque application P2P est de donner à chaque peer un flux continu de données vidéo : pour cela, les peers doivent éviter d'avoir des trous dans leur buffer-maps dans la position proche de la limite de jeu. Le procès d'échange vidéo est géré par un chunk-scheduler, qui agit dès que le peer peut utiliser son débit sortant. Dans les systèmes push, n'importe quel peer p exécute un scheduler qui doit choisir : (i) un morceau parmi ceux stockés dans la buffer-map $B(p)$ et (ii) choisir un noeud-destination parmi ses voisins $N(p)$.

Les algorithmes de scheduling peuvent être divisés en deux classes, selon l'ordre avec lequel la sélection chunk/peer est faite : dans ce travail, nous nous focalisons sur les algorithmes qui d'abord choisissent les chunks et après les peers. Nous considérons les algorithmes proposés en [12, 17, 93, 104] que nous résumons en table 6. En suivant librement [17], nous décrivons chaque algorithme comme *c/p*, ou *c* et *p* signifient respectivement algorithme de sélection de *chunk* et *peer*.

Le scheduler le plus simple est ru/r, qui sélectionne un chunk *random* $c \in B(p)$ que est ensuite envoyé vers un random *useful* peer $p' \in N(p)$, i.e., un noeud qui ne possède pas le chunk sélectionné $c \notin B(p')$. Nous considérons après une série de schedulers qui sélectionnent le *dernier* chunk parmi leur buffer-maps, que ensuite ils envoient vers un "useful" noeud choisi avec une stratégie soit aléatoire lu/r [17], soit *network aware* $lu/\{la, ba, pa\}$. Pour ce qui concerne les stratégies NA, nous considérons une stratégie consciente de la latence lu/la [12], du débit lu/ba [104] et consciente du "pouvoir" lu/pa [93] (i.e., ou le pouvoir est le ratio du débit sur la latence B/L). La sélection est accomplie en mesurant les propriétés de chaque peer, qui sont ensuite ordonnés par la valeur de la propriété et sélectionnés aléatoirement (i.e., pas dans l'ordre strict), avec une probabilité proportionelle à la valeur de la propriété.

Intuitivement, lu/r vise à maintenir le décalage entre source et destinations le plus bas

possible en diffusant le morceau le plus récent (i.e., le dernier morceau dans la buffer-map $B(p)$). Nous considérons le simple ru/r à raison de référence, et lu/r comme il a été prouvé optimal dans des scénarios homogènes [17]. Schedulers NA $lu/\{la, ba, pa\}$ [12, 93, 104] sont, par contre, censés améliorer la performance de lu/r, spécialement dans le cas de scénarios hétérogènes : en détail, lu/la vise à confiner le trafic localement par une sélection de proximité, lu/ba vise à réduire le temps de diffusion des morceaux en préférant des peers avec des hautes capacités d'upload et lu/pa vise à combiner le deux avantages.

Intéraction L3/L7 Finalement, l'efficacité des décisions de scheduling est potentiellement perturbée par des erreurs concernant (i) la précision des mesures du réseau ou (ii) le fonctionnement de l'échange des informations de contrôle.

D'un côté, (i) NA schedulers basent leur décisions sur des propriétés concernant les voisins et potentiellement les conditions du réseau au dessous. Cette propriétés peuvent être récupérées par l'utilisation "d'oracles" (comme proposé par le projet ALTO [6]), ou directement mesurées par l'application. Mesures directes peuvent être plutôt imprécises pour plusieurs raisons (e.g., trafic de fond, OS scheduling, NICs interrupt coalescing, interaction entre plusieurs mesures), et ça peut causer des mauvaised représentations du voisinage et mauvaises décisions de scheduling. Afin de mesurer l'impact des erreurs de mesure sans être spécifiquement lié a des techniques particulières, nous nous sommes résolu à utiliser un modèle de haut-niveau ou le procès de mesure est contrôlé par un seul paramètre α qui décrit l'ampleur de l'erreur.

De l'autre coté, (ii) l'information de contrôle peut ne pas être distribué dans les délais, ou même perdue at L3. En fait, en cas de algorithmes de gossiping utilisant UDP, cette information ne serait pas retransmise, déformant ainsi la vision que le peer a de l'état du système. L'inconsistance peut aussi être due à une dissémination lente des informations de contrôle (e.g., un système pourrait désirer de limiter le débit de son flux de contrôle en limitant la fréquence des message de mise a jour des buffer-maps). Si nous considérons des systèmes mesh-push, le deux types d'erreurs se traduisent en une connaissance périmé de l'état des buffer-maps du voisinage : dans ce cas, un peer peut décider de programmer une transmission d'un morceau vidéo même si la destination a déjà reçu le morceau, résultant ainsi une transmission inutile (chunk collision). Afin de mesurer l'impact de la signalisation sans être lié à des algorithmes spécifiques, nous utilisons encore une abstraction de haut niveau et nous modelons les erreurs provoqué par la perte des paquets ou informations périmes de l'état du système comme des erreurs des buffer-maps.

Analyse Emulative

Introduction

Dans cette section nous présentons ModelNet-TE, une extension de ModelNet qui habilite l'émulation de l'ingénierie du trafic. En outre, nous portons la version originale du cœur ModelNet de BSD à Linux, et nous le rendons disponible à la communauté scientifique [70]. L'outil ModelNet-TE est interoperable, flexible et passe à l'échelle. L'interoperabilité e le passage à l'échelle sont directement hérédité par le code original de ModelNet qui permet de exécuter des milliers d'instances d'applications. La flexibilité, au contraire, est une clé de ModelNet-TE que nous avons conçu pour être un outil réutilisable, dans laquelle les chercheurs peuvent facilement intégrer leur propre mécanismes de TE à coté de ceux que nous fournissons déjà [31, 72].
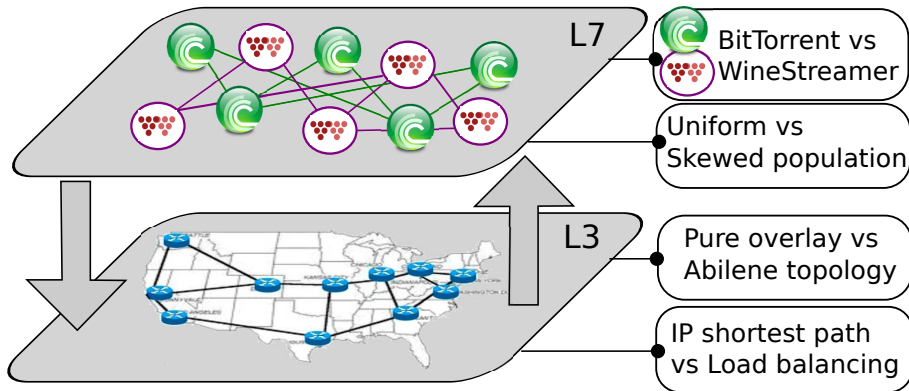
Figure 7: Sommaire des éléments pris en compte dans cette section

Nous utilisons ModelNet-TE pour évaluer l'interaction non coordonné entre le trafic engineering au niveau 3 et le polices du contrôle du trafic appliqués par les applications au niveau 7. En fait, malgré un nombre de travaux ait étudié le problème du "selfish routing", [45,47,58,75,90,99] la plupart de ces travaux adoptent une approche théorique, qui est vrai spécialement pour le cas de interaction non contrôlé de dynamiques de routage à plusieurs niveaux [45,47,58]. De l'autre coté, même si plusieurs études expérimentaux existent [16,29,87,91,101,106,123] néanmoins ils négligent l'interaction avec le niveau réseau dessous. Si leur approche est nécessaire pour comprendre le comportement des applications, il ne permet pas de comprendre l'impact du traffic engineering (TE) sur le trafic des utilisateurs; ni il permet aux développeurs P2P du comprendre comment leur algorithmes performent sur des réseaux réactives.

En voulant remplir cet écart, nous étudions l'interaction L3/L7 à travers ModelNet-TE. Pour prouver la flexibilité de ModelNet-TE, et pour recueillir un ensemble complet de résultats, nous conduisons une campagne expérimentale qui, comme montré en figure 7, considère un riche ensemble de (i) topologie niveau 3 et algorithmes de routage et (ii) applications niveau 7 e modèles de population. Au niveau 3, nous considérons soit un simple modèle de overlay pur, ou le bottleneck est seulement à l'accès, soit Abilene, une topologie populaire, ou est le cœur qui se comporte de bottleneck. Concernant l'ingénierie du trafic L3, nous implémentons un algorithme de partage de charge multi-chemin [31] que nous comparons avec le techniques standard du chemin le plus court. Au niveau 7, nous considérons deux applications réactives : BitTorrent [15], l'application de partage de fichiers la plus connue, et WineStreamer [14, 55], une application de streaming en direct. En plus, nous considérons soit une population uniforme, soit une population qui est proportionelle aux habitants de chaque ville.

En résumé, cette section atteint deux principales résultats. Premier, nous offrons à la communauté scientifique un émulateur de réseau complet, open source, customizable, avec la capacité de faire du TE en directe. Deuxième, nous conduisons la première campagne approfondie, exploitant une méthodologie expérimentale, qui se focalise sur l'interaction entre le dynamiques P2P et le réseau L3 au dessous. Les résultats expérimentaux donnent le suivants aperçus : (i) les bottlenecks dans le réseau peuvent avoir un impact profond sur la performance des applications; (ii) le modèle de population, outre à modéliser le trafic perçu par le réseau, peut contribuer à déterminer la performance; (iii) TE peut améliorer la performance du réseau (e.g., en égalisant la charge sur les liens) au détriment des performance au niveau utilisateur (e.g., du aux effets des interactions inattendues avec TCP ou la logique d'échange P2P).
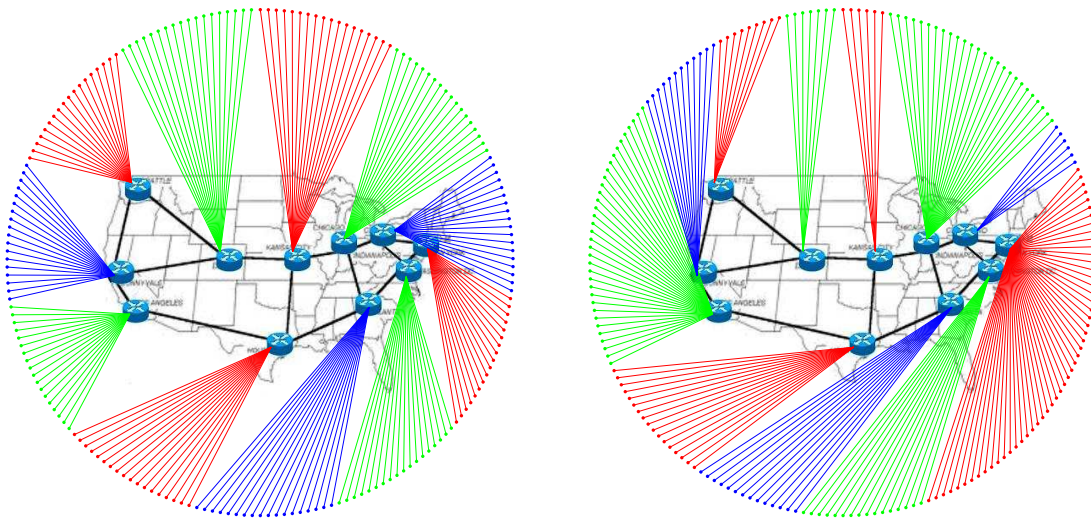
Figure 8: Topologie Abilene de niveau 3 et swarm niveau 7 : modèle de population uniforme à gauche et biaisé à droite

Scénario et méthodologie

Nous décrivons maintenant les scénarios émulés dans notre campagne expérimentale, en donnant le motivations et informations détaillés concernant nôtres choix de (i) topologie réseau, (ii) algorithmes de TE, (iii) modèles de population, (iv) applications P2P.

Réseau niveau 3

Topologie Sans considérer l'application P2P, nous considérons deux différentes topologies de réseau : (i) une topologie réaliste Abilene et (ii) un modelé de overlay pur.

Le scénario réaliste que nous concevons est représenté en figure 8 avec de lien de cœur interconnectés selon la topologie Abilene [1] qui comprend $N_R = 11$ routers déployés sur le sol des États-Unis. Dans notre setup, nous considérons le capacités de liens du cœur de $C = \{5, 10\}$ Mbps et nous modélisons la capacité d'accès des peers similaire à la FTTH (fiber to the home) avec $C_{D,i} = C_{U,i} = 5$Mbps. Il est important de mentionner que le trafic agrégé de chaque peer est en moyenne entre 720Kbps (WineStreamer) et 1.5Mbps (BitTorrent), et que comme une partie du trafic est dirigé vers des peers qui sont derrières la même passerelle, il ne consumera pas la capacité entière du lien cœur. Donc, si $C = 5$ Mbps représente un scénario pauvre, le scénario avec $C = 10$ Mbps modèle un réseau bien provisionné. Remarquez aussi que, malgré réaliste, la topologie Abilene est aussi un scénario difficile pour la répartition de la charge, car le niveau de diversité des chemins ne pourrait toujours permettre de contourner les lien congestionnés.

Pour mieux apprécier l'impact de la topologie du réseau, nous comparons le scénario Abilene avec un modèle simplifié ou tous le peers sont interconnectés avec une topologie étoile vers un router central. Ni les capacités ni le délai sont émulé dans le cœur mais que à l'accès : donc, du a notre setup physique, le cœur va à la même vitesse de notre 1/,Gbps ethernet switch (qui est plus vite du cas Abilene et ou il n'y a jamais des congestions). Toutefois, dans ce scénario nous appliquons latences d'accès réalistes, dépendantes du modèle de population.

Ingenierie du trafic Dans ModelNet-TE, la matrice de trafic (TM) est échantillonnée sur des fenêtres de w secondes ($w = 1$ dans notre cas), et ModelNet-TE peut faire de simples opérations (e.g., moyenne, std deviation, maximum, etc.) sur W fenêtres consécutives. Puis, après W fenêtres consécutives, ces demandes sont exportées du kernel vers les algorithmes de TE. Pour la campagne expérimentale, nous fixons $W = 30$, and exécutons iAWM périodiquement après W fenêtres, pour générer le nouvelles table de routage.

Applications P2P niveau 7

Au niveau 7 , nous construisons scénarios réalistes en considérant hétérogénéité dans (i) la classe des applications P2P et (ii) modèles de population

Nous sélectionnons deux applications P2P, BitTorrent [15] et WineStreamer [14, 55], qui offrent services hétérogènes et ils ont ainsi des différentes architectures. En fait, BitTorrent et WineStreamer sont plutôt différents dans leur contraintes, choix architecturales (TCP vs UDP) et logique d'échange (rarest first VS playout-deadline). Pourtant, ces applications partagent aussi des similarités (mesh overlay) qui sont un résultat naturel de l'évolution de l'écosystème du P2P.

Pour les deux applications, nous émulons un scénario de flash-crowd dans lequel une source unique injecte la vidéo pour un ensemble de $N_p = 200$ noeuds. Notez que cela est une taille raisonnable pour des applications de file-sharing: en fait, [126] observe que juste le 1% des torrents ont plus de 100 peers, alors que [84] report des tailles typiques de swarms de BitTorrent entre 300 et 800 peers. Concernant le streaming en directe, dans le chapitre 3 nous avons observé que la taille des swarms pour une chaine dépende de l'application (qui reflet la popularité de l'application plutot que la popularité de la chaine), avec des swarms qui vont de 500 peers en TVAnts jusu'à 180,000 peers pour PPLive pour le contenus les plus populaires. Donc, $N_p = 200$ peers peut représenter une chaîne populaire sur une application moyennement populaire, ou un contenu moyennement populaire sur une application populaire.

Pour simplicité, nous considérons les capacités des peers homogènes : notez que l'effet des swarms hétérogènes avec plusieurs capacité sont connu [53] par un point de vue purement applicatif, et pourrait être digne de investigation vue par une interaction L3/L7.

Modèle de population La différence entre le modèle uniforme et proportionnel est représenté graphiquement en figure 8, et la correspondante distribution des latences en figure 9. La distribution des latences montre l'impact des la population biaisé et trois pics se produisent : ils correspondent a (i) bas délai pour les communications proches (derrière la même passerelle), (ii) délai modéré pour communications mi-porté et (iii) pour les communications loin (entre le deux cotes). Notez aussi que le pic à haut délai est prononcé, car la plupart de la population est distribuée sur le deux cotes. Au contraire, la distribution uniforme porte à une distribution complètement différente, qui est pas réaliste par rapport aux mesures faites par des projets comme [120].

Résultats expérimentaux

Dans cette section nous reportons des résultats de notre campagne expérimentale, pour raisons d'espace nous montrons ici seulement l'impact des modelés de population et de la capacité du réseau.
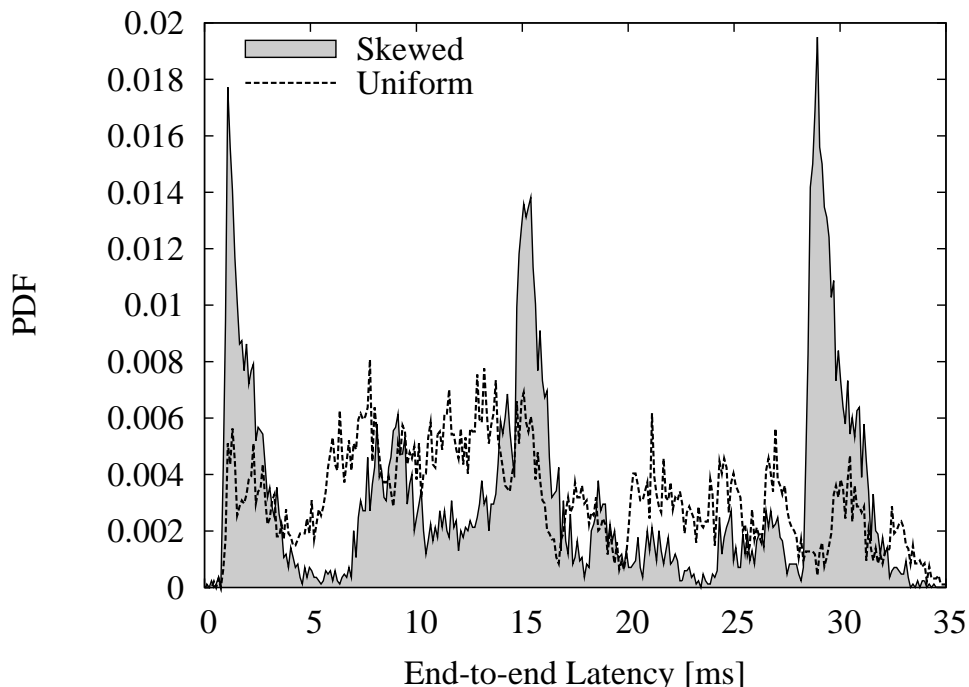
Figure 9: Modèle de population uniforme contre biaisé : distribution des latences

Impact du modèle de population et de la capacité du réseau

Nous nous focalisons maintenant sur la performance des applications niveau 7, en considérant le débit de téléchargement des peers BitTorrent et (ii) la pourcentage de morceaux correctement reçus pour WineStreamer. Nous pensons que ces métriques sont les meilleures pour représenter la qualité aperçue par les utilisateurs : en fat (i) est lié à l'efficience du système et au temps pour compléter le téléchargement ; concernant (ii), la qualité vidéo est baissé par les morceaux qui arrivent après la limite de temps ou qui sont perdus. Les deux métriques sont évalués sur des fenêtres de 10 secondes. Dans la suite nous reportons les résultats collectés en 5 expériences pour chaque configuration.

Nous considérons d'abord l'impact que la capacité C des lien du cœur et le modèle de population ont, et nous montrons la CDF (cumulative distribution function) du débit e de la vitesse de réception des morceaux, mesurés pour toute la population en figure 10 (pour l'instant nous utilisons la topologie Abilene).

Nous considérons d'abord la distribution de la population. La considération générale est que la population biaisé est bénéfique car, si l'application tiens en compte la latence ou la capacité, [5], elle peut établir relations de voisinage avec noeuds qui sont attaché a la même passerelle, en confinant ainsi le trafic au bord du réseau et en évitant les étroits liens du cœur.

En se focalisant sur les clients BitTorrent, nous voyons que les débit plus bas sont atteint par les noeuds du scénario uniforme à $C = 5$Mbps: après, nous trouvons deux scénarios dont les performances sont presque équivalentes. Elle peuvent être atteinte par, soit (i) doublant la capacité

---

[5]Comme TCP est avantagé par des RTT petits, le applications qui préfèrent noeuds à haute capacité, préférerons aussi probablement les noeuds proches. Même pour des applications comme PPLive, qui utilisent UDP au niveau 4 et mesurent le débit au niveau 7, nous avons vérifié expérimentalement que la préférence vers la capacité induit une agrégation des noeuds proches (voir chapitre [97])
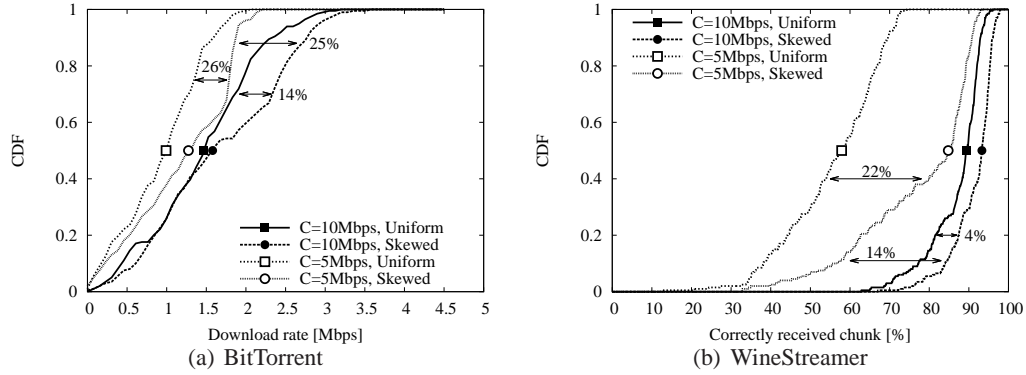
Figure 10: Impact des différents modèles de population (i) et des capacités des liens du cœur ($C = 5$Mbps VS $C = 10$Mbps). Les flèches sont utilisées pour souligner les gains de performance sur la moyenne de la distribution

sous le même modèle de population ou (ii) considérant un modèle biaisé avec la même capacité. Notez en fait que le débit entrant moyen augmente du 25% et 26% respectivement, comme reporté dans la figure 10(a).

Si nous considérons les clients WineStreamer, nous voyons que l'impact du modèle de population reste considérable, bien que dans ce cas les capacités des liens du cœur jouent un rôle déterminant à cause des contraintes du streaming. En fait, si nous considérons le scénario à basse capacité $C = 5$Mbps de la figure 10(b), en moyenne le 22% en plus des morceaux sont reçus dans le modèle de population biaisé par rapport au modèle uniforme. Pourtant, changement du modèle de population ne suffisent pas, car la pourcentage des morceaux reçus pour le scénario à $C = 5$Mbps est encore basse pour certains noeuds (particulièrement pour ceux qui sont derrière à liens très chargé), alors que la situation s'améliore nettement pour le scénario à $C = 10$Mbps.

Contents

Chapter 1

Introduction

1.1 Foreword

At the beginning of this thesis, peer-to-peer paradigm was becoming more and more popular as new services were deployed in the internet. According to [43], in 2008 around 70% of the traffic on European Internet backbones was generated by P2P applications. However, for the time being, the promised breakthrough of P2P-TV did not maintain itself: Cisco Visual Networking Index 2010-2012 [21] affirms that streamed video is surpassing P2P traffic and that video directly streamed to enabled-televisions[1], through OTT (Over-the-top) technologies, will be in the next years the fastest growing video service together with mobile video. Our opinion is that this is mainly due to ADSL limitation (i.e. P2P systems do not have enough upload bandwidth to sustain the service) but it can change with Fiber-to-the-home FTTH in the near future. Hence, this is the reason to sustain the importance of this study.

The motivations behind the initial growth of P2P-TV systems are manifold: first, great part of the intelligence required by a P2P system is at the network's edge so that it is not necessary to upgrade its infrastructure to deploy a new application/service. Second important aspect is that there are no Single points of Failure: overlays are built in a distributed fashion and do not require centralized equipment to coordinate the exchange of information. Third, due to its decentralized nature, it is difficult to have a control over data traffic (e.g., tracking illegal contents) and this allowed file-sharing systems to gain more and more popularity. [21] claims that approximately 70 to 80 percent of P2P traffic in 2010 was carrying video content.

Nowadays, P2P paradigm spreads over a wide range of network-services, from distributed computation to distributed file systems; it is also affecting the way video content is delivered. Operators offering the so-called *triple play* [2] are currently relying on multicast technology which, if on one hand guarantees optimal performance, on the other lets video traffic only to be distributed into the single AS. Video-P2P technologies, while keeping an high level of efficiency, could remedy to this limitation and allow distributors to build world-wide video-distribution markets.

Well-know live video systems such as `PPLive` [89], `TVAnts` [112], `SopCast` [107] are already attracting a huge number of users. `PPLive` claims to have 200 million user installations and 104 million active users each month in 2011 [109]. Such huge numbers added to the intrinsic characteristic of modern P2P overlays[3] are worrying network operators and internet service providers which are beginning to adopt strategies to face P2P phenomenon. Comcast, a US provider, [23]

---

[1] by means of set-top boxes, Internet enables devices as Microsoft XBox 360 or Internet-enabled televisions

[2] High bandwidth internet connectivity, television and telephone services

[3] High number of connection, streams with high bandwidth demand, not optimized traffic, etc,

is the most famous case and it recently admitted to handle differently P2P from traditional traffic; moreover, some studies have shown that P2P traffic daily trend is inverted from daily traffic. This fact seems to be due to the effect of traffic manipulation from internet providers [44].

The challenge therefore is to make P2P systems, and especially live-video P2P systems, behaving friendly toward the network layer and also to make possible the cooperation with service providers. This fact can lead to advantages for both users and service providers: a better engineered P2P traffic can drastically improve the user experience (e.g., lower delays and higher perceived quality) while containing operators' costs. In fact, if a P2P node is able to chose neighbors in the same autonomous system, traffic generated will not transit through expensive access links. We call this ability *network awareness*.

1.1.1   Network Awareness

The definition of *network awareness* is straightforward: A P2P application is network-aware if has some knowledge about the underlay network and uses this information for its inner algorithms (i.e. chunk scheduling or topology management).

Intuitively this information can answer to the questions: is there any peer i can contact in my Autonomous System? How long is the round trip time toward that peer? How many IP hops packets must pass through? A partial role of this thesis work is to individuate which network properties can be taken into account to improve P2P systems.

Approach	Work	Year	Topology Navigation
P2P Measurements	[57]	2009	Latency
	[104]	2008	Bandwidth
	[93]	2008	$\frac{Bandwidth}{Latency}$
	[12]	2006	Locality
ISP Cooperation	[37]		Oracle (IETF ALTO WG)
	[3]	2007	Oracle
	[124]	2008	iTracker

Table 1.1: An overview of Network-Aware approaches

The ways these data are collected can vary across systems and Table 1.1 summarizes main efforts in this field. Work is divided in two big families, P2P measurements and ISP cooperation. With former techniques applications can perform measures on their own (e.g., use tools as `ping` to carry out latency measure or *CapProbe* [46] for path capacity estimations) without the need of external supporting infrastructure. However, while measuring latency, hop distance, loss rate or throughput measurement is trivial, other metric such as path capacity or available bandwidth can be tricky. Furthermore those measure usually are achieved by injecting additional traffic in the network which in turn can bias the measure itself resulting in imprecise values. Finally, techniques are generally devised to single measurements, so that multiple measurements at the same time interfere leading to wrong awareness of the network status. The second group of techniques leverages the help of ISPs which can deploy so-called *oracle* nodes in strategic point of the network. Oracles expose query interfaces to allow the exchange of information with the P2P swarm to help nodes choosing the best neighbors according to certain criteria. One positive aspect of this solution is the optimal topology knowledge of ISPs which may also potentially exploit oracles to do some basic traffic engineering. In this thesis, we focus on P2P measurement class.

1.2 The Big Picture

With reference to Figure 1.1, we now describe the content and organization of this thesis.
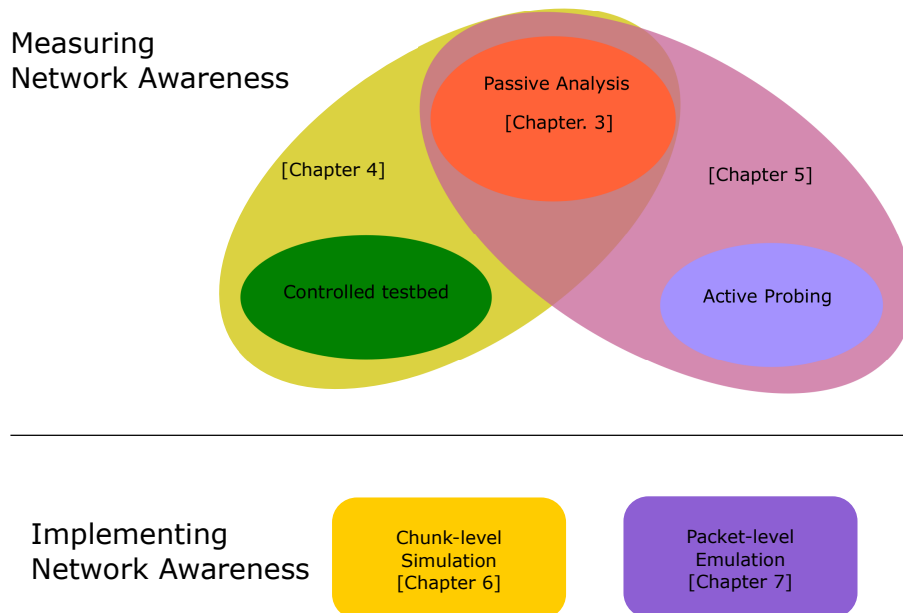
Figure 1.1: The big picture

1.2.1 Measuring Network-Awareness

One of the preliminary steps in the context of the *NapaWine* project [73] is to get an overview on the actual state-of-the-art of most important deployed P2P systems (namely `PPLive`, `TVAnts`, `SopCast` and others). Specifically, our aim is to assess the level of network awareness embedded in these applications. The major obstacle is that these massively-used systems are closed-source and we have no information about either their inner algorithms or protocols they use for data exchange. Therefore the first challenge is to find a methodology that enables the evaluation; we dropped the reverse engineering approach since it entails high costs and is hence applicable only to few applications/versions. Consequently we developed methodologies that use a *black-box* approach and that are usable to examine every present and future system that follows the P2P paradigm. Next section will explain in more detail our achievements.

Passive Approach

First work in this direction is presented in Chapter 3 in which we set-up a pan-European measure campaign among project partners to run unmodified and undisturbed P2P clients in order to collect traces from a certain number of vantage points. We then define a framework to quantify which network parameters influence the application choices and we then apply the methodology to the dataset.

Here the main idea is to use *Preferential Partitions* (PP): each peer in our testbed contacted a number of other peers in the rest of the world. Let $\mathcal{P}(p)$ denote the set of peers contacted by the peer p. Then, given a network property $X(\cdot)$, we divide $\mathcal{P}(p)$ into two sub-set using the value of $X(\cdot)$ such that one class should intuitively be preferred from the application (e.g., if we use

`RTT` as a property, we divide *close* Vs. *far* peers). At this point we compute the number of bytes that are exchanged with peers in the preferred partition over the total. Intuitively the greater this number is, the greater the bias w.r.t. the preferential partition is.

From the analysis of the dataset we observe that `TVAnts` and `PPLive` clients mildly prefer to exchange data in their own Autonomous System. In `SopCast`, instead, this clustering effect is less intense. However, in all cases, no preference versus country, subnet or hop count is shown.

Active Approach

The main drawback of purely passive analysis is that only *peer-wise* metrics[4] can be analyzed. Conversely, path-wise metrics (i.e. properties related to paths), are instead difficult to infer by simple analysis of traces, since we do not know the details of the protocols used by applications. To overcome this limitation, it is necessary to use an active approach as the one described in Chapter 4 in which we setup a completely controlled environment to test the behavior of applications.

Here, we force application instances to download the video stream from a controlled source, then we start to change path properties (e.g. capacity, latency, loss rate or number of IP hops, etc.) and we observe how the application reacts to these alterations.

We apply this methodology to `PPLive` and we investigate which path-wise property biases more its trading preferences. Our main finding is that `PPLive` seems mainly bandwidth greedy, but does not show any preference toward peer proximity based on RTT delay.

Again, with a purely active approach we can evaluate the awareness of an application towards some network properties but we are not able yet to draw a comprehensive picture. We need indeed the contribution of both techniques. Correlating the above findings with the offline analysis of passive traces, we gather that bandwidth preference alone may provide a non-negligible level of geographical clustering among peers as a beneficial side-effect.

Passive Analysis augmented with Active Probing

Finally, as we just saw, active and passive methods alone can not contribute in having a global overview of the applications' network-awareness: for this reason we exploit both methodologies at the same time. In Chapter 5 we present a comprehensive framework named Sherlock to (i) analyze P2P applications as black-boxes and (ii) to compactly describe the traffic they generate. Sherlock can collect information about an application by means of passive and active measures and show at a glance its distinctive fingerprint in form of *kiviat charts*.

We implement the Sherlock framework in the demonstration software *P2PGauge* we presented at *SIGCOMM09* [94] and which we release as open source in [76] building a tool able to measure the level of network awareness embedded in current P2P applications by correlating passive and active techniques. Results of our investigation are applied to the analysis of the well-known P2P-TV application `SopCast`.

1.2.2   Implementing Network Awareness

In the second part of the thesis, we move from the analysis of existent P2P-TV systems to original designs and algorithms and we develop methodologies and tools to test them under realistic conditions.

---

[4]properties related to a single peer and not to the path (e.g., Autonomous System, country, IP address

Simulation-based Study

Together with other NapaWine project partners we developed a chunk-level simulator called P2PTV-Sim which is specifically conceived for P2P television. Its main design goals are (i) being easily customizable and (ii) as realistic as possible. Our aim is to understand if network-aware algorithms are robust to measurement errors and wrong state knowledge or if imprecise information impact negatively on the overlay performance.

In Chapter 6 we carry on a simulation analysis that considers several factors, modeling the L7 overlay (e.g., chunk scheduling, topology management, overlay topology, etc.), the L3 network (e.g., end-to-end latency models, fixed vs dynamic conditions, etc.), and the interaction of both layers (e.g., measurement errors, loss of signaling messages, etc.). To depict a comprehensive system view, results are expressed in terms of both user-centric and network-centric metrics. In a nutshell, our main finding is that P2P-TV systems are generally robust against measurement errors (e.g., propagation delay or capacity estimation), but are on the contrary deeply affected by signaling errors (e.g., loss or outdated system view), which are often overlooked without justification.

Still, despite the invaluable insight a simulator can offer on the inner workings of an overlay, it may not be enough to capture all the potential flaws of a P2P system design: indeed, many implementation details can not be analyzed by means of simulation.

Emulation-based Study

To overcome this limitation and deepen our analysis, we make use of a packet-level network emulator technique. In more detail we consider Modelnet which is an emulation environment that allows the experimentation of real network applications by emulating arbitrary network topologies with latencies, capacities and losses. Building on Modelnet we develop Modelnet-TE, which is able to perform real-time Traffic Engineering (TE).

In Chapter 7 we carry on an experimental campaign of L7/L3 routing layers interaction. As TE algorithm we consider the classic minimum congestion load-balancing, that we compare against standard IP routing. As example P2P applications, we take BitTorrent, one among the most popular file-sharing applications nowadays, and WineStreamer, an open source live-streaming application developed during NapaWine project and available at [119]. We emulate BitTorrent and WineStreamer swarms over both realistic topologies (e.g., Abilene) and simplistic topologies that are commonly in use today (e.g., where the bottleneck is sited at the network edge) under a variety of scenarios.

Results of our experimental campaign show that user-level performance may be significantly affected by both the TE mechanism in use at L3 (e.g., due to interactions with TCP congestion control or P2P chunk trading logic), as well as scenario parameters that are difficult to control in the wild Internet, which thus testifies the interest for tools such as ModelNet-TE.

Part I

Measuring network awareness

Chapter 2

Preliminary discussion

Since the first part of the thesis deals with problematics that are very close, we provide in this chapter contents (related work and a passive dataset collected in the context of NapaWine project [73]) that are common to chapters 3, 4 and 5.

2.1 Related Work

A fairly large number of public P2P architectures and algorithms for the support of video distribution over the Internet has been proposed in the last three-four years within the scientific community [10, 18, 20, 79, 129]. Despite their clear merit, the above systems have only gained limited popularity – especially in comparison with commercial systems. The latter systems have indeed attracted a larger audience, up to several millions of users. The fact that such commercial systems follow a closed and proprietary design has motivated further research [2, 5, 41, 42, 56, 100, 105, 115, 116, 122], aimed at understanding these systems through on-field measurements.

Such works, which exploit partial reverse engineering techniques and typically rely on active crawling methodologies, face the daunting task of understanding and implementing part of the system under analysis. As a consequence, this methodology is limited by the ability to break closed and proprietary systems, and we believe that they can be hardly extended to characterize all the possible P2P-TV applications. For example, [41] investigates PPLive, whereas [56] focuses on the commercial re-engineer of Coolstreaming, and [122] and [121] considers UUSee. For instance, authors in [121] exploit the application-layer logs of UUSee: at the same time, we stress that our work differs from [121] concerning two important points. A first difference arises in the methodology employed, which in the case of [121] limits the applicability of the effort: indeed, authors not only have knowledge of the P2P-TV system inner workings, but also base their analysis on application-layer logs, which requires thus to instrument the application under analysis and is thus clearly not applicable in case of closed-source proprietary systems. Second, our work does not have the same motivation: while authors of [121] want to explore and dig P2P overlays dynamics and graph properties over long period of time, we focus on the network awareness of the applications under study; despite these different interests we arrive to a common conclusion. In fact they say that a significant fraction of neighbor peers falls into the same ISP, despite UUSee does not explicitly take into consideration ISP membership. These observations allows them to conclude that the reason behind the clustering is that "as connections between peers in the same ISPs have generally higher throughput and smaller delay than those across ISPs, they are more inclined to be chosen as active connections". Interestingly, in chapter 4 we show that a similar

geographical clustering can be observed in PPLive, which we also show *not* to be sensitive to RTT preference. To the best of our knowledge, this clustering effect solely induced by bandwidth preference has not been observed by other researchers to date.

Other works, such as [2,42,115,116] instead study specific aspects of a P2P streaming system. For instance, [115] gives some preliminary results on the node degrees of popular versus unpopular channels in PPLive, while [116] instead investigates the stability of PPLive peers. Quality of service is of concern in [2,42]. Authors in [42] exploit an analysis of PPLive buffer maps, collected through protocol reverse engineering, to infer QoS metrics such as network-wide playback continuity, startup latency, playback lags among peers, and chunk propagation timing. Authors in [2] focus on similar metrics but instead exploit logs made available from an unspecified commercial P2P streaming system.

Despite all the above valuable work, to date, very few measurement studies compare different systems, such as [5, 100, 105], which are closest to our work. Authors in [100] analyze and compare PPLive and SopCast, investigating the time evolution of different metrics, like transmitted/received bytes, number of parents and children, etc. Authors in [105], present instead a comparative evaluation of PPLive, PPStream, SOPCast and TVAnts. Analysis is carried on in terms of flow-level scatter plots of mean packet size versus flow duration and data rate of the top-10 contributors versus the overall download rate.

In [5], authors set-up an *active* testbed to investigate the congestion control algorithms of different P2P-TV applications. Using active probes, authors enforce artificial bandwidth limitations, packet loss and delay, and examine P2P-TV reaction to adverse network conditions. However, not all metrics potentially exploited by the overlay for neighbor selection and chunk scheduling can be artificially enforced – as, for instance, it is the case of the geographical and AS location of the contributing peers.

Our work presented in chapter 3 differs from [5, 100, 105] in several aspects. The first is the aim, as our work focuses on a systematical exploration of the metrics, if any, that drive the peer-selection in the different systems. Second, we consider different aspects related to the overlay setup and download policies which are complementary to those addressed in [5, 100, 105]. An important last difference lies on the scale of the testbed used in chapters 3, 4 and presented in 2.2, which involves multiple vantage points scattered across European countries and it is representative of very different network setups.

Inspired by [5], and willing to overcome the problems found in chapter 3, chapter 4 refines their setup, by (i) considering a more controlled setup and by (ii) jointly applying different impairments, so to assess the relative importance of multiple path-wise properties.

2.2   Passive Data Set

In this section we present the data set used in Chapters 3 and 4, whose main features are summarized in Tab. 2.1. Partners of the NAPA-WINE project took part in the experiments by running P2P-TV clients on PCs connected either to their institution LAN, or to home networks having cable/DSL access. In more details, the setup involved a total of 44 peers, including 37 PCs from 7 different industrial/academic sites, and 7 home PCs. PCs are distributed over four countries, and connected to 6 different Autonomous Systems (AS), while home PCs are connected to 7 other ASs. Therefore, the setup is representative of a significant number of different network environments. In the following, we refer to the set of PCs used during the experiment as "NAPA-WINE peers".

In P2P-TV systems, hosts running the application (called peers) form an overlay topology by

Table 2.1: Summary of the hosts, sites, countries (CC), autonomous systems (AS) and access types of the peers involved in the experiments.

Host	Site	CC	AS	Access	NAT	FW
1-4	BME	HU	AS1	high-bw	-	-
5			ASx	DSL 6/0.512	-	-
1-9	PoliTO	IT	AS2	high-bw	-	-
10			ASx	DSL 4/0.384	-	-
11-12			ASx	DSL 8/0.384	Y	-
1-4	MT	HU	AS3	high-bw	-	-
1-3	FFT	FR	AS5	high-bw	-	-
1-4	ENST	FR	AS4	high-bw	-	Y
5			ASx	DSL 22/1.8	Y	-
1-5	UniTN	IT	AS2	high-bw	-	-
6-7				high-bw	Y	-
8			ASx	DSL 2.5/0.384	Y	Y
1-8	WUT	PL	AS6	high-bw	-	-
9			ASx	CATV 6/0.512	-	-

setting up virtual links over which they transmit and receive information. A source peer is responsible to inject the video stream, by chopping it into segments (called chunks) of few kilobytes, which are then sent to a subset of its neighboring peers (called neighbors). Each peer can contribute to the chunk diffusion process, by retransmitting them to its neighbors following a swarming like behavior, as in file sharing P2P systems like BitTorrent. The major differences between P2P-TV systems and traditional P2P file sharing applications are i) that the source is generating the stream in real time, ii) that data must be received by peers at almost constant rate, and iii) that chunks must arrive almost in sequence so that they can be quickly played at the receiver.

We considered three different applications, namely `PPLive`, `SopCast` and `TVAnts`, and we performed several 1-hour long experiments during April 2008, where peers were watching the same channel at the same time. Packet-level traces were collected and later analyzed. Since P2P-TV applications are mostly popular in Asian countries, we tuned each application to CCTV-1 channel during China peak hours [41]. In all cases, the nominal video stream rate was 384kbps, Windows Media 9 Encoder was used, and the video quality perceived by users was very similar across systems. The entire dataset contains more than 120 hours of experiments, corresponding to more than 140M packets. Collected traces are also made available to the research community upon request.

Chapter 3

Passive Analysis

In this chapter, whose results have been published in [22], we gather the level of network awareness of P2P-TV applications by means of a strictly passive analysis of traffic traces generated by P2P-TV applications. Since we do not know any detail about the applications' design and inner workings, our methodology must allow the study of applications as black-boxes. This procedure, in contrast with reverse engineering approaches, permits to analyze any application, even future, who follows the P2P paradigm. We run a large testbed experiments described in section 2.2 which consists in passive traces collected from 40 vantage points scattered across Europe for three different P2P-TV applications, namely PPLive [89], SopCast [107] and TVAnts [112]. By applying the proposed methodology, we highlight which parameters affect the peer selection and data exchange policies. This work arose from collaboration between Politecnico di Torino, Budapest University of Technology and Economics and Télécom-ParisTech in the context of the NapaWine European project. Our contribution to the work has been the development of the preferential partition framework and the analysis of all the traces collected during the experimental campaign.

The rest of this Chapter is organized as follows. First, we present preliminary results in 3.1, where we present a preliminary quantitative description of the performed experiments. We then introduce the methodology for the analysis of the experimental data in 3.2, introducing the metrics that we will use to assess P2P systems network awareness. Experimental results are reported in 3.3, while we devote 3.4 to a spatial and temporal analysis of the peer selection process.

3.1 Preliminary Results

We recall that the description of the dataset used in this section is presented in 2.2 towards which we invite the reader for more details.

Let us now give some preliminary definition. It has been previously observed that P2P-TV peers exchange packets of typical length, i.e., very short packets carrying signaling information, and much longer packets carrying video information [41]. Let $N_v(i, j)$ be the number of packets sent from peer i to peer j whose size is equal to the typical video packet length. To distinguish between peers that exchanged mainly signaling information, and peers that exchanged actual video content too, we say that Peer i is a "TX (transmitting) contributing" peer for j if $N_v(i, j)$ is larger than threshold $M = 5$, i.e., peer i transmitted at least 5 video packets to peer j. At the same time, j is a "RX (receiving) contributing" peer for i, i.e., j received at least 5 video packets from i. We verified that this heuristic gives accurate and conservative results for classifying the contributing peers. Results are also consistent with results of the heuristic presented in [41] in which only

Table 3.1: Mean and coefficient of variation of some statistics collected during all the experiments.

	PPLive		SopCast		TVAnts	
	Mean	cv	Mean	cv	Mean	cv
Avg RX rate [kbps]	549	0.22	482	0.05	415	0.04
Avg TX rate [kbps]	3150	1.02	262	1.04	396	0.65
N. contacted peers	22263	0.53	740	0.29	222	0.19
N. RX contrib peers	382	0.50	139	0.43	54	0.31
N. TX contrib peers	958	0.73	137	0.54	66	0.54
% non-resp. peers	27	0.87	25	0.73	30	0.31

PPLive was analyzed.

We start by giving some preliminary insights from the collected data.

Both the average value over all the NAPA-WINE peers and the coefficient of variation (i.e., $cv = \sqrt{\sigma^2(X)/E^2(X)}$, where $\sigma(X)$ and $E(X)$ are the standard deviation and average of X, respectively) are reported. Tab. 3.1 presents the following simple metrics which are evaluated considering all NAPA-WINE peers: i) receiving data rate, ii) transmitting data rate, iii) number of contacted peers (i.e. the number of peers that successfully exchanged at least one packet), iv) number of RX contributing peers, v) number of TX contributing peers, and vi) percentage of peers that have never replied to any message.

The first and the second rows show the average inbound and outbound data rate, including both video and signaling traffic. As we can expect, on the reception side, no significant difference can be observed among the different applications, as testified by the small coefficients of variation. This is due to the fact that the dominant component of the received traffic is constituted by the video content, whose average rate is the same for all the peers and applications (recall that all the considered applications adopt the same streaming encoding technique). For `PPLive` dataset, the higher cv and average values suggest that the receiver rate can be higher than the stream rate. This is due to the large incoming traffic that high-bandwidth peers receive, i.e., to the signaling messages they have to handle which are sent by the peers receiving the uploaded video content.

More interestingly, the transmission rates are significantly different. Indeed, the transmission data rates are strongly dependent on the specific mechanism adopted by each system to distribute the video content. First, the transmission data rate is largely correlated to the upload bandwidth of peers; all the applications successfully exploit high bandwidth peers, demanding to some of them a significant contribution. To confirm this, we investigated the upload rate of each NAPA-WINE peers and we found that high-bandwidth NAPA-WINE peers are acting as "amplifiers", i.e., they upload much more than what they download. Instead, peers that are connected by CABLE/DSL show much smaller upload rate. On this regard, we observe that `PPLive` may be significantly demanding, so that high bandwidth peers push their average transmission data rate to more than 10Mbit/s, with short time peaks reaching 30Mbit/s. The high coefficient of variation is also due to difference among the upload capacity of the peers.

Huge differences among the systems arise considering the number of contacted peers, which is on the order of tens of thousands for PPLive, up to one thousand for SopCast, and in the order of few hundreds for TVAnts. We believe that the huge difference in the number of contacted peers is mainly due to the algorithms used to discover and to maintain the overlay, on which we will come back later in 3.4.

The fourth and the fifth rows report the number of RX contributing peers and TX contributing peers, respectively. If we compare the number of contacted peers against the number of contribut-
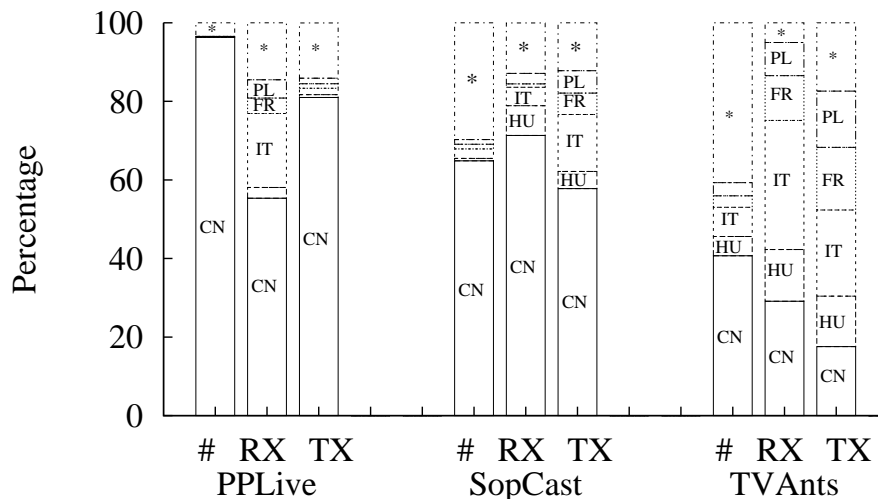
Figure 3.1: Geographical breakdown of the number of peers, received and transmitted bytes.

ing peers, we can see that `TVAnts` exploits the contacted peers at the highest degree, i.e., one fourth of the contacted peers are also contributing peers. Considering `SopCast`, about 1/5 of the contacted peers are used to download some content. Finally, in the `PPLive` case, the number of contacted peers is more than 50 times higher than the number of contributing peers. Focusing on the number of TX contributing peers (i.e., peers to which some content has been transmitted to), we observe that peers with low upload bandwidth serve fewer peers than peers with high bandwidth. This fact is expected since low upload bandwidth peers have limited capability to upload data to other peers. For `PPLive`, the significantly larger number of both contacted and contributing peers explains the higher upload rate.

The last row shows the fraction of the peers which did not reply at all, i.e., failing peers. All of the systems show a very high failing ratio (25% - 30%). This hints to little optimization on the P2P-TV control plane, so that outdated information is still distributed among peers. Moreover, we noticed that in all experiments, NAPA-WINE peers tried to contact peers with private IP address, with some peers performing address scan of whole subnetworks.

Fig. 3.1 shows the geographical distribution of the number of contacted peers, the amount of received and transmitted bytes, labeled #, RX and TX respectively. China (CN) and countries in which experiments were performed are shown, with the rest of the countries labeled with a star. Percentages are expressed over the total number of observed peers, which amounts to 181729 for `PPLive`, 4057 for `SopCast` and 550 for `TVAnts`. As expected, China is the predominant country, though it is easy to gather that a non negligible fraction of the data is exchanged within the countries of NAPA-WINE peers: this hints to a bias in the peer selection, which will be more rigorously investigated in the following sections. The large difference in geographical distribution of contacted peers shows that different algorithms are used by the different applications. In particular, `TVAnts` seems to adopt a "smart" choice in selecting peers. Indeed, among the 550 total peers, 154 are located in Europe and 229 in China. Considering the watched channel and time of the experiment, the popularity of the application should be much higher in China than in Europe, so that we can conclude that the observed peers are only a biased and small fraction of the total active population. On the contrary, `PPLive` adopts a less smart peer discovery policy, so that the total number of contacted peers is more than 50 times higher than `TVAnts` or `SopCast`. In this case, 748 peers are located in Europe only (180.000 in China).

3.2 A Framework for Peer Selection Analysis

As previously stated, our aim is to develop a rigorous framework to unveil the "network-awareness" exhibited by P2P-TV applications, i.e., which network parameters current P2P-TV systems take into account when distributing the stream. We define a flexible framework that allows us not only to inspect the level of "awareness" of a P2P system with respect to the underlying network, but also to assess whether peers behave fairly with respect one to another, i.e. if the peers are incentivized to the mutual data exchange. In particular, we consider:

- **AS**(p): the Autonomous System where peer p is located

- **CC**(p): the Country, which peer p belongs to

- **NET**(p): the subnetwork, which peer p belongs to

- **HOP**(p, e): the IP hop-distance between peers p and e

- **SYM**(p, e): the symmetry of byte-wise data exchanges between peers p and e

3.2.1 Framework Definition

Let $p \in \mathcal{W}$ denote a peer that belongs to the NAPA-WINE set $\mathcal{W}$. Let $\mathcal{P}(p)$ denote the set of *contributing* peers, p exchanges data with. That is, $\mathcal{P}(p)$ is composed by the peers to which p transmitted or/and from which p received some video information. Let $\mathcal{U}(p)$ denote the subset of peers to which p is uploading video content, and $\mathcal{D}(p)$ the subset from which p is downloading video from. $\mathcal{U}(p)$ and $\mathcal{D}(p)$ are two (non disjoint) subsets of $\mathcal{P}(p)$, and $\mathcal{U}(p) \cup \mathcal{D}(p) = \mathcal{P}(p)$.

Let $e \in \mathcal{P}(p)$ be an arbitrary peer that exchanges traffic with p. Denote by $B(p, e)$ the amount of bytes transmitted from p to e, so that $B(e, p)$ represents the amount of bytes received by p from e.

Consider now a generic network parameter $X(\cdot)$, and denote with $X(p, e) \in X$ the observed value of $X(\cdot)$ for the pair (p, e). We partition $\mathcal{P}(p)$ into two classes based on $X(p, e)$, such that one class should intuitively be preferred from the application (e.g., good vs bad peers). More formally, we partition the support X into two disjoint sets: the preferred set X_P and its complement $X_{\overline{P}}$, such that $X_P \cup X_{\overline{P}} = X$ and $X_P \cap X_{\overline{P}} = \emptyset$.

For the ease of notation, let $\mathbf{1}_P(p, e)$ be the identity function which takes the value of 1 if $X(p, e) \in X_P$ and 0 otherwise; similarly, $\mathbf{1}_{\overline{P}}(p, e) = 1 - \mathbf{1}_P(p, e)$. Without loss of generality, let us focus on the upload traffic of a NAPA-WINE peer $p \in \mathcal{W}$, and let us define:

$$Peer_{U|P}(p) = \sum_{e \in \mathcal{U}(p)} \mathbf{1}_P(p, e) \tag{3.1}$$

$$Byte_{U|P}(p) = \sum_{e \in \mathcal{U}(p)} \mathbf{1}_P(p, e) \cdot B(p, e) \tag{3.2}$$

$$Peer_{U|\overline{P}}(p) = \sum_{e \in \mathcal{U}(p)} (1 - \mathbf{1}_P(p, e)) \tag{3.3}$$

$$Byte_{U|\overline{P}}(p) = \sum_{e \in \mathcal{U}(p)} (1 - \mathbf{1}_P(p, e)) \cdot B(p, e) \tag{3.4}$$

where U and D subscripts are used to indicate the upload and download traffic respectively. $Peer_{U|P}(p)$ counts the number of peers of which p is a contributor and which belongs to the preferential partition X_P. Similarly, $Byte_{U|P}(p)$ represents the total amount of bytes uploaded

from peer p to peers in the preferential partition X_P. Conversely, $Peer_{U|\overline{P}}(p)$ and $Byte_{U|\overline{P}}(p)$ represent the number of peers and bytes to which p is uploading despite they belong to the non-preferential partition $X_{\overline{P}}$. Considering now the whole set $\mathcal{W}$ of NAPA-WINE peers, we define the total amount of peers and bytes as:

$$Peer_{U|P} = \sum_{p \in \mathcal{W}} Peer_{U|P}(p) \tag{3.5}$$

$$Byte_{U|P} = \sum_{p \in \mathcal{W}} Byte_{U|P}(p) \tag{3.6}$$

Similar definitions hold for $Peer_{U|\overline{P}}$ and $Byte_{U|\overline{P}}$.

Finally, we define the peer and byte preference as:

$$P_U = 100 \frac{Peer_{U|P}}{Peer_{U|P} + Peer_{U|\overline{P}}} \tag{3.7}$$

$$B_U = 100 \frac{Byte_{U|P}}{Byte_{U|P} + Byte_{U|\overline{P}}} \tag{3.8}$$

Intuitively, P_U expresses the chance that the peer selection mechanism favors the discovery and data exchange among peers belonging to the preferred partition X_P. Similarly, B_U quantifies the chance that any given byte is uploaded to peers belonging to the X_P class. Clearly, the greater P_U and B_U are, the greater the bias with respect to the preferential partition of metric X is. The advantage of using these simple metrics is that they allow a *direct* and *compact* comparison of different network properties and P2P-TV systems, since they are neither sensitive to the unit of measure, nor to the actual value of X.

Downlink metrics P_D and B_D can be defined by considering $e \in \mathcal{D}(p)$ in the previous derivation.

3.2.2 Preferential Partitions

As preferential classes, we consider the following:

- **AS**: $\mathbf{1}_P(p, e) = 1$ if and only if $AS(p) = AS(e)$,
  i.e., both peers are located in the same Autonomous System[1];

- **CC**: $\mathbf{1}_P(p, e) = 1$ if and only if $CC(p) = CC(e)$,
  i.e., both peers are located in the same Country;

- **NET**: $\mathbf{1}_P(p, e) = 1$ if and only if $HOP(e, p) = 0$,
  i.e., peers belong to the same subnet;

- **HOP**: $\mathbf{1}_P(p, e) = 1$ if and only if $HOP(e, p) < median[\text{HOP}]$,
  i.e., the number of hops between e and p is smaller than the median distance among all peers.

- **SYM**: $\mathbf{1}_P(p, e) = 1$ if and only if $1/2 < B(e, p)/B(p, e) < 2$,
  i.e., the amount of data received (sent) is at most twice the amount of data sent (received).

---

[1]CC and AS have been determined by querying the "whois" database.

Table 3.2: NAPA-WINE induced Bias

	Contributors		All-peers	
App	Peer%	Bytes%	Peer%	Bytes%
PPLive	0.95	3.54	0.10	3.33
SopCast	10.25	17.71	4.60	19.45
TVAnts	29.82	56.31	15.56	56.06

While for AS, CC and NET the preferential set choice is straightforward, the HOP and SYM cases require additional discussion. Considering HOP metric first, the hop count $HOP(e, p)$ has been evaluated as 128 minus the TTL of received packets, since 128 is the default TTL considering Windows O.S. We use the median of the distribution as threshold to define two subsets. Since the actual HOP median ranges from 18 to 20 depending on the application, we use a fixed threshold of 19 hops for all applications. This means that, approximately 50% of the peers falls in the preferential class.

In case of incentive mechanism, we classify a data exchange as 'symmetric' when the amount of data received is at most *twice* the amount of data sent, and vice versa. We point out that while this only enforces a loosely symmetrical relationship, we verified that the results are not very sensitive to these threshold choice (see Sec. 3.3.4).

3.2.3 Preliminary Analysis and Issues

Given the black box approach based on passive measurement, several issues could undermine the significance of the results unless carefully dealt with. The first issue is that the NAPA-WINE peers *induced a bias* during the experiments. Recall that among NAPA-WINE peers there are several high-bandwidth peers, located in Europe only. Furthermore, all peers within the same institution are in the same LAN, and AS. This possibly represents an uncommon population subset.

A quantification of the induced bias is given in Tab. 3.2. It reports the percentage of i) NAPA-WINE peers over all peers observed during each experiment, and ii) bytes exchanged among NAPA-WINE peers over all exchanged bytes. Results are reported considering contributors only, or all peers. As first important remark, NAPA-WINE peers clearly prefer to exchange data among them. For example, considering contributors in the PPLive experiment, NAPA-WINE peers contribute to more than 3.5% of exchanged data, even if they represent less than 1% of the contributing peers. Similarly, they are about 10% and 30% of peers for SopCast and TVAnts respectively, but they contribute to 18% and 56% of exchanged bytes. We stress that by restricting the analysis to the set of peers other than NAPA-WINE, it will be possible to highlight and quantify which properties of the NAPA-WINE peers cause such a strong bias. To solve the issue concerning the induced bias, we introduce the set $\mathcal{P}'(p) \subset \mathcal{P}(p)$. Subset $\mathcal{P}'(p)$ is constituted by the peers in $\mathcal{P}(p)$ excluding the NAPA-WINE peers, formally $\mathcal{P}'(p) = \mathcal{P}(p) \setminus \mathcal{W}$. We evaluate the preference metrics also over the filtered set, getting P'_D, P'_U, B'_D, B'_U, accordingly. Intuitively, restricting the observation to $\mathcal{P}'$ is equivalent to consider peers not involved in the experiment. For example, we expect that a preference versus a metric noticed in the full contributor set should be noticeable also in the set deprived of NAPA-WINE peers. In case the bias is still evident, this means that the preference was *not* artificially induced by NAPA-WINE peers.

Another problem concerns the fact that it exists a *correlation* between the considered metrics: for example, peers within the same subnetwork (NET=1) traverse paths of zero hop (HOP=0), belong to the same Autonomous System (AS) and Country (CC) as well. It may be therefore

Table 3.3: Network Awareness as Peer-wise and Byte-wise Bias

| | | Download | | | | Upload | | | |
| | | Non-Napa | | All Contributors | | Non-Napa | | All Contributors | |
Net	App	B'_D %	P'_D %	B_D %	P_D %	B'_U %	P'_U %	B_U %	P_U %
AS	PPLive	6.5	0.6	12.8	1.3	0.8	0.2	1.8	0.5
	SopCast	0.6	0.7	3.5	3.9	1.7	0.7	6.4	3.9
	TVAnts	7.3	3.3	32.0	13.5	11.6	1.8	30.1	9.6
CC	PPLive	6.5	0.6	13.1	1.4	1.1	0.3	2.1	0.6
	SopCast	0.6	0.8	4.0	4.4	1.7	0.8	7.2	4.4
	TVAnts	7.6	4.0	37.9	16.3	14.3	3.1	37.7	12.5
NET	PPLive	-	-	9.9	0.8	-	-	1.4	0.3
	SopCast	-	-	2.0	2.6	-	-	3.5	2.6
	TVAnts	-	-	18.1	6.7	-	-	18.1	5.4
HOP	PPLive	42.2	41.1	51.4	42.4	30.4	40.4	31.7	41.0
	SopCast	29.0	40.7	37.9	48.0	45.9	43.0	56.9	49.8
	TVAnts	62.1	55.0	81.1	71.9	57.8	53.0	78.9	67.2
SYM	PPLive	3.3	4.8	4.3	5.0	-	-	-	-
	SopCast	6.7	13.0	7.8	14.2	-	-	-	-
	TVAnts	12.4	10.9	20.0	14.3	-	-	-	-

difficult to properly isolate the impact of each metric. At the same time, this correlation is likely to hold for the NAPA-WINE peers mainly, since they form "clouds" of high-bandwidth PCs within the same LAN, CC, and AS. Considering the set $\mathcal{P}'$, where the correlation related to the locality among peers is smaller, it will be possible to identify which metric has the highest impact.

All the observed parameters can be evaluated considering separately the download and upload direction of traffic, e.g., we can observe from (to) which countries the NAPA-WINE peers prefer to download (upload) the content. Notice that, for HOP metric, we can only directly measure $HOP(e, p)$, but not $HOP(p, e)$ which can be in general different from $HOP(e, p)$ due to Internet path asymmetry. However, we point out that the adoption of a coarse-granularity should minimize this issue. Indeed, it is likely that $HOP(e, p) \in HOP_P$, then $HOP(p, e) \in HOP_P$ as well, i.e., it is unlikely that the reverse path $HOP(p, e)$ is short when the direct path $HOP(e, p)$ is long. Finally, note that to compute the SYM metric it is necessary to compare the amount of transmitted and received data between any pair of peers.

3.3 Experimental Results

Empirical evaluation of PPLive, SopCast and TVAnts network-awareness is reported in Tab. 3.3. Specifically, we report, for both upload (U) and download (D) directions, the peer-wise (P) and byte-wise (B) preference metrics for each of the different network properties early considered. Tab. 3.3 details results referring to the whole contributor set (P_U, P_D, B_U, B_D) and to the contributor set excluding the NAPA-WINE peers (P'_U, P'_D, B'_U, B'_D).

3.3.1 AS and Country Awareness

We first turn our attention to location awareness by considering the AS and CC metrics. Considering download direction, it can be seen that `SopCast` is unaware of AS location. Indeed, P_D is almost equal to B_D, which suggests that peers in the same AS are not preferentially selected to download data from. On the contrary, both `PPLive` and `TVAnts` show higher AS-awareness. Considering non-NAPA-WINE contributors, a `PPLive` peer downloads from P_D'=0.6% of peers B_D'=6.5% of traffic, i.e., there is a byte preference 10 times larger than a peer preference. The same factor holds including NAPA-WINE peers (which then do not bias the results). Similarly, for `TVAnts`, B_D'=7.6% of the bytes are downloaded from P_D'=3.3% of the non-NAPA-WINE contributors, i.e., a B_D'/P_D' ratio equal to 2. Also, notice that 0.04% of *all* peers are in the same AS of NAPA-WINE peers in case of `PPLive`, and 3.6% in case of `TVAnts`. Still, as 1.3% of the *contributing* peers are located in the same AS for `PPLive`, and 13.5% for `TVAnts`, we can conclude that `PPLive` exhibits a stronger preference for peers within the same AS than `TVAnts`.

Looking at the downloaded traffic with respect to the peer Country, we notice that almost the same percentages are observed as in the AS preference case. Since two peers in the same AS are also located within the same Country, we can conclude that no country preference is shown, i.e., the CC preference is due to the AS preference. Finally, considering the upload directions, similar conclusions can be drawn.

3.3.2 NET Awareness

We now evaluate the potential preference to exchange traffic with peers in the same subnet (NET). The set of peers in the same subnet includes only NAPA-WINE peers, i.e., $\mathcal{P}' = \emptyset$. Results show that also in this case, `PPLive` and `TVAnts` only exhibit NET awareness, for both upload and download directions. Indeed, about 10% and 18% of the bytes are received from about 1% and 7% of hosts which are in the same subnet respectively. Conversely, `SopCast` does not show any evidence of subnet awareness. However, the NET preference can be also enforced by the AS preference. Looking at the ratio between P over B for the AS and NET preferences, we observe that they are very similar. This points out that peers in the same autonomous system but not in the same NET are equally preferred as the peers in the same NET (and in the same AS). Therefore, the AS preference is stronger than the NET preference. Notice also that the AS locality is overall quite marginal, so that the majority of the traffic is still coming from other ASs. As such, there is large margin to improve the network friendliness of P2P-TV applications.

3.3.3 HOP Awareness

We also investigate the HOP count preference. In this case, no particular evidence of preference toward shorter paths is underlined. Indeed, looking at the non-NAPA-WINE peers, almost no difference emerges comparing P' and B'. Only `TVAnts` shows a small preference to download from closer nodes.

To further testify this finding, Fig. 3.2 reports the Cumulative Distribution Function (CDF) of contacted peers (solid line) and of the received bytes (dashed line) versus the distance between peers in hop count, not including the NAPA-WINE peers. `TVAnts` only shows a slight commitment to the closest peers, while `SopCast` and `PPLive` seem to ignore peer distance considering the hop number.
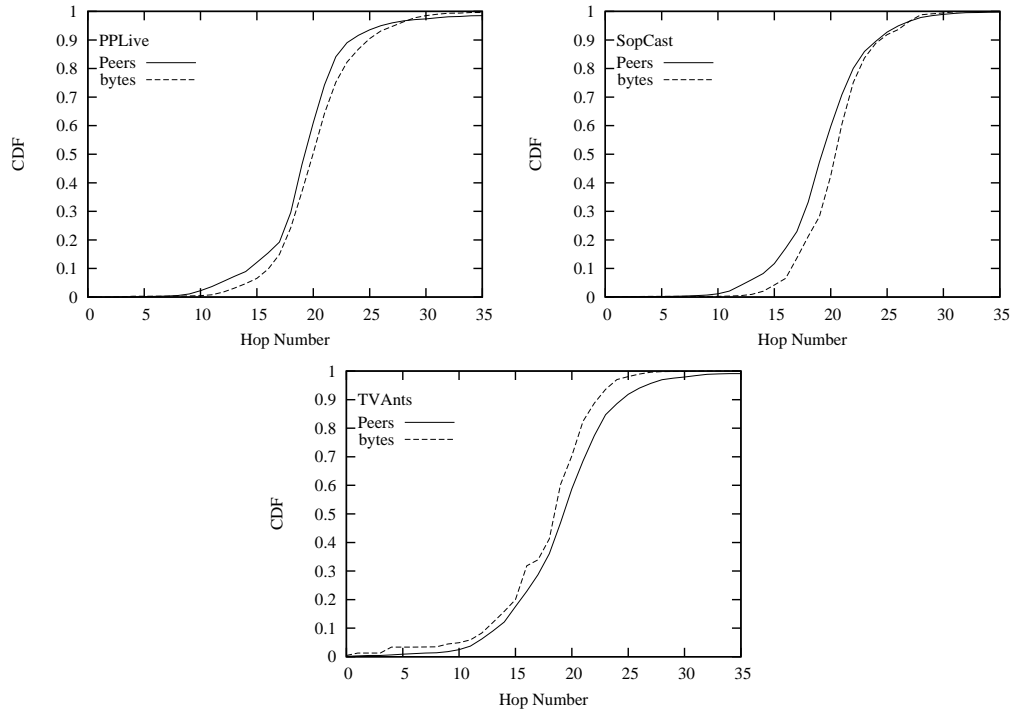
Figure 3.2: CDF of the number of peers and the amount of received data versus the number of hops.
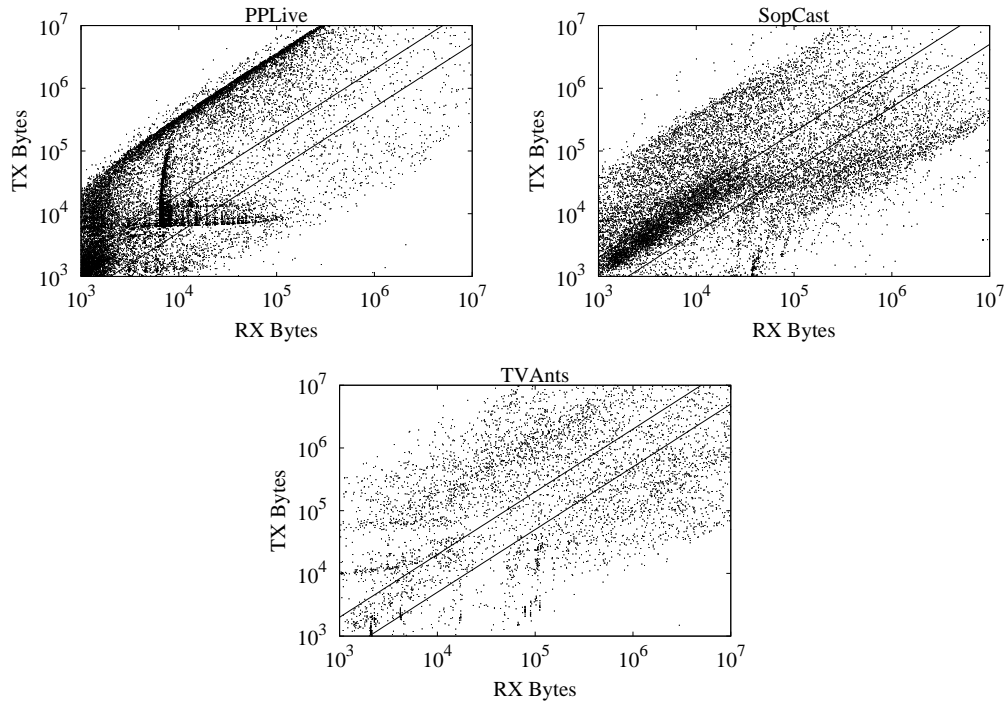

Figure 3.3: Scatter plot of RX versus TX bytes, guidelines represent the tit-for-tat boundaries TX/RX=2 and RX/TX=2

3.3.4 SYM Incentive Mechanism

Considering P2P file sharing applications, incentives mechanisms have been successfully introduced to improve system performance. For example, BitTorrent clients play a tit-for-tat game with other peers, so that the more a peer sends to a neighbor, the more it will receive from it. This enforces a sort of symmetry between the amount of bytes sent and received by peers.

We explore whether there exists some incentive mechanism that enforces symmetry in P2P-TV systems as well. Results are reported in Tab. 3.3: Even if we arbitrarily report SYM under the download section of the table, we recall that it is a metric that requires to compare the amount of traffic exchanged in both directions (upload and download) between two peers. Considering non NAPA-WINE peers, it emerges that only a small percentage (from 5% considering PPLive to 13% considering SopCast) of the links are symmetrical. Moreover, the amount of data exchanged between these peers is not predominant (less than 12%). This suggests that P2P-TV systems do not enforce any tit-for-tat like mechanism. Indeed, being the download rate constrained by the actual video rate, these systems are engineered in such a way that peers with limited upload capacity can receive the video stream anyway, even if they are not able to re-distribute it.

This is highlighted in Fig. 3.3, which reports the amount of transmitted versus received bytes considering contributing peers. Intuitively, if a tit-for-tat like incentive mechanism were implemented, then a strong correlation should be observed so that points accumulate along the $y = x$ diagonal. Log/log scale is used to better represent results. The area between the TX/RX=2 and TX/RX=1/2 lines corresponds to symmetrical exchanges as previously defined. Looking at Fig. 3.3, it can be seen that the wide majority of points fall outside this area, as already reported in Tab. 3.3. Only in the SopCast case, a cloud of points lies in the symmetry strip, though such points correspond to moderate amount of data (i.e., few thousand Bytes). Considering PPLive, we observe that a lot of points accumulate along the $y = 10x$ line, corresponding to peers that mostly download data from the NAPA-WINE peers[2]. The dense points accumulating around $y = 10^4$ and $x = 10^4$ are also a consequence of a private mechanism of the application. Summarizing, no evidence of a symmetric tit-for-tat like incentive emerges for any system.

To summarize, we have shown that the three applications behave differently, and by means of inference on passive measurements, we have empirically quantified these differences. While our results point out the lack of network awareness of such systems, the picture is far from being complete: for instance, the different behaviors are a direct consequence of specific, proprietary and therefore unknown mechanisms adopted by such systems. However, we point out that by pure black-box measurement is unfortunately impossible to understand what are the specific algorithms implemented, as well as the parameters adopted by each system.

3.4 Dynamics of Contacted Peers

In this section we supplement the analysis of peer selection, by inspecting its dynamics from both a temporal and a spatial point of view as well.

3.4.1 Temporal Analysis

To better understand the peer selection process, Fig. 3.4 plots the dynamics of the contacted peers versus time. One arbitrary NAPA-WINE peer is represented in each figure, since they are qualitatively all similar. For PPLive the behavior of both high-bandwidth and DSL nodes are reported.

---

[2]The factor 10 can be explained by considering that PPLive protocol uses some acknowledgment message to possibly confirm the reception of data packets. The size of ACKs is about 1/10th of the size of data packets.
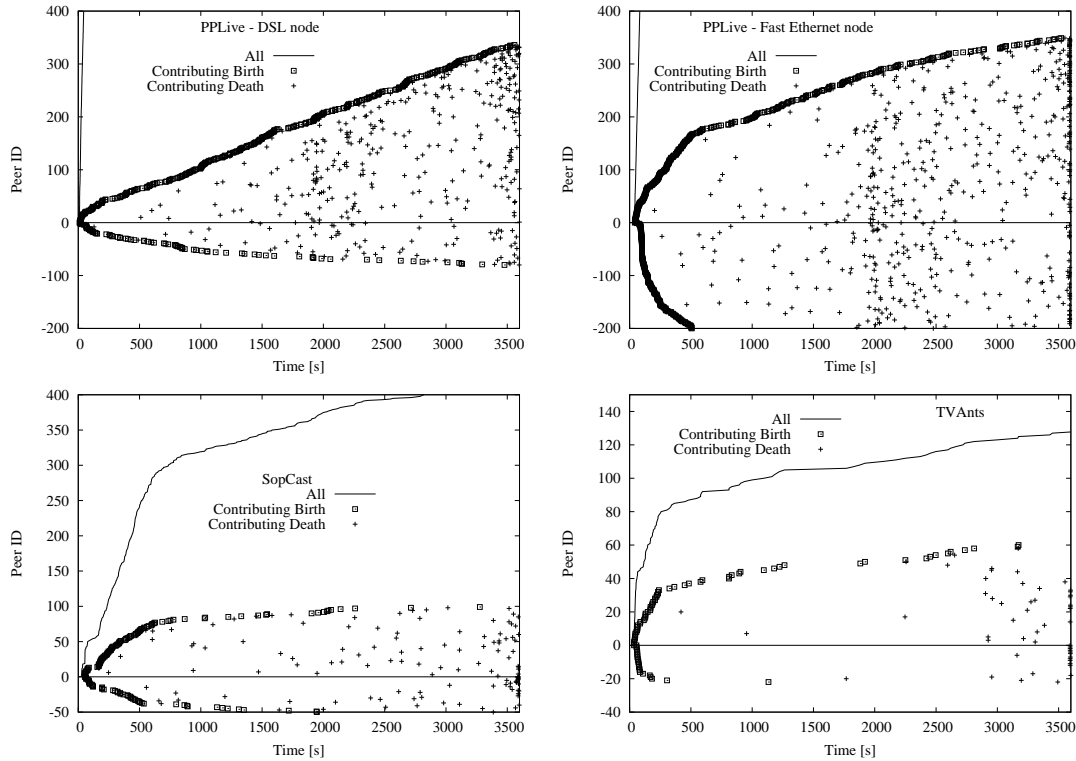
Figure 3.4: Arrival and Departure process of the all and video contributing peers.
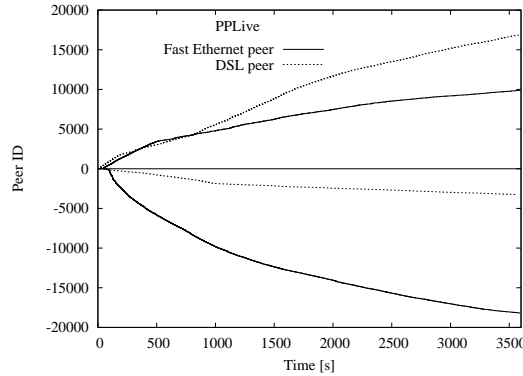
Figure 3.5: Arrival and Departure process of the all peers for `PPLive`.

The continuous line reports the total number of contacted peers versus time, while the squared dots show the arrival of contributing peers, whose departure is shown by the crosses in the same line. In this context, the arrival and the departure of a peer is identified by the time of the first and last observed packet from it, respectively. Positive y-axis reports the remote peers that were contacted by the NAPA-WINE peer first, while negative y-axis reports the peers that were the initiator of the connection toward the NAPA-WINE peer. For `PPLive`, the evolution of the number of contacted peers is reported in Fig. 3.5, since it is much larger larger than other quantities.

Both `TVAnts` and `SopCast` limit this rate as soon as a good set of contributing peers is obtained (after about 250s and 500s respectively). On the contrary, `PPLive` has a stronger greedy behavior, essentially contacting new peers at an almost constant rate. These different overlay

exploration algorithms clearly drive the total number of contacted peers, which is much higher for `PPLive`.

As already observed in Tab. 3.1, the number of contributing peers is limited to few tens for `TVAnts` and `SopCast`. In addition, the set of contributing peers is rather stable along the whole experiment duration, i.e., the contributing peer contact time lasts several tens of minutes. In the case of `PPLive` on the contrary, the number of contributing peers is much higher (several hundreds, up to one thousand) and it exhibits a higher degree of variability. This can be explained considering the fact that the number of possibly good candidates is higher, and the peer selection policy continuously tries to improve performance by testing new peers. No major difference is shown between DSL or high bandwidth nodes considering the number of contributing peers that are contacted by the NAPA-WINEpeer. On the contrary, the number of peers that initiated a connection to the high-bandwidth peer is larger than the one that initiated a connection to the DSL node. That is, the high-bandwidth peer gets more requests. This suggests that the information about the peer upload capacity is made available to other peers.

3.4.2   Spatial Analysis

Finally, we complete our analysis of the peer selection process by considering the spatial properties that can be inferred by exploiting our large number of measurement points. Fig. 3.6 shows the CDF of the common contributing peers, i.e., the probability that a contributing peer is seen by N different NAPA-WINE peers (on the x-axis).

For example, Fig. 3.6 shows that for PPLive, there are 50% of the peers can be observed by only 1 NAPA-WINE user and 70% of the peers can be observed by either 1 or 2 NAPA-WINE users. For SopCast, about 30% of peers are seen by only one NAPA-WINE peer, and 40% as seen by two NAPA-WINE users. These percentages reduces to less than 15% for both cases. Similarly, considering the probability that a peer has been contacted by at least 20 NAPA-WINE peers, we notice that for PPLive, the probability is close to one, meaning that a negligible set of peers have been contacted by more than 20 NAPA-WINE hosts; for SopCast, this probability is 0.8, meaning that there are about 20% of the peers that exchanged data with more that 20 different NAPA-WINE hosts; for TVAnts, there are 50% of the peers that are contacted by more than 20 different NAPA-WINE hosts.

In case a random independent and identically distributed (i.i.d.) selection is performed, the common peers CDF follows a Binomial distribution. On the contrary, in case of a correlated choice (i.e., when certain peers are preferred to other peers), a different trend is expected; for example, a more linear CDF would suggest that peers prefers to contact the same subset of peers.

We assume that the number of contributing peers that exchanged data with NAPA-WINE peers during the experiment is a small fraction of all available peers. This assumption is supported by Fig. 3.1, in which TVAnts population is largely biased, suggesting a "smart" choice by peers. Furthermore, Fig. 3.4 shows that both TVants and Sopcast use a peer discovery mechanisms which is very greedy during the first part of peer life, after which the peer discovery rate slows down.

Fig. 3.6 clearly shows that for `SopCast` and `TVAnts` experiments the selection of peers from which to download is performed according to some algorithm that tends to correlate peer choice.

For example, consider the probability that no more than 20 NAPA-WINE peers select the same contributing peer. For `PPLive`, is almost one, it is about 0.8 for `SopCast`, and it is about 0.5 for `TVAnts`. Indeed, for `TVAnts`, there are some peers that have been selected as contributing peers by most of the NAPA-WINE peers.
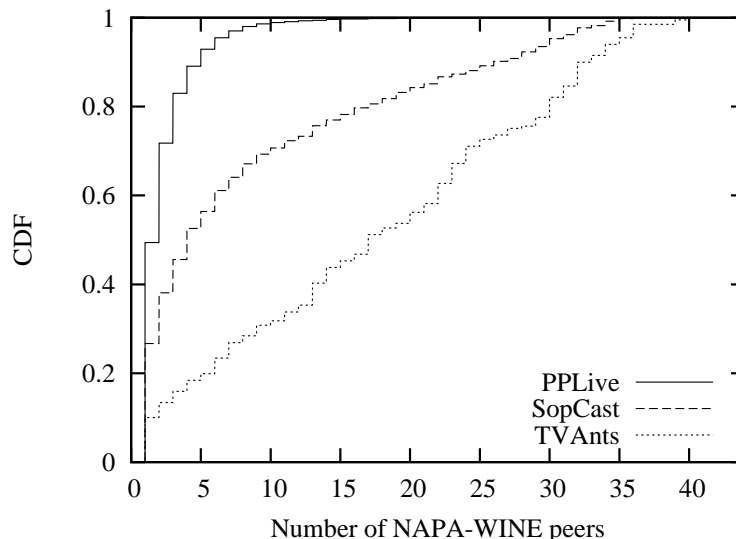
Figure 3.6: CDF of common contributing peers.

3.5   Conclusions

In this chapter we have proposed a methodology to highlight which metric is exploited by P2P-TV applications to optimize the video delivery. Considering three popular P2P-TV applications, namely PPLive, SopCast and TVAnts, we conclude that only TVAnts and PPLive exhibit a mild preference to exchange data among peers in the same Autonomous System. At the same time, no preference versus country, subnet or hop count is shown by any system. Despite the content is available from peers on the same LAN, about 82% of the video chunks are fetched from peers outside the LAN considering TVAnts. Percentages grows to 90% for PPLive and 98% for SopCast respectively. Moreover, only 32% of the content is fetched from peers inside the Autonomous System where TVAnts peers are. Even worse, PPLive and SopCast peers receive the large majority of traffic from outside the AS (87% and 96% respectively). The presented results underline the need for the development of newer and network friendlier P2P-TV systems, an interesting topic deserving future investigation. To this extent, the principal goal of the NAPA-WINE project is to design a novel P2P-TV system that *explicitely* optimizes ISP resource utilization. According to the NAPA-WINE vision, peers should download/upload the stream from/to nearby peers, they should minimize the path length, and in general they should leverage information about the network status. According to the results presented in this chapter, very little network awareness is embedded in current P2P-TV applications.

We believe that a much higher level of "network-awareness" has to be embedded in P2P-TV systems to better exploit and optimize the ISP resource utilization. In the context of the NAPA-WINE project, we are currently investigating how to reach this goal, e.g., to improve traffic localization, seeking shorter paths, exploiting topology knowledge, etc.

Chapter 4

Hybrid Analysis

In this Chapter, whose results have been published in [97], we develop an hybrid methodology which overcomes the problems found in chapter 3 (i.e., the evaluation of preference towards path-wise properties) and that exploits *active techniques to artificially enforce path-wise metrics* in a controlled testbed to analyze chunk trading policies. We modify the analysis procedure presented in chapter 3 developing a new *passive technique* that is used on the same data set of section 2.2 to infer application preference with respect to peer-wise metrics.

We apply this methodology to the study of `PPLive` where the combination of both techniques allows us to draw conclusions that are otherwise precluded using either methodology alone.

Let us briefly summarize the main contributions of this chapter are as follows:

- First, we propose a black-box methodology, based on a combination of active and passive measurement techniques, to assess the level of network awareness and friendliness of currently deployed Internet P2P-TV applications.

- Second, we apply our methodology to the analysis of `PPLive` finding that geo-localization, while not explicitly enforced by the application, actually arises as beneficial side effect of bandwidth preference.

- Third, the methodology allows to investigate the relative preference of different metrics as well: in the case of `PPLive`, we find that the peer selection process is continuously updated, with a relative preference among path-wise properties that depends on the actual magnitude of the metrics.

The reminder of the chapter is organized as follows. After having described the methodology in Sec. 4.1, we apply it to the analysis of `PPLive` reporting experimental results of active and passive techniques in Sec. 4.2 and Sec. 4.3 respectively. Finally, Sec. 4.4 concludes the chapter.

4.1 Methodology

As previously outlined, our aim is to define a methodology able to tell whether a P2P-TV system has some level of knowledge of the underlying IP network, and whether it exploits this knowledge to bias the selection of the overlay neighborhood – especially for what concerns the chunk download preference. From a high level perspective, we can define two categories of metrics that pertain to network awareness:
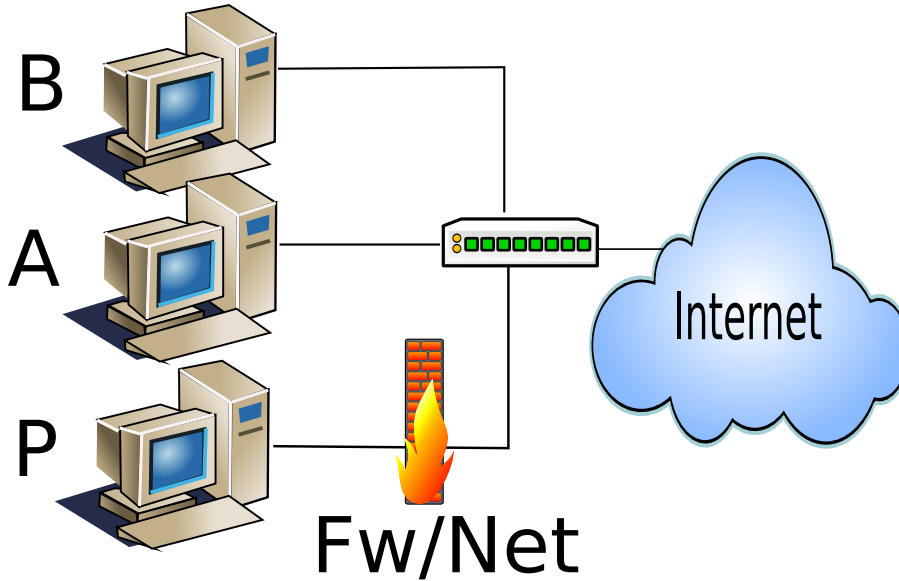
Figure 4.1: Path-wise metrics: Active testbed setup

- **Path-wise metrics**, such as IP path length (hops), loss rate, RTT delay, available bandwidth, etc., are determined by the conditions on the end-to-end path between two peers in the overlay.

- **Peer-wise metrics**, such as Autonomous Systems, geographical location, /16 or /24 IP prefix, access capacity, etc., instead only depend on properties of a single peer.

In this work, we use two separate sets of experiments to assess the awareness of a P2P-TV system with respect to metrics falling in either of the two above categories:

- On the one hand, we exploit an **active measurement** technique to enforce controlled artificial conditions (such as path length, delay, loss and bottleneck bandwidth) on a specific network path in an Internet-scale testbed.

- On the other hand, we adopt a live **passive measurement** approach, where we perform contemporary live measurement of unmodified peers from multiple vantage points, in order to investigate properties (such as AS or geographical location) belonging to real overlay peers.

4.1.1 Path-wise: Controlled Testbed

For preference related to path metrics, we setup a controlled testbed to enforce artificial network conditions as in [5], from which our approach differs for two main reasons. First, we decide to *completely* control the path metrics. This means that, unlike [5] where impairments are additionally enforced beyond the actual network conditions, we know precisely the conditions of the different peers involved in the experiments. Second, we not only test the impact of each metric in isolation, but also investigate their combined effect as well.

The configuration used for all active experiments is shown in Fig. 4.1. We use three modern desktop PCs equipped with dual-core processor running native installations of Windows XP and of the P2P-TV application, which in our case is `PPLive` 2.4. Two machines A and B are connected

to a network switch through their $100\,\mathrm{Mbps}$ Ethernet interfaces. Traffic is observed at the probe PC P, which is connected to the switch through a machine, referred to as Fw/Net in the picture, running a linux kernel 2.6 and acting as a bridge, firewall and network emulator. Notice that a large population of users, as well as the primary source of the video itself, is reachable through the Internet.

At start-up, all machines A, B, and P runs undisturbed clients for 5 minutes. During this start-up period (where we verify that play-out starts and that the clients are visually synchronized within a 1-2 seconds range), P naturally receives most of the traffic from Internet hosts. Then, at time $F_{on} = 5\,\mathrm{min}$, firewall rules are established at Fw/Net to block traffic coming from the Internet toward P, which can thus only receive traffic coming from either A or B. In this case, hosts A and B will still receive the video from the remote Internet peers, but our probe P will be forced to receive the totality of the video from A and B.

We then introduce, starting at $R_{on} = 10\,\mathrm{min}$, artificial network emulation rules (such as packet loss, RTT delay, bottleneck bandwidth limitation, etc.) on the path that joins our probe P to the hosts A and B from which he is receiving the video content. We point out that, our aim being to understand how the system biases its peer selection during *normal* operation, we verify that probe P is correctly receiving the video stream. When a path metric X is considered in isolation, we artificially worsen network conditions (e.g., increase packet loss rate, delay, etc.) solely on the path from machine A to P, by properly configuring the queueing discipline on Fw/Net. Rules on path from B to P are instead enforced only to investigate the relative importance of different metrics (e.g., delay on $A \rightarrow P$ and loss on $B \rightarrow P$). Finally, artificial network conditions are turned off at time $R_{off} = 20\,\mathrm{m}$ and firewall limitations are removed at $F_{off} = 25\,\mathrm{m}$.

The above setup allows us to focus on the *breakdown* of the bit-rate received at P between its contributors (i.e., A, B and Internet hosts), and to expresses in a visually simple way the network awareness of the P2P-TV application. Consider indeed the period when artificial network emulation rules are applied on the path from A to P, for instance: clearly, in absence of bias with respect to a given metric X, we expect the breakdown of the traffic received at P to be unaffected from variations of X. Conversely, a varying breakdown will reflect system awareness to X, with the extent of the breakdown variation as rough indication of the system sensitivity to X.

4.1.2   Peer-wise: Multiple Live Measurement

The analysis of preference related to peer properties is instead based on a large testbed, setup in the context of the NAPAWINE project [73] which we described in 2.2. In this case, we mine the data gathered from the experiment in order to infer additional information concerning the P2P-TV system bias on peer-wise properties.

The main idea is to use a *correlation-based* analysis of any given peer-wise metric X and the amount of bytes exchanged between contributor peers. As peer-wise metrics X, we will consider the Autonomous System (AS) and geographical Country (CC) properties. In this case, our aim is to test whether two peers that belong to the same AS/CC exchange more data than faraway peers, gauging the importance of X in the peer selection process.

4.2   Experimental Results: Path-wise Metric

Let us start by inspecting path-wise preference, by means of the controlled testbed depicted early in Fig. 4.1. Initially, we study path-wise metrics in isolation, enforcing either (i) decreasing bottleneck bandwidth, (ii) increasing packet loss rate, (iii) increasing RTT delay, (iv) increasing IP hop count on the path from host A to the probe P.

We then inspect the relative importance of the above metrics in the peer selection process by jointly considering different metric combinations, applying one condition (e.g., bottleneck bandwidth) on the path from the host A to the probe P, and a different condition (e.g., packet loss rate or RTT delay) on the $B \rightarrow P$ path.

4.2.1 Bottleneck Bandwidth

Results of the first experiment are reported in Fig. 4.2-(a). Time of the experiment runs on the x-axis, while the firewall start and end times are reported on the top axis as a reference. A decreasing bandwidth profile is enforced starting at R_{on} by means of a token bucket filter, with steps of $C = \{50, 10, 1, 0.5, 0.25\}$ Mbps every 2 minutes, as shown by the thick black line. Values of the bottleneck bandwidth are reported on the right y-axis, and the bottleneck is removed at R_{off}. The time evolution of the aggregated *received* rate at P is reported in the top portion of the plot, averaged over 20 seconds intervals. It can be seen that, after an initial start-up phase $t < F_{on}$ in which the incoming rate peaks up to 1.2Mbps, the aggregated received throughput at P is steady around 400 Kbps, which account for both signaling and video traffic. Moreover, notice that the aggregate rate is undisturbed during the *whole* experiment, hinting to the fact that traffic shaping did not perturbed the perceived quality of service.

Bottom plot of the figure report the *breakdown* of the traffic incoming at P with respect to the different hosts that sent the traffic to P: host A is depicted at the bottom with light (green) color, host B with dark (red) color and the remaining Internet hosts with a dashed pattern. It is easy to gather that, before firewall rules are in place $t < F_{on}$, more than 80% of the incoming traffic is received through Internet hosts. As soon as firewall rules start at $t = F_{on}$, P is forced to receive traffic exclusively from hosts A and B: since during $F_{on} < t < R_{on}$, no bottleneck bandwidth is enforced yet, the traffic splits roughly equally between A and B, as the network condition and play-out time of hosts A and B are alike. Then, as soon as a 50 Mbps bottleneck bandwidth kicks in at R_{on}, PPLive *immediately* starts preferring the unconstrained host B, which then provides the most significant portion of the traffic to P.

This observation is important as it means that (i) PPLive is extremely sensitive to the bandwidth and (ii) that it may over-react or perform faulty bandwidth estimations. This behavior might be due to the token bucket shaper used to enforce bandwidth limitations, causing strange arrival patterns that mingle the bandwidth estimation algorithm. Moreover, packet time stamping may also bias the results (e.g., by poor timing due to clock drift, clock skews due to NTP synchronization, etc). More likely, since packets of the same chunk are sent out in bursts [41], implementation of hardware card and drivers may play a very important role as well, especially due to *interrupt coalescing*: this feature, aimed at avoiding the overkill of raising an IRQ signal for every packet received, makes the cards wait during a short time-window for the arrival of other packets prior to notify the reception to the upper layers. Since packet time-stamping is then performed by the OS, this means that interrupt coalescing has a possibly very nasty impact on the bandwidth estimation as well, because two consecutive packets received by the card during the same interrupt coalescing window will then be sent to the upper layer almost at the same time. As a consequence, bandwidth and capacity estimation tools that are based on packet pair or packet dispersion, will rather measure the PCI bus speed more than bottleneck in the network.

Finally, bottleneck limitations are removed at R_{off}, which partly removes the breakdown bias toward host B; this holds until the firewall limitations are removed as well at F_{off}, after which contributors are again to be found mainly among Internet hosts.

Figure 4.2: Path-wise metrics: PPLive aggregated received rate at P (top) and its breakdown between A, B and Internet hosts (bottom) for varying bottleneck bandwidth (a) and packet loss (b). Profiles of the bottleneck bandwidth and packet loss impairments are reported as solid black lines directly in the plot and refer to the right y-axis. PPLive shows a great sensitivity towards path capacity variations while it reacts only to very high loss rates

4.2.2 Packet loss

We then conduct similar separate experiments for the other considered metrics, enforcing a single impairment at any time. Results for the packet loss rate experiments are reported in Fig. 4.2-(b) using the same similar visual presentation (i.e., aggregated rate, breakdown and packet loss profile).

We notice again that the incoming traffic rate at P is steady for any packet loss rate: indeed,

since only the path $A \rightarrow P$ is impaired, P has still the possibility to receive the rest of the video from B. In this experiment, starting at time R_{on}, we increased packet loss rate experienced by host A using the profile $L = \{0.01, 0.1, 1, 10, 20\}\%$.

As the picture clearly shows, PPLive is not very sensitive to packet loss: noticeable changes in the breakdown happens only when packet loss percentage *exceeds* 10% –indeed when loss rate is 10%, still more than half of the data is downloaded from the impaired host A. If we couple this observation to the fact that, despite packet loss increases considerably, the sent traffic rate (measured at A and not shown in the picture) does not increases proportionally, we can conclude that PPLive seems to use an effective FEC techniques, as already observed in [5].

4.2.3 IP Distance

We then test whether PPLive is aware of the host proximity, which we measure in terms of the number of IP routers that packets cross on their path across the network. Applications using raw UDP sockets can infer IP proximity by means of the Time To Live (TTL) field of IP packet header, which is initially set to an OS-dependant value (namely 60 or 64 for BSD and Linux, 128 for Windows) and then decremented by one unit at each hop in the network. We artificially increase the number of hops on the path from $A \rightarrow P$ by subtracting the number of additional hops $H = \{1, 2, 32, 64, 100\}$ from the IP TTL header field at FW/Net.

As before, a change in the hop profile is enforced every two minutes, and the results are shown in Fig. 4.3-(a). As there is no noticeable change in the breakdown irrespectively of the additional hops values, we can conclude that PPLive is either unaware the IP hop distance between two hosts, or that its inner algorithms do not rely on this piece of information.

4.2.4 RTT Delay

Finally, we verify whether PPLive does instead take latency measurement into account. We increase the delay on the $A \rightarrow P$ path so that the Round Trip Time (RTT) equals $RTT = \{0.1, 0.2, 0.5, 1, 2\}$ s, and report the results in Fig. 4.3-(b).

The plot clearly assert that PPLive is not very sensitive to the delay, as the breakdown do not show any noticeable change until the round trip delay grows very large ($RTT > 1$ s). We can thus conclude that PPLive peers do not bias their download policy in terms of nodes proximity, neither in terms of IP hop distance, nor in terms of delay. In other words, PPLive does not seem to implement proximity techniques such as [3, 12, 39, 71, 79, 92, 93].

Yet, another very important remark can be gathered from the picture. Indeed, when $RTT > 1$ s, the breakdown drastically drops: this suggest that PPLive *does* actually measure RTT, which is however not used afterward for the proximity-based video contributors selection. This behavior can be explained by recalling that one of the main aim of scheduling in P2P-TV streaming is to reduce as possible the play-out delay of the whole system. Therefore, despite useful video content may be available at high RTT peers, such peers are preferentially discarded as they may not *timely* contribute to the video content delivery, and as such they would increase the whole system play-out delay. In other words, it seems that PPLive, streaming to keep a low end-to-end play-out delay, uses RTT as a "sanity-check" and disregard such peers on purpose to avoid the system pollution.

4.2.5 Combined path-wise metrics

In order to further refine the knowledge concerning PPLive network awareness, we investigate how PPLive reacts to different combination of impairments, so to sketch a relative order of
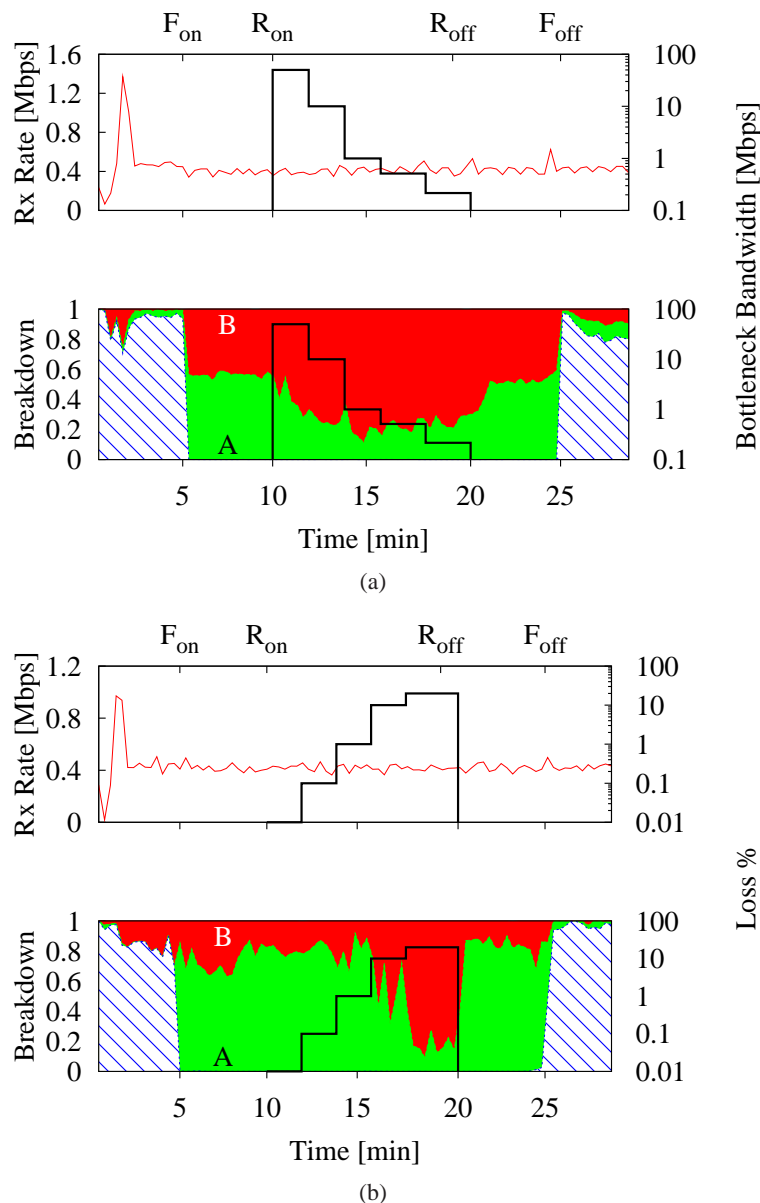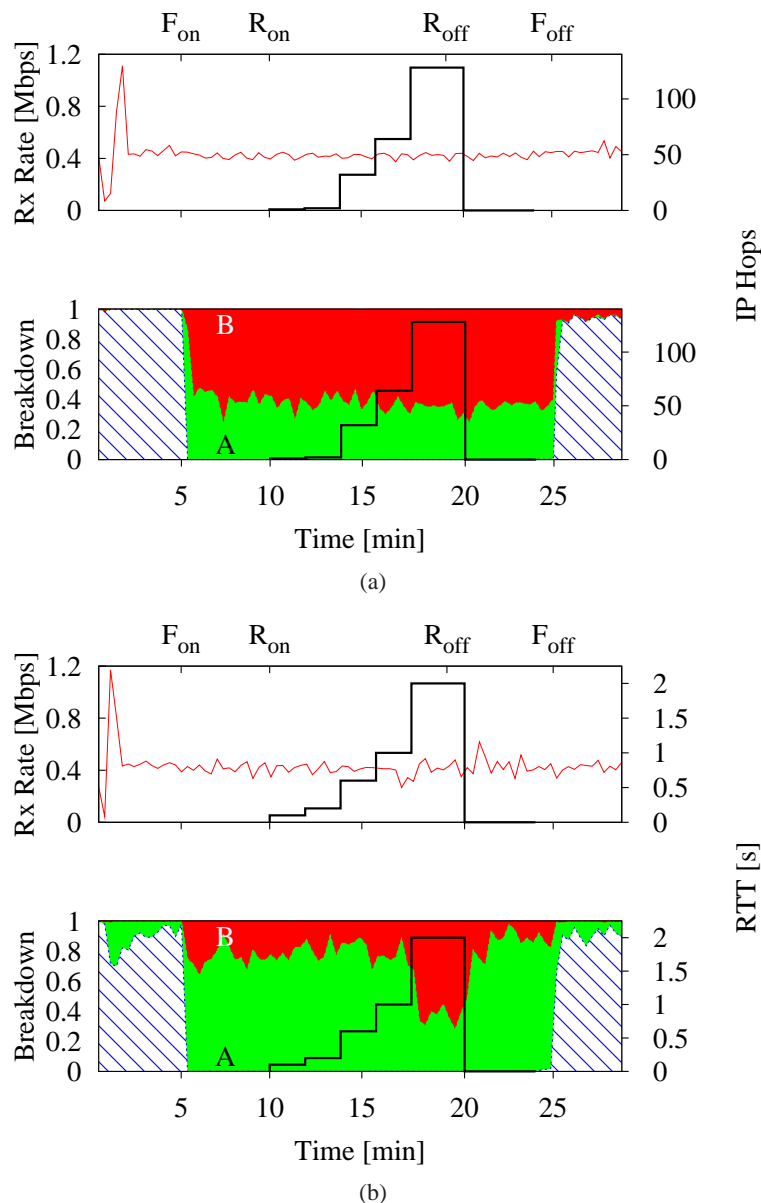
Figure 4.3: Path-wise metrics: PPLive aggregated received rate at P (top) and its breakdown between A, B and Internet hosts (bottom) for varying IP hop distance (a) and RTT delay (b). Profiles of the IP hop distance and RTT delay impairments are reported as solid black lines directly in the plot and refer to the right y-axis. Here, PPLive shows no awareness of IP distance and a mild sensitiveness towards Round Trip Time.

importance of the above metrics. As we shown that PPLive is not sensitive to IP hop count, we now limitedly consider packet loss, RTT delay and bottleneck bandwidth limitations, applying an impairment X on the $A \rightarrow P$ path, and another impairment Y on $B \rightarrow P$ at the same time.

For the sake of readability, we consider only a couple of values for each metric (i.e., $X_{hi} > X_{lo}$ and $Y_{hi} > Y_{lo}$) and investigate PPLive behavior on the four different operational points resulting from their combination $(X, Y) \in (\{X_{hi}, X_{lo}\} \times \{Y_{hi}, Y_{lo}\})$. Results are shown in Fig. 4.4, which reports for the sake of readability the value of the impairment applied to a specific path directly on
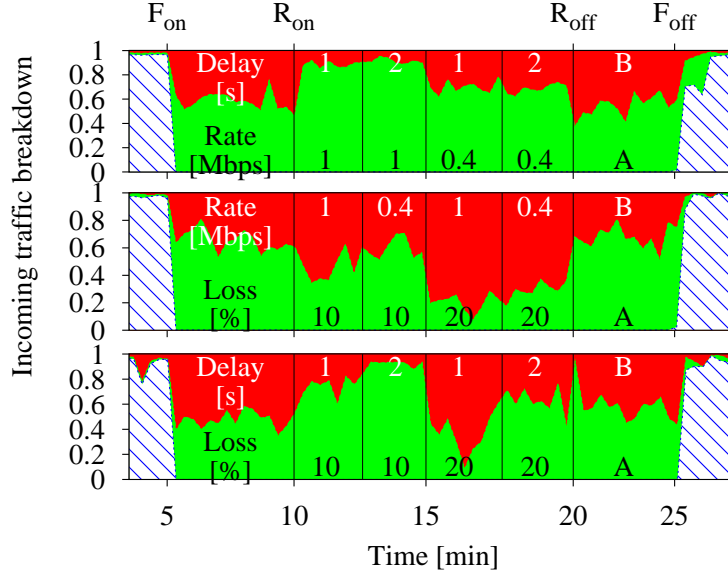
Figure 4.4: Path-wise metrics: Combined impairments, considering RTT delay vs bottleneck bandwidth (top plot), bottleneck bandwidth vs packet loss (middle plot), RTT delay vs packet loss (bottom plot). Relative importance of impairments depends on their actual magnitude.

the plot. In this case, to avoid cluttering the pictures, we no longer report the aggregated received rate but limitedly depict its breakdown.

4.2.5.1 Delay vs Rate

Top plot of Fig. 4.4 reports the case in which we apply a delay $RTT = \{1, 2\}$ s on the $B \rightarrow P$ path and enforce a rate limitation of $BW = \{0.4, 1\}$ Mbps on $A \rightarrow P$. It can be seen that preference goes toward bandwidth limited host, and is mainly driven by the bottleneck bandwidth: indeed, almost all content is downloaded from A when the rate limit is set to 1 Mbps, irrespectively on the delay toward B. Once the bottleneck rate drops to $BW = 0.4$ Mbps, the number of video chunks downloaded from B slightly increases (even though the rate limit would allow almost the whole content to be downloaded from A), but no noticeable effect of RTT variation is shown.

4.2.5.2 Rate vs Loss

Middle plot of Fig. 4.4 refers to a rate limitation $BW = \{0.4, 1\}$ Mbps on $B \rightarrow P$ and loss rate $L = \{10, 20\}\%$ on $A \rightarrow P$. In this case, contrary to the previous experiment, we see that both metrics have an impact in determining the breakdown. When $L = 10\%$ breakdown is roughly equal for A and B, with a slight bias toward A when bandwidth toward B drops at 0.4 Mbps. When losses instead grows to $L = 20\%$, the bandwidth limited host is always preferred, though the actual bandwidth limit still slightly influences the breakdown value.

4.2.5.3 Delay vs Loss

Finally, bottom plot of Fig. 4.4 refers to a delay enforcement of $RTT = \{1, 2\}$ s on $B \rightarrow P$ and loss rate $L = \{10, 20\}\%$ on $A \rightarrow P$. Again, both metrics play an important role in determining

the breakdown, depending on the impairment level. As expected, loss $L = 10\%$ do not constitute a significant impairment, as such lossy path is preferred toward high RTT path: when RTT=1 s, peer A is almost completely ignored. Behavior changes completely as soon as losses grow to $L = 20\%$, in which case breakdown favors completely B when $RTT = 0.5$ s and is more fairly split when $RTT = 1$ s.

The above observations allow us to conclude that the *relative* preference of path-wise metrics is weighted on the ground on the actual magnitude of the impairment. In principle, we point out that by exploring a wider number of (X, Y) impairment couples, it should be possible to get an even finer picture of the relative preference, but this falls outside the scope of this work.

4.3 Experimental Results: Peer-wise metric

In this section, we refine the picture of PPLive awareness by mining data gathered in the multiple-vantage point testbed. Following the methodology defined in [41], we extract from our traces about 16500 peers that contribute by providing video content. We focus on peers' Autonomous System (AS) and geographical location, which we represent by Country Code (CC) information. For each probe peer x in the testbed, we analyze the CC and AS properties of all its contributors peers y, gathered by *whois* and open IP databases queries. We point out that contributors y in this case may be either probes taking part in the experiment, or external peer of real users. As such, we no longer control the properties related to their path, and we need to infer them from packet level traces in case of need.

As core tool in this case, we use a correlation-based analysis, inspired by Principal Component Analysis (PCA) technique. While PCA [81] is often used for dimensionality reduction –i.e., to transform a set of correlated variables into a smaller subset of uncorrelated variables, called principal components– in our case our aim is to gauge the extent of the correlation, so to show the existence of a dependence (if any) between these variables.

More precisely, let us consider a set of N contributor peers $p_1..p_N$ observed during an experiment. By denoting with $X_i = X(p_i)$ the value of property X for peer p_i and similarly with $Y_i = Y(p_i)$ the value of property Y for the same peer, we measure the correlation between X and Y over the whole experiment as:

$$\rho(X, Y) = \frac{cov(X, Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y} \tag{4.1}$$

where μ_X and μ_Y are the means of X and Y over all samples, σ_X and σ_Y are the sample standard deviations of X and Y respectively. Usually, (4.1) is referred to as the Pearson product-moment correlation coefficient [24] which, dropping the sum bounds for the sake of readability, can be rewritten as:

$$\rho(X, Y) = \frac{\sum X_i Y_i - nE[X]E[Y]}{(N - 1)\sigma_X \sigma_Y} = \frac{N \sum X_i Y_i - \sum X_i \sum Y_i}{\sqrt{N \sum X_i^2 - (\sum X_i)^2} \sqrt{N \sum Y_i^2 - (\sum Y_i)^2}}. \tag{4.2}$$

4.3.1 Autonomous System and Geo-location

We are interested in assessing if PPLive is AS- and CC-aware, and whether its video scheduling policy exploits such information – i.e., if in other words our PPLive probes tend to download video content from contributors falling in the same AS or CC. By mining the experimental data, we find that, despite only $1.3\%(1.4\%)$ of peers fall in the same $AS(CC)$ of the probe, about the

12.8%(13.1%) of bytes are downloaded from them. To further quantify this evident degree of geo-localization among contributors, we evaluate the coefficient of correlation ρ between the amount of bytes RX received from any given contributor x and the fact that this contributor belongs to the same AS or CC. Considering all probes x in the experiments, and by using the indicator function $I(x, y) = 1$ when both x and y belong to the same AS or CC, we obtain $\rho(RX, AS) = 0.21$ and $\rho(RX, CC) = 0.17$ respectively, which accounts for modest (though not negligible) correlation.

This is however surprising, since the controlled testbed early suggested that PPLive peers are greedy in terms of bandwidth, but that are not sensitive otherwise to the fact that contributors are "close" in underlay terms (e.g., IP distance or latency).

4.3.2   Bandwidth

We are therefore interested in assessing if (and to what extent) the geo-location can be a (rather desirable) *side-effect* of PPLive bandwidth sensitivity. Therefore, we further evaluate the bandwidth (BW) between probes and contributor by measuring the throughput of chunks, which are typically sent out in packet bursts, to further investigate the existence of correlation between peer-wise metrics.

4.3.2.1   Bandwidth estimation techniques

Since we are unaware of the technique actually employed by PPLive to measure the available bandwidth, we adopt a hands-on approach: we estimate BW using multiple techniques, and require an agreement of our observations over *all* techniques. Considering only the downstream traffic direction of a contributing peer toward one of our testbed probes, we evaluate the BW over *windows* of fixed length. We express the window length in terms of either (i) a number of consecutive packets N or (ii) a temporal duration ΔT. Let us denote by t_i and B_i, respectively the arrival time and size of the i-th packet downloaded by probe x from contributor y during the current observation window. In case of fixed-length packet trains, we estimate the bandwidth BW_N over the current window as the amount of bytes carried by the train of consecutive N packets as:

$$BW_N = \sum_{i=2}^{N} B_i/(t_N - t_1) \tag{4.3}$$

In case of fixed-duration trains, we estimate the bandwidth as:

$$BW_{\Delta T} = \sum_{i=1}^{N(\Delta T)} B_i/\Delta T \tag{4.4}$$

where $N(\Delta T)$ is the number of packets received during ΔT.

As far as the window length N and duration ΔT are concerned, we point out that their choice is made complex not only by the fact that we are unaware of the chunk size and chunk start time, but also from the fact that the estimation can be severely influenced by factors such as interrupt coalescing[1] (which however affects both our estimates and PPLive methodology as

---

[1]Interrupt coalescing is a feature of modern network cards that limits the number of interrupt to improve performance. Usually when a packet is received by the NIC, the card raises an interrupt to notify the OS that a packet has arrived and action is needed. At high speeds this can quickly overwhelm the OS interrupt handling and slow down system performance. When interrupt coalescing is enabled, the network card raises the interrupt only when a train of packet has been received or a timer has expired; in this way the number of interrupt that have to be serviced by the OS is lowered. In our measure this feature becomes a problem because the timestamp done by pcap libraries is done at
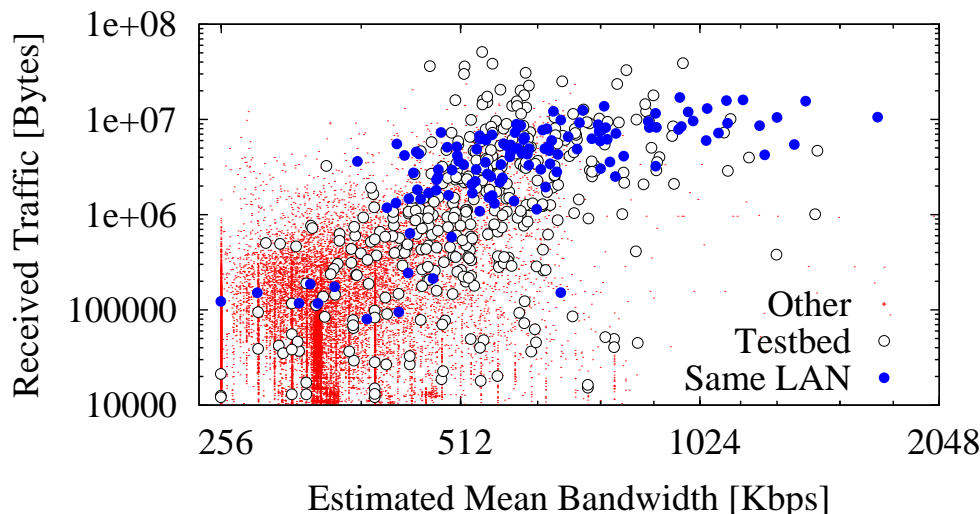
Figure 4.5: Scatter plot of received traffic versus estimated mean bandwidth (log-log scale).

well). We argue that choosing large values of N and ΔT would yield less noisy results, but due to chunk scheduling policy, it may introduce a *bias* in the result. Intuitively, counting the number of packets over large time windows equals to count the number of chunk exchanged, rather than their actual transmission throughput: using large windows, peers that more actively contributed to the transmission will thus appear as *both* preferred and high-bandwidth, introducing thus an artificial correlation between the two terms. To avoid this bias, shorter windows should be preferable. At the same time, too small values of N and ΔT should be avoided: indeed, as packets are sent out in bursts, it may happen that interrupt coalescing (which we verified to be present in out traces) can squeeze the packet arrival pattern and increase the estimated BW.

In reason of the above observations, we select values of $N = \{5, 10, 20\}$ packets and $\Delta T = \{50, 100, 250\}$ ms. For every peer pair, we then construct a series of several BW samples gathered during the whole experiment, of which we then compute the mean and 99-th percentile (p_{99}) values: we argue that both statistics are relevant, since the mean value is indicative of instantaneous network conditions, whereas p_{99} may be more representative of peer y access capacity. For interest reason, we will only report a subset of results, selecting $N = 10$ packets and $\Delta T = 100$ ms, since we verified that the same conclusions holds using the other parameters as well.

Scatter plot of the amount of received traffic versus mean bandwidth $BW_{\Delta T}$ is depicted in Fig. 4.5, using $\Delta T = 100$ ms and log-log scale. Blue points represent exchanges between any two probes in the same LAN, white points are used for probes belonging to our testbed but belonging to different institutions, whereas any other contributor is represented with a small red dot. First, hosts within our testbed, and especially hosts within the same LAN, achieve higher rates with respect to Internet hosts. Moreover, the estimated $BW_{\Delta T}$ values are sound and consistent with our expectation.

---

kernel level and when the copy of a packet train is triggered by a coalesced interrupt we actually measure the PCI bus speed.

Table 4.1: Correlation $\rho(X, Y)$ between bandwidth (BW), received bytes (RX) and peer localization information (CC,AS) for different groups of contributor peers (LAN, Internet, all)

Y :		$BW_{\Delta T}$		BW_N	
X :		mean	p_{99}	mean	p_{99}
AS		0.28	0.44	0.27	0.29
CC		0.27	0.42	0.26	0.26
RX	all	0.43	0.54	0.45	0.53
	LAN	0.62	0.75	0.51	0.61
	!LAN	0.33	0.40	0.37	0.45

4.3.2.2 Correlation-based analysis

Another interesting observation to gather from Fig. 4.5 is that host achieving higher data rates, also tend to contribute more data, and that this behavior is consistent across all three host groups.

To further quantify this behavior, we evaluate the coefficient of correlation $\rho(RX, BW)$ between the amount RX of bytes received by a given probe and the bandwidth BW toward that contributor. For comparison purposes, we also evaluate the coefficient of correlation between the estimated bandwidth BW between two peers and the fact that they belong to the same AS or CC (using the indicator function as before). Though we are aware of the fallacies of correlation based analysis, we point out that we do not seek to *prove* directional cause-effect relationship between variables, but that we rather compare the magnitude of the correlation and relatively weight their impact.

Results are reported in Tab. 4.1 for different bandwidth estimation methods. In case of $\rho(RX, BW)$ we also consider different peers subsets: namely, peers falling in the same LAN, peers that do not belong to the same LAN and all the peers altogether. As expected, we observe that irrespectively of the BW evaluation method considered, there is medium correlation between received bytes and bandwidth $\rho(RX, BW)$, which is clearly stronger for peers belonging to the same LAN. Moreover, notice that this correlation is stronger with respect to $\rho(RX, AS)$ or $\rho(RX, CC)$ reported earlier, even when all peers are considered. Also, notice that the correlation between bandwidth and geo-location $\rho(BW, AS)$ (i.e., the fact that high bandwidth contributors can be found within the same AS), is of the same order of magnitude of $\rho(RX, AS)$ (i.e., the fact that video content is downloaded from contributors within the same AS). Overall, these observations suggest that, even outside the LAN environment, peers are primarily looking for bandwidth and that the early noticed geo-localization may be a beneficial side-effect of the enforced bandwidth preference *alone*. Finally, we point out that part of the correlation might be explained by means of (i) mutual dependency between AS and BW, as well as (ii) additional hidden factors which causes both AS and bandwidth preference. At the same time, such hidden factors (e.g., preference based on IP/16 address similarity) are likely to play only an additional role beside the one played by the bandwidth, to which we shown early `PPLive` being extremely sensitive to.

4.4 Conclusions

This work proposed a methodology, based on the joint use of active and passive measurement technique for the analysis of the network awareness of currently deployed Internet P2P-TV system. The technique have been designed so to consider P2P systems as a black-box, and as such can be applied to future systems as well. As a case study, we applied the methodology to the analysis of

`PPLive` a very popular system nowadays, gathering interesting results, that we briefly summarize here.

First of all, by means of active testbed methodology, we find `PPLive` to be extremely sensitive to bandwidth, only mildly sensitive to losses and mostly unaware of IP distance, expressed in terms of either delay or IP hop count, which is in agreement with [5]. Refining further this picture, we find that actually the peer selection process is continuously updated, with a relative preference among path-wise properties that depends on the actual magnitude of the impairment.

Interestingly, by the correlation analysis of peer-wise preference gathered through the passive technique, we find that the very same bandwidth sensitivity of `PPLive`, seems to induce a desirable side-effect: namely, a moderate geo-clusterization of peers within the same AS and CC. Yet, it seems that `PPLive` does not, for the time being, explicitly enforce AS-awareness, which remains thus a new exciting challenge for the next steps of its evolution.

Chapter 5

A comprehensive framework to test Network Awareness

In this chapter we present a new methodology developed to inspect network awareness as well as its implementation in a demonstration tool, named `P2PGauge`, that has been presented in *SIGCOMM* demo session in August 2009 [94]. All results contained in this chapter have been published in [95].

In chapter 3 we studied the network awareness of P2P-TV application exclusively analyzing passive traces and we concluded that it is difficult to have a comprehensive understanding of the network awareness since path-wise metrics are hard to measure and the experimental methodology we used had some drawbacks as the bias present in the dataset. To overcome this limitations, chapter 4 presented a controlled testbed that allowed the study of chunk scheduling policies by completely controlling the application environment and challenging the application with modifications of the underlying network properties. This enabled the study of path-wise features but still we could not investigate the comprehensive set of features which possibly guide the chunk trading logic of applications. In this chapter we present an analysis framework, and its implementation `P2PGauge` [76], that exploits, at the same time, both active and passive techniques to measure and visualize the network awareness of P2P applications.

It is necessary to point out that the `P2PGauge` tool exploits active probing of peers contacted by unmodified P2P clients: whilst the tool is able to process offline traces, active probing should be better performed *simultaneously* to the running P2P application, as otherwise contacted peers may go offline (and thus be no longer available for later probing, compromising the accuracy of the dataset). Therefore, using `P2PGauge` as monitoring and analyzer tool, we performed a set of 1-hour long experiments running the SopCast application. In these experiments, a single probe peer in France is used to join different channels at different hours, exploring thus a wider spectrum of content locality and channel popularity. Moreover, care has been taken in order to consider *local* content (e.g., European Champions League football matches, or matches of the French Ligue-1) as well as *foreign* content (e.g., news and movies in foreign languages). Beside the availability of a larger number of metrics, there is another important difference between the dataset presented in section 2.2 and the dataset collected by means of P2PGauge. Indeed, in this case each experiment is carried out in isolation, while in previous experiments all peers watched the same channel at the same time. Thus, the previous dataset was possibly *biased* by the presence of several high-bandwidth peers, located in Europe, that were moreover sometimes co-located within the LAN of a single institution[1]. As a consequence, unless specific care is taken, there is the possibility that a

---

[1]For more detailed discussion about this topic please refer to section 3.2.3
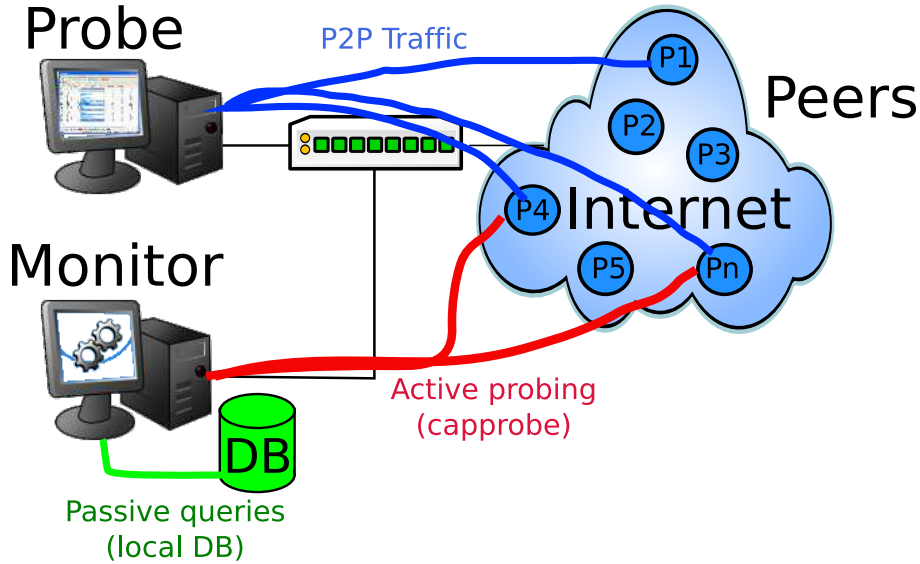
Figure 5.1: P2PGauge analysis process

self-induced artifact increases the observed geolocalization as observed in chapter 3. The fact that each observation is carried out independently, guarantees instead that such bias does not affect the new dataset, on which we focus on the following.

In the remainder of this chapter, we briefly explain the analysis process and the set of features and metrics, which we then apply to the study of SopCast network awareness. Although P2PGauge takes into account both *timescales* (e.g., short term snapshot vs long-term averages) and *traffic directionality* issues (i.e., meaning that it is possible to either separately analyze the download/upload application behavior), in the following we limitedly consider the long-term, unidirectional, aggregated traffic volume for the sake of simplicity.

5.1   Analysis Process

We describe the analysis process with the help of Fig. 5.1. In our experiments, an unmodified SopCast client runs on the probe machine, whose traffic is sniffed by the P2PGauge tool running on the monitor machine. P2PGauge analyzes the traffic generated by SopCast and collects statistics about (i) peer-wise features by passive analysis and (ii) path-wise features by sending active probes toward peers contacted by the monitored SopCast client.

Prior to delve into the features and metrics selection, let us stress an important implication of this choice. As far as passive methodology is concerned, P2PGauge gathers peer-wise features by means of a local database [65] (e.g., geolocalization and AS number, etc.) or through simple inference and analysis (e.g., IP prefix length, throughput, hop-count, etc.). Passive analysis cannot interfere with the observed P2P application traffic, but may be rather limited by database access speed: since the database API supports more than 40,000 queries per second, this clearly does not constitute a bottleneck.

However, the tool also performs *active measurements* to gather path-wise properties, thus possibly interfering with the observed P2P traffic: as such, active path-wise measurement should be limited as much as possible. Notice indeed that, although measurements are performed by a dedicated machine, monitor and probe machines share the same access link. Consider for instance

Table 5.1: Summary of the features measured by the application, highlighting whether an active or passive methodology is used.

| Feature		Type	Method
AS	Autonomous System	Peer	Passive (DB)
CC	geographical Country	Peer	Passive (DB)
NET	IP address similarity	Peer	Passive
RTT	Round Trip Delay	Path	Active
CAP	Capacity	Path	Active
HOP	IP hop-count distance	Path	Active

the issue of path capacity estimation: expensive active-path probing techniques (such as bandwidth measurement by means of packet trains) are not suitable for our purposes, and we rather need light-weight measurement technique (such as those based on packet-pair dispersion). In reason of this observation, we resort to CapProbe [46] to actively estimate the bottleneck capacity, the RTT delay and the IP time-to-live (from which we can infer the IP hop path distance). For each peer, we perform $N = 100$ measurements by sending pairs of back-to-back ICMP packets, and each pair is spaced by $\Delta T = 0.5$ seconds.

To limit the number of probes during intense network discovery phase, we further upper-bound the number of concurrently active path-probing processes at $C = 50$. Although the amount of active-probing traffic is limited to $R = 2C/\Delta T = 200$ packets per second, performing active experiments for the whole peer population may be a prohibitive task. Furthermore, concurrent experiments may have mutual influence, thus we would like to reduce their occurrence. To this extent, we recall that a large number of peers is only contacted once (i.e., during the network discovery phase), but is not contacted later on – thus is not involved in the content exchange. While such peers may constitute a significant percentage of the peer population (e.g., in case of PPLive), they are nevertheless irrelevant as far as the traffic volume is concerned. As we are interested in the bulk of the traffic volume, we thus limit active measurements only to peers that actively contribute to the video stream. Specifically, we consider only peers who send at least two packets in a time window ΔT. This simple heuristic still allows to focus on the bulk of the traffic volume (e.g., above 95% for the worst case application, namely PPLive), while significantly limiting the bias induced by active probing traffic. Notice that this heuristic is robust and applies also to other classes of P2P services such as filesharing.

5.2   Features Definition

Table 5.1 briefly summarizes the features that we take into account when measuring network awareness specifying if they are peer or path wise and the type of measure we use to analyze them. The choice of the features pertaining network awareness has already been preliminary discussed in previous chapters. We point out that in some cases it may be possible to measure the same feature (e.g., IP TTL, RTT, etc.) with either methodology. Still passive measurement can be less reliable than active ones: e.g., in case of RTT, the difficulty lies in matching data packets with the corresponding application-layer acknowledgement. We therefore follow a conservative approach, and adopt on the most accurate methodology for each feature. More precisely, we exploit *passive analysis* to infer AS, CC and NET properties, while we use *active probing* to measure the CAP, RTT and HOP features, which are described as follows:
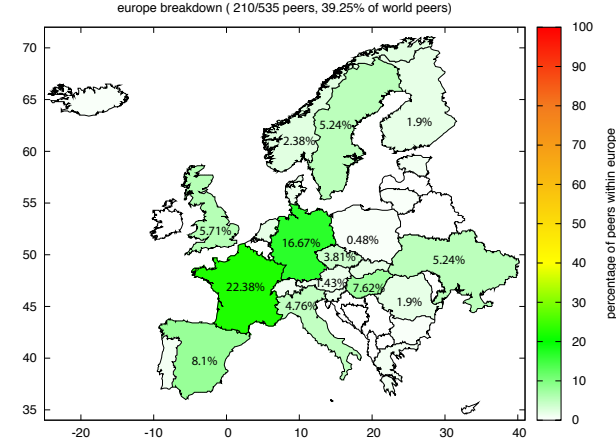
(a) Breakdown of peers in European countries
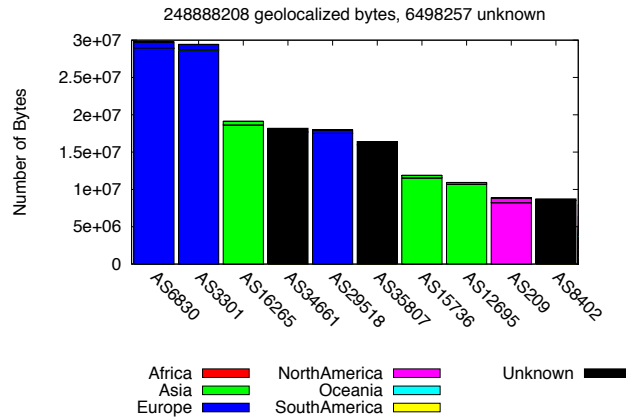
(b) Breakdown of bytes in Autonomous Systems

Figure 5.2: Example of geolocalization analysis (passive metrics) that can be done with P2PGauge.

Autonomous System (AS) and *Country Code (CC)*: For these peer-wise properties, we rely on same public database [65], used early to gather cross-layer metrics, which enables us to map public IP addresses to Country Codes (CC) or Autonomous System (AS) numbers. From this kind of analysis P2PGauge is able to show at a glance the geographical distribution of traffic and its breakdown into ASs as showed in figure .

Network prefix (NET): Namely, the length of the bitwise prefix match between the monitored peers IP address and the IP addresses of the peers it contacts. This feature gives a raw estimation of peers distance in the IP address space: when two peers are in the same sub-network, they likely share a longer prefix than faraway peers.

Path capacity (CAP): We measure the bottleneck capacity along the path between two peers with CapProbe [46], a packet-pair technique that infers capacity based on the dispersion of the acknowledgement packets on the backward path. Bottleneck capacity is measured over $N = 100$ packet-pairs measurements.

Round Trip time (RTT): RTT measurements are directly available as a side effect of Capacity probing. Indeed, CapProbe sends $N = 100$ packet-pairs, from which we gather the same amount of RTT samples.

IP hopcount (HOP): The IP hop-count distance corresponds to the number of layer-3 nodes traversed by an IP packet. Usually, we infer as in chapter 3 this value from the TTL field in the IP header of the CapProbe packets. However, we found that, in some cases, this value is mangled by non-standard networking devices: in case P2PGauge notices such anomalous behaviour, it falls back on the more reliable (but longer and more costly) path discovery operation by means of the common Traceroute tool.

5.3 Metric Definition

P2PGauge acquires a great number of informations: namely, the amount of sent and received traffic, along with the path-wise and peer-wise features early described is stored for each remote peer contacted. This raw information has thus to be processed in order to be displayed on a Kiviat chart. At the same time, a careful selection of display metrics should be made, in order not to loose too much information in the data processing. In this section, we present two out of the four metrics implemented in the P2PGauge software namely the *preferential partition* and the *Kullback-Leibner distance*. `P2PGauge` implements other two metrics (*correlation* and *Bhattacharyya distance*) that are not presented here as they are out of the scope of this chapter as they do not bring additional value to our work but let us stress that our framework can be easily integrated with other metrics demonstrating thus its flexibility.

5.3.1 Preferential Partition (PP)

As the simplest and most intuitive metrics, we ressort to the preferential partition (PP) metric already defined in chapter 3. Let us denote with $\mathcal{N}_k$ the set of peers who where discovered by the application from time 0 to time $T = k\Delta T$. For each feature F, the set $\mathcal{N}_k$ is split in two disjoint groups $\mathcal{N}_k = \mathcal{N}_k^{close(F)} \cup \mathcal{N}_k^{far(F)}$, so that peers that are "close" to the monitored peer X in terms of the feature F are grouped altogether.

Specifically, we use the following rules to partition the sets. We consider peers falling in the same AS and CC of the monitored peer to be part of the close peer set. As far as the NET feature is concerned, we use a fixed threshold of 16 bits, above which we consider peers to be close. Finally, for the RTT, HOP and CAP features we use a relative threshold, equal to the median value computed over all peers: namely, peer whose RTT and HOP values are below the median threshold are considered to be close, while peers having a bottleneck capacity CAP higher than the threshold are included in the preferential set.

Based on this simple partitions, we now quantify the preference level by evaluating the percentage of bytes that the monitored peer X has exchanged with peers belonging to the preferential set $\mathcal{N}_k^{close(F)}$, as:

$$PP_F = \frac{\sum_{Y \in \mathcal{N}_k^{close(F)}} B(X,Y)}{\sum_{Y \in \mathcal{N}_k} B(X,Y)} \tag{5.1}$$

Considering the RTT feature, Fig. 5.3-(a) exemplifies the preferential partition as a gray shaded zone: in the scatter plot, each (x, y) point corresponds to the amount y of bytes exchanged with a
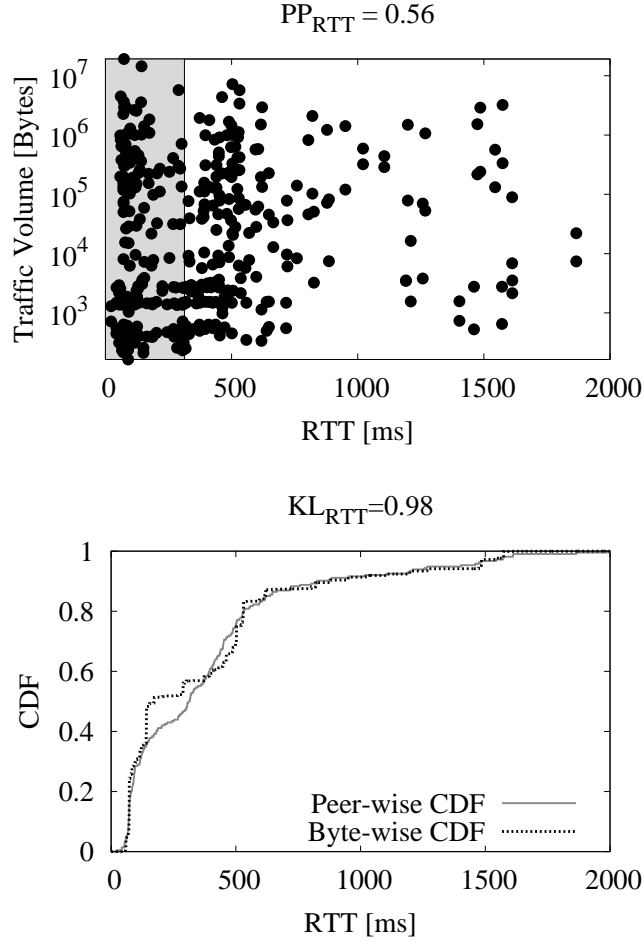
Figure 5.3: Network-awareness metrics: (a) Preferential partitioning PP_{RTT} and (b) Kullback-Leibner divergence KL_{RTT} of the RTT feature.

peer having a given RTT equal to x. In the case of figure, about 56% of the data is exchanged with the 50% of peers that constitutes the preferential set (notice that, since we used the median RTT as threshold, the population size is equal for both sets), hinting thus toward a slight preference for peers that are close in IP-latency terms.

5.3.2 Kullback-Leibler (KL)

As a second metric, we consider the Kullback-Leibler (KL) divergence [51] (5.2), which is a known measure of the distance between two probability distribution functions (pdf) p and b:

$$KL(p\|b) \;=\; \sum_{x \in X} p(x) \log \frac{p(x)}{b(x)} \qquad (5.2)$$

We use the KL divergence to measure difference between the *peer-wise* and the *byte-wise* pdf of a given feature F. In other words, we evaluate the pdf of F, either counting each peer once, or by taking into account the volume of traffic that remote peers have exchanged with the monitored peer. The KL divergence tells us whether the two distribution matches (KL$\simeq$0), or whether some

discrepancies arises instead (KL>0). Notice that, as opposite to before, a large KL value cannot be directly read as *preference* indicator: rather, it merely pinpoint the existence of a *bias* between the number of peers exhibiting a given value for a feature F, and the amount of bytes exchanged with those peers. For instance, a large KL_{AS} value does not mean that a large amount of bytes is exchanged with peers falling in the *same* AS, but rather expresses the fact that *some* AS possibly contributes for a significant portion of the traffic, inducing a distortion in the byte-wise pdf with respect to the peer-wise one. In other words, high KL values correspond to high bias, which however do not necessarily translate into higher awareness.

An example of the KL_{RTT} metric is reported in Fig. 5.3-(b) considering the same dataset depicted in Fig. 5.3-(a). In this case, dashed and continuous lines are used to represent the byte-wise and peer-wise RTT cumulative distribution functions respectively. In the case of figure, notice that the two curves do not overlap, which is especially visible for RTT$\in [200, 300]$ ms, and that yield to a value of $KL_{RTT} = 0.98$. This means that there is a group of peers, whose RTT is about [200,300] ms, that contribute more data than others: notice indeed that such a couple of highly-contributing peers is clearly visible in Fig. 5.3-(a) in the same RTT range.

5.4   Experimental Results

In order to show the awareness of the application "at a glance" we resort to use the Kiviat representation [50] which has been introduced in networking research by [66] to report noteworthy characteristics of different classes of applications (e.g., Web, interactive, VoIP, etc.). A Kiviat chart consists of several axis represented in the same planar space. Each axis reports a different feature, as in figure 5.4 where AS, CC, NET, CAP, RTT, HOP are reported, respectively by their preferential partition value 5.4(a) and Kullback-Leibler distance 5.4(b). For each feature we report its mean value μ over all peers in the dataset: by joining mean values of different features together with a line we obtain a closed shape – the Kiviat chart. Furthermore to show the variability of application behavior among different peers, we draw thin lines to represent the standard deviation σ of the features, and depict them relatively to the average (i.e., thin line represent $\mu \pm \sigma$) and we shade the area between the curves for the sake of readability. For each feature we report the maximum range value under the feature label of each axes directly in the graph.

Kiviat charts in figure 5.4 are arranged in such a way that features gathered by passive inference (i.e., AS, CC and NET) are represented on the three top axis, whereas features involving active probing (i.e., CAP, RTT and HOP) are represented on the three bottom axis.

Let us consider the preferential partition metric first, which is depicted in Fig. 5.4-(a). It is easy to notice that, despite experiments include content that is very popular in EU (e.g., Champions League matches) and possibly also very local (e.g., French Ligue-1 matches), nevertheless SopCast managed to find a few peers that were located in the same network ($PP_{NET} \simeq 0\%$), AS or CC ($PP_{AS} \simeq 1.6\%$ and $PP_{CC} \simeq 4.5\%$) boundaries and to exchange data with. We would like to stress that in figure 5.4 we take into consideration all the aggregate of traffic without distinguishing between inbound and outbound traffic; P2PGauge instead allows the user to chose which kind of traffic to observe. As already said earlier, we underline that, since we just analyze the application in isolation, we don't have to worry about an experimental bias issue.

Taking into account the capacity feature, we can notice that SopCastshows a slight preference for higher bandwidth peers ($PP_{CAP} > 50\%$) meaning that SopCast presents a slight preference in exchanging data with peers whose capacity is higher than the median. This observation is in accord with results gathered early in this thesis in chapter 4

On the contrary, no such preference is shown for close peers, as only about half of the overall

(a) Preferential Partition: notice that preferential partition value is included between 0 and 1.

(b) Kullback-Leibner divergence: KL values are included between 0 and 2.
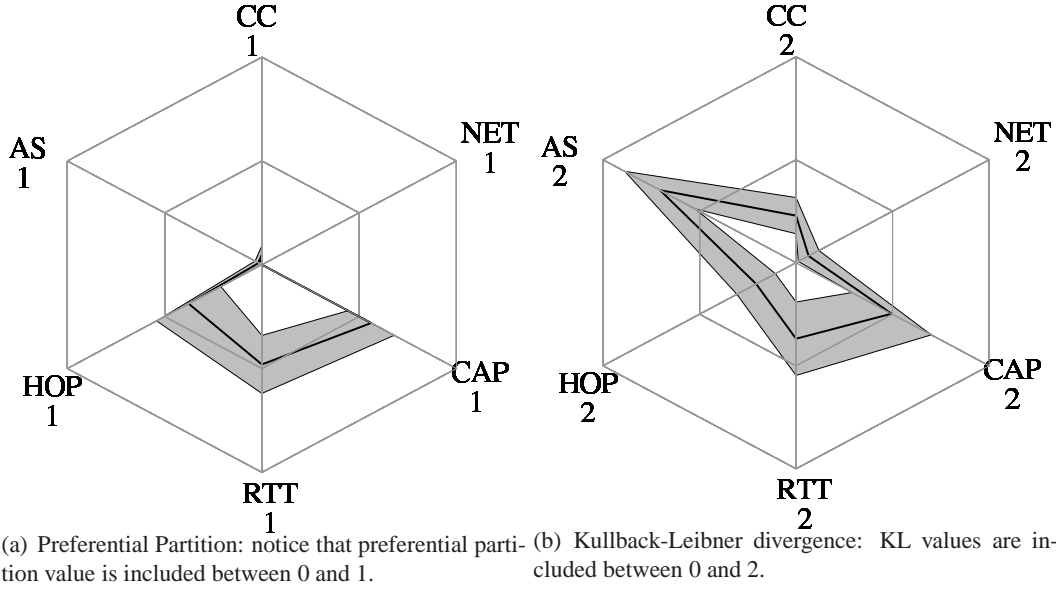
Figure 5.4: Network-awareness representation: Kiviat charts of SopCast dataset. Features gathered with passive measurement are displayed on top axis (AS, CC, NET), features requiring active measurement on the bottom axis (HOP, RTT, CAP).

traffic volume is exchanged with peers close in terms of RTT latency ($PP_{RTT} \simeq 50\%$), hinting that the application is not actively trying to reach traffic locality by confining the traffic within close peers. Similarly, the fact that $PP_{HOP} < 50\%$ confirms that slightly longer IP paths may be taken to find those high-capacity peers. Again the application could try to be network friendly by taking into account the number of hops instead of using DBs to pinpoint peers in the same AS but this seems not to be the case.

Let then finally consider the Kullback Leibner plot of figure 5.4(b). In this case, we recall that a larger KL value expresses a larger bias, but not necessarily larger awareness. In this case, a large bias is exhibited for the capacity KL_{CAP} metrics, corroborating in this case the hypothesis of a greedy selection policy. An even larger bias is visible for KL_{AS}, which in this case corresponds to an unbalanced traffic distribution. In this case, a few ASes act as main contributors: however, such ASes differ from the monitored peer AS, and their occurrence may rather be the result of other peer-selection policies (e.g., possibly due to the presence of high capacity peers in such ASes). Overall, we can conclude that current popular P2P-TV applications such as SopCast, have not yet considered network-awareness issues.

5.5   Conclusions

This chapter presented a comprehensive framework for the characterization of P2P applications' network awareness based on a black-box measurement and analysis of the traffic they generate, coupled to an expressive data representation exploiting Kiviat graphs. We implemented the methodology in a demonstration software called P2PGauge that we used to carry an experimental campaign of 1 hour long analysis of the SopCast application during which we joined different channels of different hours to have a wider spectrum of scenarios.

Results presented in this chapter confirm the ones obtained early in this thesis showing that SopCast has a slight preference in exchanging content with high capacity peers while it does

not show any specific preference either towards peers close in term of RTT, hop distance or IP distance, either toward nodes located in the same country or autonomous system

Part II

Implementing network awareness

Overview

In this second part we abandon the study of the network awareness of existent P2P-TV applications and we focus on the interaction of network aware algorithm with the underlying network layer. In chapter 6, we make use of a chunk-level P2P-TV simulator to gauge the impact of non ideal scenarios on the P2P systems performance, then, in chapter 7, we use a packet-level emulation tool to gauge the effect of layer 7 routing over a reactive network in which traffic engineering is active.

Chapter 6

Simulation Analysis

In the last years, a number of different proposals have targeted mesh-based P2P streaming [12, 17–19, 35, 56, 61, 62, 82, 92, 93, 102, 104, 111, 128, 130]. With few exceptions [60, 101] such proposals have typically been studied in isolation, possibly focusing on very specific aspects of the system (notably, chunk scheduling policies), in possibly highly ideal settings (e.g., overlay-only studies, homogeneous settings, synchronous timelines, perfect neighborhood knowledge, etc.). As such, a thorough comparison of the different proposals under a common and realistic framework is missing so far. A first aim of this work is thus not to propose any new system, but rather to compare existing ones. A second aim is instead to understand how the performance of these system declines under more realistic scenarios.

In this chapter, we implement some of these algorithms [12, 17, 93, 104] in a custom event driven simulator, and evaluate their performance considering important (but often overlooked) factors, which we model with increasing levels of realism. A first issue is that "network aware" P2P-TV systems typically makes chunk scheduling and topology management decision based on some measured properties of other peers in the swarm: yet, as gathering precise and reliable measurements is notoriously difficult in the Internet, it is important to understand the implication of *measurement errors* in the system performance. A second issue is that scheduling algorithms are generally evaluated assuming a perfect, though unrealistic, knowledge of the system state (e.g., neighbors buffer maps): as such, it is important to evaluate the impact of *state inconsistency* (e.g., due to lost or outdated control information) as well.

Our main findings can be summarized as follows. On the positive side, we find that system performance are rather robust to measurement errors, as performance degrades gracefully even for very large capacity and latency measurement errors. Conversely, we find that state inconsistencies significantly degrade the achievable performance even for very low signaling error rates: as such, signaling should not be neglected in future studies aiming at a realistic assessment of the quality provided by P2P-TV services.

This chapter extends the results published in [98] adding the study of more network models and examining the impact of a topology management mechanism. It also contains a section the analyzes the impact of measurement errors on systems performance and new experiments that justify our scenarios decisions. The simulator tool used in this chapter (P2PTV-Sim available at [78]) has been developed by NapaWine partners; in particular Politecnico di Torino provided the core of the simulator and basic scheduling algorithms. We developed the L3 network layer, the measurement error framework and some scheduling algorithms used in our comparison.

The rest of the chapter is outlined as follows: section 6.1 presents work related to ours, 6.2 describes the architecture of our simulation environment and 6.3 presents results of the impact of both layer 7 and layer 3. Finally section 6.4 reports the analysis of the interaction between L7 and

L3 and section 6.5 summarizes the contribution of the chapter.

6.1   Related work

Starting from [17], many schedulers have been proposed that also incorporate awareness to the network environment (such as bandwidth [104], latency [12], and their ratio [93]). At the same time, many works proposing novel algorithms [12, 17, 93, 104, 130] have been studied in isolation, possibly adopting a highly idealized view of the system and of the network models. Though simplistic, this viewpoint is nevertheless necessary to gather solid theoretic foundations for specific algorithms design choices. In this work, we focus on such class of schedulers, which we analyze in a common framework under more realistic conditions. With this respect, closest work to our is [60] that, by means of simulation, however limitedly compare two systems (namely *SplitStream* [18] and *PRIME* [61]).

   We point out that full blown systems [19, 56, 61, 82] have also been evaluated by means of middle to large scale deployments of real prototypes. With the exception of [101] (that compares *Chainsaw* [80] and *SplitStream* [18] and is the closest work in spirit to ours), and of the methodology presented in chapter 3 (that analyzes *PPlive*, *SopCast*, and *TVAnts*), real systems have however been studied in isolation. Moreover, what makes the comparison difficult is that experimental conditions are hardly reproducible. Also, although performance results are in this case realistic, as systems are very tightly designed, it is often not possible to isolate and understand the impact of different factors in the overall system performance.

   This chapter aims at reducing the gap between the above classes of work, by performing a thorough and realistic, but controlled and reproducible comparison of relevant systems proposed in the literature [12, 17, 93, 104]. In order to provide a fair comparison, we consider several algorithms that perform well in ideal settings, and implement them in a common simulation framework. We then challenge these algorithms by plugging different models, representative of a realistic Internet environment, so to assess their performance into the wild.

6.2   Framework Description

This section overviews the framework we devised to compare P2P-TV systems, which is available as open-source software at [78]. The custom chunk-level event-based simulator takes into account several components, which are visually presented in Fig. 6.1. From an high-level point of view, the framework consists of two layers: namely, the underlaying physical L3 network and the logical L7 overlay, which are coupled by different models of their possible interactions.

   From the L3 point of view, at the edge of the architecture we have end hosts, which are physically interconnected to the L3 network by access links, that acts as bottleneck, and that are modeled as a capacity–delay pair. Hosts are attached to edge routers, which constitute the entry point of P2P-TV traffic in the network, which we model with increasing levels of details. From the L7 viewpoint, hosts run P2P-TV applications, which we express in terms of the algorithms (e.g., chunk scheduling, peer selection, topology management) they implement, and of the overlay graph resulting by those algorithms. Finally, we model L7/L3 interaction by taking into account that, in the real world, different sources of error can slip in at any point of the process (e.g., loss of signaling packets, bias on measurement of L3 properties performed by L7 overlay peers, etc.).

   In the remainder of this section, we further detail each component, motivating the realism and soundness of our choices. At the same time, we point out that the framework is extremely flexible, and can easily accommodate other models for the different components as well: as such, where
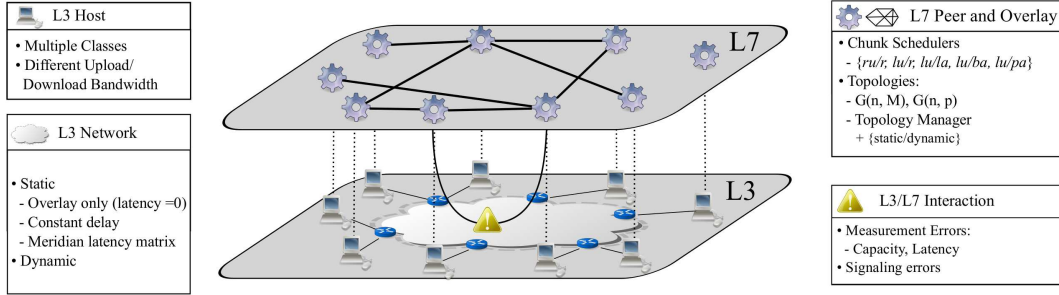
Figure 6.1: Sketch of the evaluation framework: overview of L3 and L7 components under study, including L3/L7 interaction

Table 6.1: Breakdown of hosts into classes

Class	Ratio	BW_D	BW_U	t_{TX}
I	10%	∞	5.0 Mbps	20 ms
II	40%	∞	1.0 Mbps	100 ms
III	40%	∞	0.5 Mbps	200 ms
IV	10%	∞	0 Mbps	∞

relevant, we list other interesting models that could be investigated by further research but that are out of the scope of this work.

6.2.1 L3 Components

With L3 components we indicate objects in the physical world, such as (i) hosts and (ii) routers, that are interconnected by a (iii) network.

6.2.1.1 Host

Hosts are machines running P2P applications instances, and are characterized by a physical interface to the L3 network. Hosts are divided in different classes according to their upload bandwidth BW_U, while we consider the download bandwidth BW_D to be infinite. This is a reasonable assumption in case of asymmetric access, provided that we further assume that the bottleneck is placed at the edge of the network (which represents the common case today and is generally assumed by other research on P2P-TV [93, 104] and P2P-filesharing [83]).

In our simulations, except where explicitly stated, we consider N_H=2000 hosts divided into four classes, where the average $BW_U(i)$ for the i-th class is allocated as described in Tab. 6.1, consistently with [83, 104] (we further motivate this choice in section 6.2.4.3). The first column of Tab. 6.1 reports the class breakdown: the bulk of peer population is constituted by mid-speed peers, with a non marginal presence of very-high and very-low speed peers. In class i, the uplink capacity of each peer p is set to $\nu \cdot BW_U(i)$ where ν is a random variable uniformly distributed in $[0.9, 1.1]$ (i.e., the actual uplink of each peer deviates at most 10% from the average for that class). For reference purpose, last column reports the transmission time t_{TX} of a 12.5 KBytes chunk (corresponding to about 10 full-payload packets, considering application layer header), where we consider that all the uplink bandwidth is devoted to the chunk transmission.

6.2.1.2 Router

Each host is single-homed, i.e., attached to a single access router, which models the first IP router in the aggregation network (e.g., the BRAS for ADSL access). In our simulations, we consider a number of routers equal to $N_R = 100$ and we use a simple host-to-router mapping policy: each host is randomly bound to a router, so that in average $N_H/N_R = 20$ hosts are attached per router.

As depicted in Fig. 6.1, routers are placed at the edge of the network and act as access points forming a logical full mesh at L3. Each router keeps statistics about packets passing through its interfaces and discriminates traffic between *remote* (i.e., the traffic that it injects further down toward the core) and *local* (i.e., the traffic that is reflected toward other access links insisting on the same router).

Notice that, in this way, routers directly yield a very simple measure of traffic locality as $P\% = local/(local + remote)$ (which is independent from the actual network topology, from the Autonomous System AS level topology, from the router-to-AS mapping policy, etc). We point out that the *absolute* value of this measure is heavily affected by several factors (e.g., AS topology, host-to-router mapping), which disqualify this index to be used for realistic assessment of locality awareness. At the same time, this rough indication however allows to *relatively* compare the locality awareness of P2P-TV systems, which is our main aim more that to evaluate the amount of inter-AS traffic (for which we refer the reader to [12, 85]).

6.2.1.3 Network

The L3 network models the interconnection of routers: in this work, we consider different models of network, with additional complexity and levels of details.

If we consider the access link to be the bottleneck, likely no queuing happens within the network core: as such, the network simply models the delay of the end-to-end path. In this case, the network topology is well represented by a *static* latency matrix between routers, where the latency essentially represents the propagation delay along links of the router-to-router path. Notice that, in this scenario, two host attached to the same router sense a latency of 0. We consider different models of static networks, from an ideal overlay model (where the end-to-end delay is given solely by the chunk transmission duration over the uplink bottleneck) to more realistic models such as Meridian [38] (where end-to-end delays are derived from real measurement performed among a large number of Internet hosts).

We also consider the case where congestion may still happen in the network by employing *dynamic* end-to-end latency matrices, where the latency between any two peers may thus differ from chunk to chunk. We point out that the case where the amount of P2P-TV traffic is (i) minority or (ii) prevalent shall be considered separately. In the former case, which is typical today and that we consider in this work, congestion is due to the background traffic: we model this effect by simply varying the latency between two consecutive chunks at random. In the latter case network links should be modeled as well, so that Traffic Engineering mechanisms (e.g., load balancing, periodic optimization of routing weights IGP-WO, etc.) could be applied to handle the edge-to-edge traffic matrix induced by P2P-TV traffic. In this chapter we limit our investigation to case (i); in the next chapter (7) we will study case (ii) with the help of a network-emulator tool capable of performing traffic engineering.

6.2.2 L7 Components

With L7 components, we indicate higher level components, such as (i) peers, which are instances of L7 P2P-TV applications running on L3 hosts. In more detail, we model peers by defining the

algorithms they implement: specifically, each peer has to (ii) manage the overlay topology and (iii) schedule the transmission of chunks on the overlay links.

6.2.2.1 Peer

Each peer establishes and maintains several logical connections to other peers in the overlay: we denote with $N(p)$ the set of peers in the neighborhood of p. As in mesh-push systems chunks are not received in playout order, peers need to have a buffer-map B that describes the chunks received and stored into the peer memory. Given a peer p, we indicate with $B(p)$ its buffer-map, and denote by $c \in B(p)$ the fact that peer p has received chunk c. The size of the buffer map $B(p)$ determines P2P-TV performance as in the following tradeoff: large buffer maps reduce the chunk loss probability, but increase the time lag with respect to the source chunk generation time; conversely, small buffer maps reduce the playout delay with respect to the source at the price of an increased chunk loss probability (as chunks that arrive later than the playout delay are no longer useful and thus can be considered as lost).

In order to gather performance of the system in *steady state*, in this work we do not consider churn (i.e., peers arrival or departure). While this choice may seem strange at first sight, especially given our attention to the realism of the scenario, we nevertheless show its soundness in section 6.2.4.1. Shortly, results from a measurement campaign in real ISP network show that, while churn in filesharing applications is dominated by user habits, the churn in livestreaming applications is dominated by the content schedule: in other words, users connect to watch specific content at a specific time, and stay connected during the whole program.

6.2.2.2 Overlay Topology

Logical links established by peers form an overlay topology. To enhance their performance, peers may perform topology management: i.e., they rearrange their overlay neighborhood in order to exploit population heterogeneity, so to globally optimize the topology based on local decisions.

In this work, we focus on topology management by considering it as either (i) a black box tool that induces a particular type of overlay graph or (ii) a specific algorithm that continuously adjusts the topology. In more details, for (i) we consider different random graphs with a mean degree $d_0 \simeq 20$ [1] that are created at $t = 0$ and are never changed later on, and that thus define a fixed logical neighborhood for all peers at time $t = 0$. For (ii) we additionally consider a topology management process that continuously run and adapts the initial topologies, based on the measured peer properties (e.g., latency for geolocalization or capacity for performance). Indeed, since higher capacity peers can serve more neighbors, placing them near the source allows spreading new chunks faster and to a greater number of nodes. This should turns out in a per-chunk diffusion trees (i.e., the instantaneous trees followed by each chunk, which differ from chunk to chunk) with higher fan-out and reduced depth.

We point out this to be a reasonable approach: indeed, considering a $G(n, M)$ or $G(n, p)$ topology at time $t = 0$, roughly models a system in which peers joining the system receive a small number of bootstrap peers (e.g., by means of a BitTorrent-like tracker) that constitute their initial neighborhood, which may be then continuously adjusted (e.g., by means of a BitTorrent-like peer exchange PEX function [2]). We point out that according to chapter 3, applications in the Internet may exhibit behavior closer to case (i) such as TVAnts and Joost, or to (ii) such as PPLive and

---

[1]We used two kind of random graphs, both having $n = 2000$; the difference lies in the building of graph: first class of graph is built by assigning, at $t = 0$, to each node, a fixed number of neighbors (10 in our case). Second method is to assign to each node a number of neighbors chosen by a Poisson process whose rate is 10.

[2]http://www.rasterbar.com/products/libtorrent/extension_protocol.html

Table 6.2: Chunk scheduler policies

Scheduler	Description		
ru/r [17]	Random useful chunk	/	Random peer
lu/r [17]	Latest useful chunk	/	Random peer
lu/la [12]	Latest useful chunk	/	Latency-aware peer
lu/ba [104]	Latest useful chunk	/	Bandwidth-aware peer
lu/pa [93]	Latest useful chunk	/	Power-aware peer

SopCast, which makes both cases relevant. We also point out that despite other graphs could be considered for (i), such as Barabasi-Albert [11] scale-free and Watts-Strogatz [117] small-world, this would not however add further realism to our simulation campaign – as we will see, the topology dynamics are far more important than the initial conditions at time $t = 0$.

6.2.2.3   Chunk Scheduler

The ultimate goal of any P2P-TV system is to give each peer a continuous stream of data: as such, peers must avoid having gaps in the buffer-map positions that are closer to the playout deadline. The video exchange process is handled by a chunk scheduler, which acts whenever a peer can use the host upload bandwidth. In push systems, any peer p runs a scheduler that has to choose: (i) a chunk from its buffer map $B(p)$ and (ii) a destination peer among its neighbors $N(p)$.

Scheduling algorithms can be divided in two classes depending on the order in which the chunk/peer selection is made: in this work, we focus on algorithms that first chooses the chunk to send and then the destination peer. We consider the chunk scheduling algorithms proposed in [12, 17, 93, 104] which we summarize in Tab. 6.2. Loosely following [17], we denote each algorithm as *c/p* where *c* and *p* stand for *chunk* and *peer* selection algorithm respectively.

The simplest scheduler is the work-conserving ru/r, that select a *random* chunk $c \in B(p)$ which is sent to a random *useful* peer $p' \in N(p)$, i.e., a peer that misses that chunk $c \notin B(p')$. We then consider a series of schedulers that select the *latest useful* chunk in their buffer-map, which then they send to a useful peer (i.e. a peer that has not received yet the chunk) selected according to either a lu/r random strategy [17] or a *network-aware* criterion $lu/\{la, ba, pa\}$. As far as network-aware strategies are concerned, we consider a latency-aware lu/la strategy adapting [12] from file sharing to P2P-TV applications, a bandwidth-aware lu/ba strategy [104], and a power-aware lu/pa strategy [93] (i.e., where power is the ratio of bandwidth to latency B/L). Selection is performed by measuring the property of each peer, which are then ranked according to the property value (e.g., low latency, high bandwidth or power) and selected *probabilistically* (i.e., not in strict order), with a probability that decreases with increasing ranking.

Intuitively, lu/r aims at keeping the playout delay from the source as low as possible by diffusing the most recent chunk at their disposal (i.e., the latest in their buffermap $B(p)$). We consider the simple ru/r for reference purposes, and lu/r as it is proven to be optimal in ideal homogeneous settings [17]. Network-aware $lu/\{la, ba, pa\}$ schedulers [12, 93, 104] are instead expected to enhance performance beyond lu/r, especially in case of heterogeneous realistic scenarios: in more details, lu/la aims at locally confining the traffic by proximity peer selection, lu/ba aims at reducing the chunk diffusion time by preferring peers with higher upload capacities and lu/pa aims at combining both benefits.

6.2.3 L3/L7 Interaction

Finally, the efficiency of scheduling decisions is possibly perturbed by errors affecting (i) the precision of network property measurements or (ii) the fate of control information exchanged by peers, that have largely been neglected in the reference work we consider [12, 93, 104]

On the one hand, (i) network-aware schedulers base their chunk-scheduling decision on properties concerning neighbors and possibly the underlying network conditions (e.g. path RTT, available bandwidth, peer upload capacity BW_U, etc.). Such properties can either be retrieved through an "oracle" entity (such as an IETF ALTO [6] compliant server in an ISPs), or directly measured by peers themselves. Direct measurement can be rather imprecise for several reasons (e.g. cross traffic, OS scheduling, NICs interrupt coalescing, unexpected interaction between simultaneous measurement probes, etc.), which can in turn lead to unfaithful neighborhood representation and wrong scheduling decisions. In order to assess the impact of measurement errors without being bound to specific measurement techniques, we resort to a high-level model where the measurement process is controlled by a single parameter α describing the magnitudo of the error.

On the other hand, (ii) control information can be not timely disseminated, or event lost, at L3. Indeed, in case of gossiping algorithms using UDP, such information would not be retransmitted, distorting thus the vision that each peer has of the system state. Inconsistency can also be due to slow dissemination of control information (e.g., a system may wish to limit the amount of signaling traffic injected at L3 by reducing the refresh rate of control information exchange). Considering mesh-push P2P-TV systems, both types of errors translate into out-of-date knowledge concerning neighbors' buffer maps: in this case, a peer may decide to schedule the transmission of a chunk even if the destination has already received that chunk, resulting in an unnecessary chunk transmission (i.e., a chunk collision). In order to assess the impact of signaling without being bound to specific algorithms (nor to their settings), we resort to a high-level abstraction, and model errors due to packet loss or out-of-date system knowledge as error on the buffer-maps.

6.2.4 Preliminary studies: simulation parameters

This section describes our careful selection of some crucial parameters of our simulations campaign, such as (i) the population size and stability, (ii) the sharing ratio and (iii) the upload bandwidth. We stress that our choice was to gather simulation scenarios that are as representative as possible of the Internet and real-life: hence, we used own measurement, or other relevant measurement performed by colleagues in the scientific community, to derive a realistic set of simulation parameters. For simplicity reasons preliminary simulations of sections 6.2.4.2 and 6.2.4.3 have been realized with $N_H = 2000$ homogeneous peers, organized in a random graph with mean degree $d_0 \simeq 20$ and trading a video stream composed of 1500 chunks; latency matrix is fixed according to the Meridian dataset [38, 69] as well as chunk-size C that is fixed at 100kbit according to [40, 59]; buffer maps can store 50 chunks and statistics are collected starting from the 500th chunk to avoid initial transient.

6.2.4.1 Population size and stability in real P2P-TV systems

To justify our assumption of absence of churn in the population, we show in Fig. 6.2 the stability of a real user base population in the operational network of a major European ISP that we continuously monitor in the context of the NAPA-WINE project [55]. For the same project, we have developed state of art P2P-TV classifiers, that are either based on the stochastic analysis of the packet payload (KISS [32]) or on the behavioral analysis of the connection pattern (Abacus [114]). The classifiers are comparably accurate [33, 34] and an open source implementation
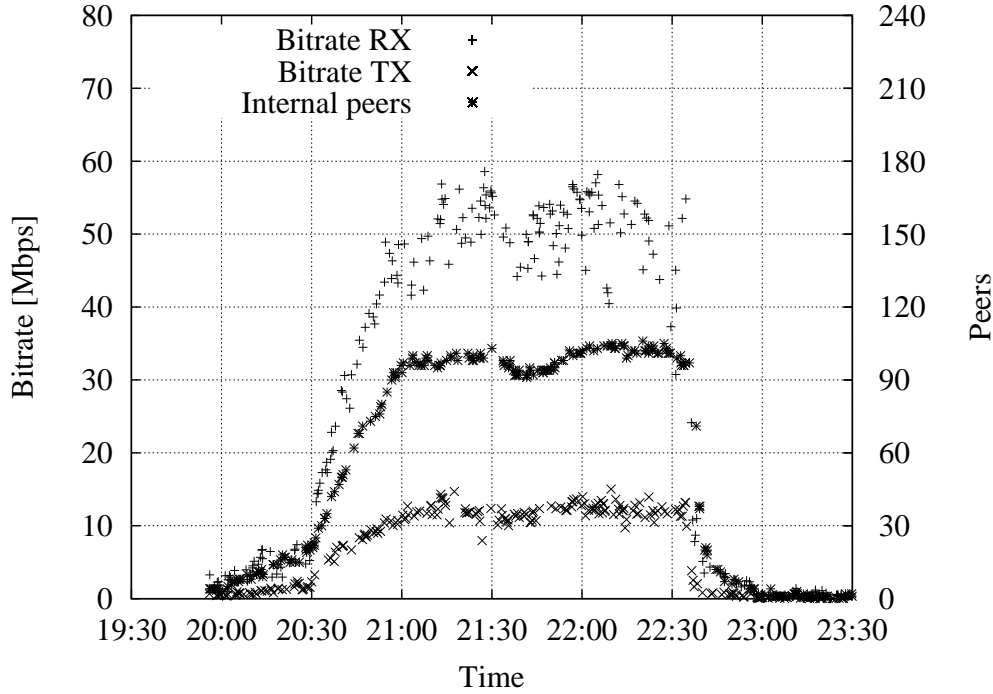
Figure 6.2: Temporal evolution of the number of peers, received and sent traffic volume during a typical P2P-TV event (SopCast application), at a PoP of a major European ISP that we continuously monitor.

is available at [48]. We run the classifiers on several probes in different major European ISP, so that we are able to recognize the traffic of popular applications such as PPlive [89], SopCast [107], tvants [112], as they are used by real user in operational networks.

While in general the usage of P2P-TV application is episodic, as it is driven by a specific program –rather typically, a sport event– *during* the event the population remains *extremely stable*. We support this statement with the help of Fig. 6.2, which reports the temporal evolution of the number of peers, depicted with a star point on the right y-axis, during a typical sport event streamed by the popular SopCast [107] application gathered during a Championship match in April 2009. Each point in the picture reports a measurement related to 5 seconds, and we sub-sample the observation points for the sake of readability. The picture also reports, on the left x-axis, the received (plus sign) and sent (cross sign) bitrate in Mbps. As it can be seen from the picture, peers arrive in a flash-crowd pattern starting from 20h30 (thus prior that the match begins), while during the whole 1h30-long soccer match the peer population keeps extremely stable to about 100 peers (i.e., the value that we actually simulated). Then, immediately after the end of the match, peers rapidly depart and the system empties. Also, noticeable from the picture, the traffic contributed by the peer behind the residential point of presence (PoP) is lower than the received traffic, which is due to the asymmetry typical of ADSL lines.

This pattern is very common in our measurements and justify our environment choice of *absence of churn*. First, by focusing on steady-state performance of different algorithms, we gather results that are clearly statistically more significant that performance in the transient phase (i.e., during arrival or departure). Second, it is likely that the algorithm exhibiting the best performance in steady-state, will also be the best candidate in the transient phase. Finally, users are clearly

interested in the QoS during the match, while they are likely less interested in the system performance prior that the match begins: indeed, notice how arrival time roughly uniformly distributes in the 30 minutes preceding the match, which is more likely tied to user "warm-up" of the channel (i.e., joining the P2P-TV system in advance to be sure seeing the first kick of the match) and personal habits rather than reflecting actual interactive usage of the channel.

6.2.4.2 Sharing ratio

Sharing ratio k is a fundamental and critical parameter in every peer-to-peer system since it describes its capacity to disseminate data to all nodes: k is defined as

$$k = \frac{\overline{BW_U}}{\lambda_{video}} = \frac{\displaystyle\sum_{i \in N_H} BW_{Ui}}{N_H \cdot \lambda_{video}}$$

or the ratio between the total uplink capacity (i.e. the sum of uplink capacity of all nodes) and total inbound traffic; intuitively a value of $k > 1$ means that nodes uplink capacity is sufficient for the system to be self-sustainable as long as algorithms are good enough to exploit capacity. On the contrary, in a scenario where k is < 1, total up-link capacity is not high enough to guarantee that every peer receives the entire stream. Notice that, as the value of k provided by common ADSL peers in actual system can be as low as 0.2 [19], the correct working of the system is guaranteed by the presence of "amplifier" nodes providing the missing capacity.

Notice that scheduling algorithm may not be able to succesfully exploit the available system capacity even for $k > 1$. For simplicity reasons, in this section we show results collected from simulation of homogeneous swarms of peers. Figure 6.3 shows the cumulative distribution function (CDF) of chunk delays experienced by each peer, or in other words, the temporal gap between the emission of a chunk at the source and its reception at a given peer. Notice that the gray shaded zone indicates the percentage of lost chunk (i.e., those who have been actually lost at L3 or arrived after the playout deadline). These results refer to a scenario in which the source generates video at 1Gbps (i.e.chunk generation rate of 10 per second) and we modify the value of k by varying the upload bandwidth of peers BW_U.

Top plot of Fig. 6.3 shows the CDF of the chunk delay for the naïve *ru/r* scheduler, where it can be seen that for $k = 1$, almost half of chunks are lost and even when the capacity is twice $k = 2$ the needed sustainable rate, the system still experiences a non negligible amount of 1.4% losses.

In case more sophisticated schedulers are used, such as the power-aware lu/pa in the bottom plot of Fig. 6.3, we notice that the system is almost lossless for a sharing ratio of $k = 1.5$. Hence, in our simulation we design the class population to match the factor $k = 1.5$, so that the system is self-sustainable, which allows to isolate the impact of other factors (e.g., L3 network, L7 schedulers, L3/L7 interactions) on an otherwise lossless system.

6.2.4.3 Mean upload bandwidth

In previous section 6.2.4.2 we fixed a value for parameter k, binding video rate to mean upload bandwidth: hence, we need to investigate sound values for the peer upload capacity. To the best of our knowledge, there are no studies or tools, as for instance the Meridian project for latencies, which estimate end-host bandwidth distribution. Moreover, even if there were any, the gathered data would likely be strongly dependent on countries or service provider. For these reasons we perform a survey, to gather rough boundaries for the uplink capacity, as well as a sensitivity analysis, to gather reliable simulation results between these boundaries.
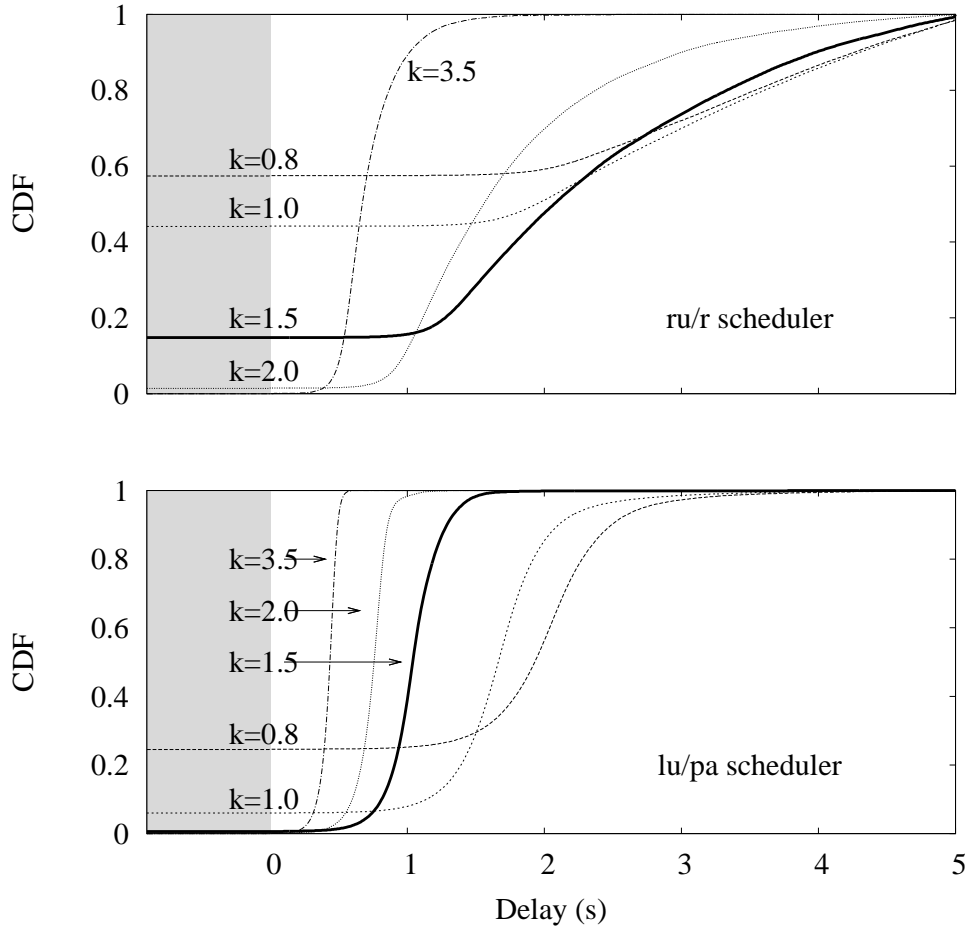
Figure 6.3: Delay distribution of ru/r and lu/pa schedulers for different values of the sharing ratio k. Notice that the gray shaded zone indicates lost chunks.

Again, as in the previous section, we consider an homogeneous peer population, where each peer belongs to a single class with upload bandwidth BW_U. We perform a sensitivity analysis by carrying on several simulations varying the relative magnitude of the propagation and transmission time, by varying BW_U. Notice moreover that latencies, chunk-size and buffer-map size are kept constant while video rate and playout deadline vary according to BW_U[3]. In more detail, we denote by γ the ratio between the propagation delay and the chunk upload time:

$$\gamma = \frac{\overline{M}}{t_{TX}} = \overline{M} \frac{BW_U}{C} \tag{6.1}$$

where $\overline{M}$ is the average Meridian end-to-end latency, t_{TX} is the chunk transmission time and C the chunk size. Intuitively γ indicates which component of the delay has the largest influences on the total chunk transfer time. For instance, when $\gamma < 1$ the transmission delay is greater than the propagation delay, so that the total chunk delay reduces in case of bandwidth-awareness; conversely, when $\gamma > 1$, propagation delay is the largest component of total chunk transmission time, which would thus reduce in case of latency-awareness.

---

[3]Video rate is bound to BW_U via parameter k and playout deadline takes into account video rate, the fixed chunk size C and the buffermap size.

Mean upload bandwidth (Mbit/s)

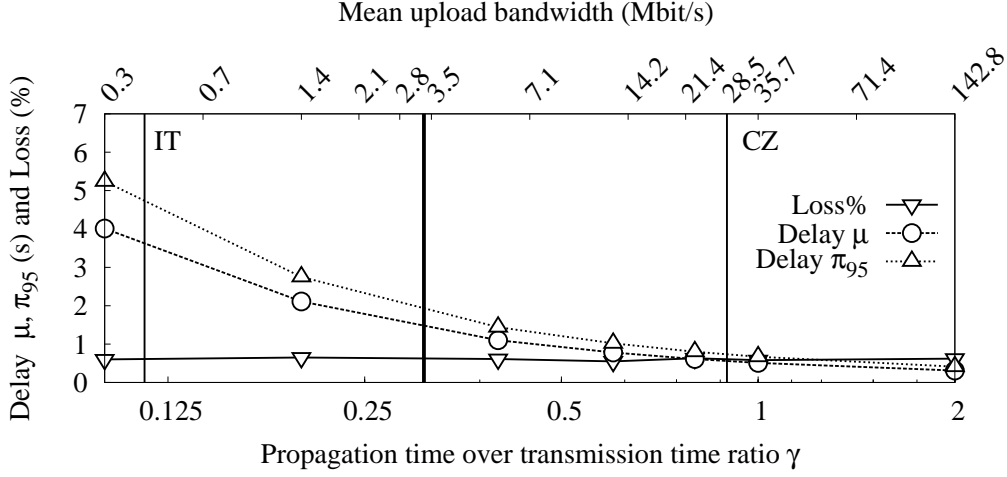Figure 6.4: Sensitivity analysis of delay and losses as a function of the propagation-delay over transmission-delay ratio γ. Top x-axis represent the average per-country bandwidth reported in [9], with landmarks for the lower (Italy, IT) and upper (Czech republich, FR) bounds; with a thick line, we report the $\overline{BW_U}$ value we selected in this chapter.

To gather valid boundaries for γ we surveyed the average uplink capacity of different European countries [9], to gather upper (CZ) and lower (IT) bounds of $\overline{BW_U}$ (and, hence, of γ). Fig. 6.4 depicts several performance metrics (i.e., chunk losses, mean and 95th percentile of the total chunk delay) as a function of γ varying in the range resulting from the survey. As γ grows, we notice a smooth decrease of the delay curves, which is an expected consequence of the transmission time reduction due to higher upload bandwidth.

The vertical thick line in between IT and CZ references represent the working point that will be used for all our simulation, which can be thought to represents an average European country. Two considerations hold: first, notice that, as loss rate remains steady over the whole interval, we can expect the results that will be shown in this chapter, to hold for a number of different European countries. Second, we gather that, as γ varies in the selected range, the delay can vary by almost one order of magnitude: hence, our simulation results should be considered as representative of an average country, and we can expect delay results to (roughly linearly) vary depending on the actual value of $\overline{BW_U}$ in the country of interest.

6.3 Simulation Results: Impact of L7 and L3

In this section, since we have fixed all the crucial parameter of the simulator, we begin the actual study of the impact of L7 and L3 factors on the system performance: we first analyze the impact of chunk scheduler and topology manager, and then evaluate the impact of L3 topologies on the system performance.

Simulations have been performed according to the following general settings. For each parameter under investigation, simulations are averaged over 6 repetitions: specifically, we consider 3 different instances of 2 different overlay graphs as described in section 6.2.2.2[4]. Unless otherwise stated, we use the Meridian dataset [38, 69] as a default realistic model of L3 network latencies with $\overline{M} = 35$ ms.

---

[4]Since impact of different overlay topologies at time $t = 0$ is not appreciable, we do not breakdown results according to them
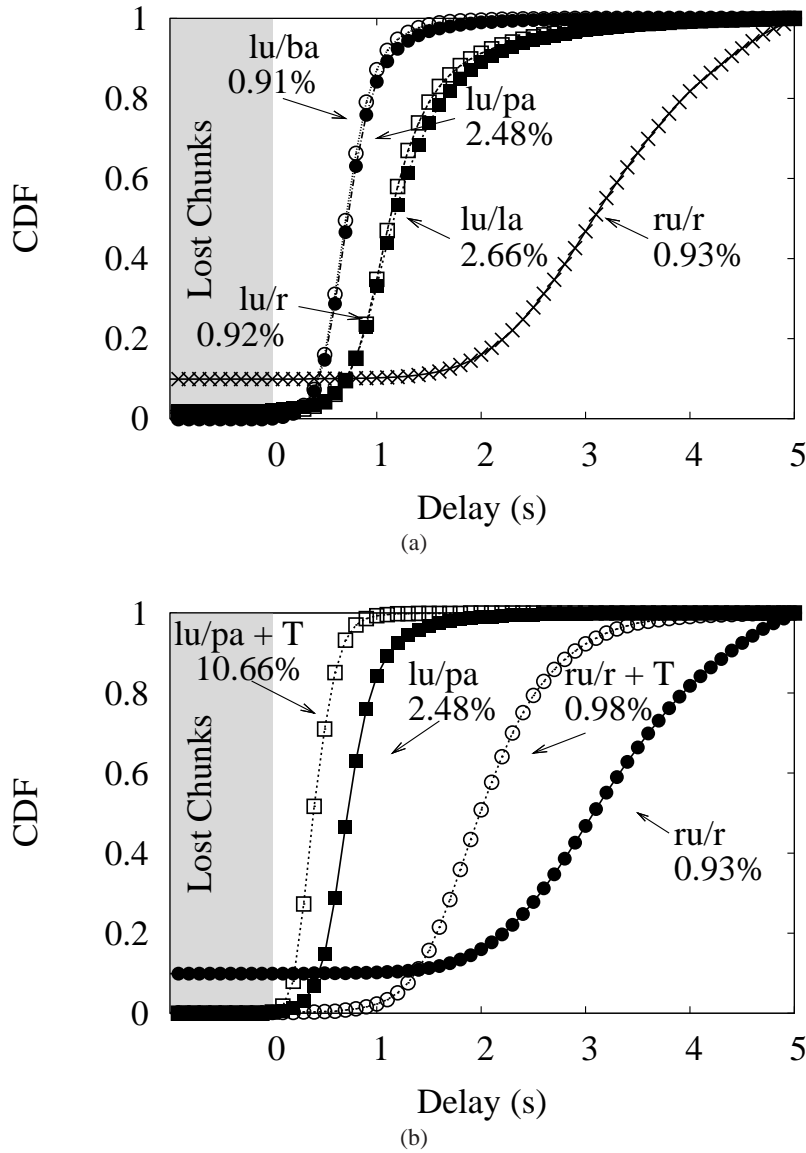
Figure 6.5: Cumulative distribution function of chunk delay: (a) Performance of different schedulers (ru/r and $lu/\{r, la, ba, pa\}$) on a Meridian network. (b) Effects of topology management for best (lu/pa) and worst (ru/r) schedulers. Labels along the curves in (a) and (b) express the traffic proximity percentage $P\%$.

Each overlay consists of $N_H = 2000$ peers, of which we simulate a lifetime of 150 seconds, during which 1500 chunks of video stream are disseminated in the overlay. We consider a single source node that streams video at an average rate of 1 Mbps, and consider 100 kbit fixed-size chunks (i.e., 10 new chunks are generated in each second). Statistics are collected starting from 500th chunk in order to avoid the initial transient. We consider that buffer maps store 50 chunks, which correspond to a playout delay of 5 seconds.

6.3.1 L7 Algorithms

6.3.1.1 Schedulers

Curves in Fig. 6.5(a) follow the same semantic of those presented in section 6.2.4.2. Each curve represents a different scheduler, and we indicate lost chunks (i.e., chunks that arrived later than the playout deadline) as chunk with negative delay (i.e., falling into the gray shaded zone): this is especially visible for the simplest scheduler ru/r, where the fraction of lost chunks exceeds 10%. The picture further reports the traffic-locality $P\%$ percentage along each curve. Recall that $P\%$ represents the fraction of chunks that do not traverse the core network (i.e., the destination host is attached to the same router of the sender host), and is thus a rough indication of network friendliness.

With the exception of ru/r, other schedulers limit the fraction of lost chunk (which is very close to 0%), but instead differ by chunk delay and locality $P\%$ measures. Considering lu/r and lu/la, both strategies select the latest chunk and send it to peers which do not own it: lu/r selects a destination peer at random, while lu/la proportionally prefers closer neighbors. Clearly, locality improves when latency-aware lu/la peer selection is performed with respect to lu/r (from 0.92% to 2.66%). Recall that two hosts attached to the same router sense a latency of 0 so when a scheduler take into account latency information, hosts behind the same router are likely to be chosen. At the same time, notice that lu/r and lu/la are very close in terms of delay, despite lu/la preference for low latency neighbors. This can be explained with the fact that the propagation delay has a less prominent impact with respect to transmission delay, especially considering that chunks possibly travel multiple hops on low-capacity access links.

Consider indeed that the average propagation delay between any two peers is $\overline{M} = 35\,\text{ms}$, whereas from Tab. 6.1 we have that the average chunk upload times range from 20 ms for class-I peers to 200 ms for class-III peers. This entails that, at each hop, the transmission time likely plays a great role in determining the chunk delay performance: thus, merely choosing a peer which is closer in terms of the propagation delay does not allow to improve the overall system chunk delay performance.

Finally, the lu/ba and lu/pa schedulers achieve the best delay performance. Consider that both lu/ba and lu/pa assign scores according to the destination upload bandwidth, with the power-aware lu/pa scheme taking into account the propagation latency as well. Results confirm that uploading chunks to high-capacity peers, which can in turn diffuse them fast, is beneficial to the whole system [104]. Moreover, we further gather confirmation of the fact that explicitly taking into account node latency improves locality $P\%$ but does not further ameliorate delay performance.

In addition, an interesting aspect emerges from this analysis: comparing the $k = 1.5$ homogeneous single-class system of Fig. 6.3 in section 6.2.4.2 to the heterogeneous multi-class system shown in Fig. 6.5(a), we see that peer heterogeneity plays a non marginal role in improving global efficiency [64]. Intuitively, having peers with different capacity is beneficial because high capacity peers, which can handle a greater number of active connections, will occupy the high portion of the istantaneous chunk distribution tree, close to the source, possibly as a result of topology management. On the other hand, peers with poor connectivity can occupy far positions in each chunk distribution tree, since they can serve a lower number of neighbors (or even none).

Overall, preference toward high-bandwidth peers is necessary to reduce the delay incurred by chunks; instead, preference toward low-latency peers is not helpful in reducing the chunk delay, but may ameliorate the network friendliness confining the traffic at the access.

6.3.1.2   Topology management

We now investigate the impact of topology management on the system performance. To gather performance bounds for a large class of schedulers, we consider lu/pa as *upper-bound* (since it exhibits the best results in terms of both delays and locality) and the simple ru/r as *lower-bound*

In their overlay maintenance process, peers have the chance to tune their neighborhood, both in terms of its *size* (i.e., change their out-degree) and *composition* (i.e., preferring high-bandwidth peer as in BitTorrent tit-for-tat, or trying new peer as in BitTorrent unchocking). For the sake of simplicity, we consider topology management as a feature that can be turned on or off, and select thus a single algorithm: specifically, we use the approach described in [59], which we refer the reader to, for a detailed description. Briefly, according to [59] peers continuously vary their neighbor size, selecting peers according to a "desirability" function that depends on the neighbor upload bandwidth: as we previously observed in the case of chunk scheduling, bandwidth-awareness is extremely beneficial (specifically, more beneficial than latency or power-awareness) to the overall system performance, hence our selection.

In Fig. 6.5(b), improvements induced by the topology manager are clearly noticeable for both schedulers. Taking into account ru/r, for instance, we notice a significant amelioration in terms of both losses and delays. According to [59], high-bandwidth peers have an higher fan-out, and tend to select high-bandwidth nodes in their neighborhood: in this way, chunks generated by the source will be first sent to nodes that are capable of spreading them faster, thus reducing the overall delay and, by consequence, loss rates. In other words, performance enhancements are due to the fact that high-capacity nodes "moves" up toward the source in the instantaneous chunk diffusion tree. In Fig. 6.5(b), the same scheduler ru/r can lower the mean delay by 0.75 seconds with an improvement of 25% and reduce losses to 0.25% by performing topology management.

Yet, improvements can be achieved in case of lu/pa as well: e.g., the 99th percentile of the delay reduces by 50% reaching 1.1 second. Moreover, notice that several beneficial effects combine altogether: indeed, despite being based on bandwidth-awareness only, topology management consistently increases the fraction of the traffic confined in the access network as well ($P\% = 10.7\%$). Indeed, consider that the improved neighborhood is composed mostly by high-bandwidth nodes: the power-aware scheduler is then able to select among closer nodes, that are also higher capacity than in the previous case. Overall, this yields higher odds to choose high-capacity nodes that are also connected to the same router, and as high-capacity peers can offer a larger amount of data in the same time window, traffic locality increases as well.

Summarizing, active topology management is beneficial, as it increases the chances to find higher capacity peers, thus lowering the chunk delay and hence reducing losses.

6.3.2   L3 Network

We now assess the impact of the following models for the underlaying L3 network, each of which assigns latencies between access routers in a specific way:

- **Ideal**: This represents an L3 network without latencies, or in other words, propagation delay is 0, so that in practice only the logical L7 overlay topology is taken into account.

- **Meridian**: Latencies provided by the Meridian project [38,69], where realistic latencies are gathered by means of end-to-end Internet measurements (the mean latency of the Meridian dataset equals to $\overline{M} = 35$ ms as most of peers in the data-set are from US).

- **Constant**: Latencies are constant and equal among all end-to-end paths, and the latency value is equal to the mean value $\overline{M}$ of Meridian latencies.
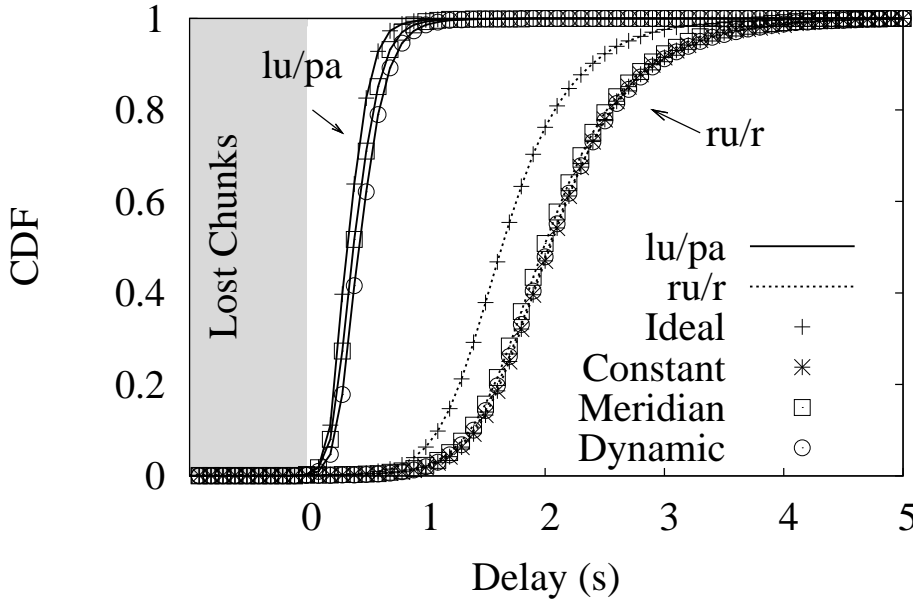
Figure 6.6: Cumulative distribution function of chunk delay: Impact of different models of L3 networks, for best (lu/pa) and worst (ru/r) scheduler.

- **Dynamic**: Latencies between any two pairs of routers are distributed according to an Exponential distribution, whose mean is fitted using the mean value $\overline{M}$ of Meridian latencies; to simulate the effect of cross traffic, yielding to different levels of congestion on a chunk duration timescale, new values of latency are extracted for each new chunk propagation.

The network models we consider range from a simplistic *Ideal* model to the realistic *Meridian* one, where one would expect network-aware algorithms to stand from the lot. The *Constant* network model is a simple, still unrealistic model, that is however fitted on real data: notice that we expect latency-aware algorithms to be ineffective in this case. Finally, we include the *Dynamic* network model as a worst case for latency-aware algorithms such as lu/pa, since the decisions are taken on the basis of measurements that however continuously change (so that each chunk between any two peers will *always* experience different latencies).

In Fig. 6.6, we show the CDF of delays for both ru/r and lu/pa schedulers using different topologies, when the topology management feature is enabled (since similar considerations hold, we avoid reporting the case where topology management is disabled). From Fig. 6.6 we notice that the performance of each scheduler remains clearly separated, and further that the L3 network model only minimally affects the chunk delay performance (i.e., given any scheduler, curves are tightly clustered across all models).

In case of ru/r scheduling, ideal L3 network exhibits remarkably lower delays than the other network models, which all have a very similar impact on the delay performance. In case of lu/pa scheduling, we instead remark that performance figures are very tight irrespectively of the network model, which surprisingly holds even for the dynamic network scenario. Again, this is due to the predominant impact of transmission delay over propagation delay, entailing that even a rather simple network model (e.g., non-null delay fitted over real measurement), yields consistent performance estimates.

Notice that this might change in case the transmission and propagation delay are comparable, which can happen when e.g., the uplink bandwidth increase, or the chunk size shrinks. Still, as

small chunk implies higher overhead (i.e., due to increased signaling rate, buffer map size, etc.) we can expect the relationship between transmission and propagation delay to hold for a while. Notice also that, although lu/pa performance are not affected by the network model given the current peer population breakdown, in future scenarios where most of nodes have high-capacity, we instead forecast an increased importance of latency-awareness over bandwidth-awareness. In this case, we may also expect the dynamic network case to constitute a stiffer scenario.

Summarizing, the impact of L3 network models is almost negligible: in case access network is the bottleneck, the chunk transmission time largely dominates the delay end-to-end delay (at least for reasonable chunk sizes and the current access rates).

6.4   Simulation Results: L3/L7 Interaction

In this section, we investigate the impact of L7/L3 interaction on the system performance: first we study latency and capacity estimation errors, afterwards we analyze the effect of signaling errors. In this section we take into account exclusively the lu/pa scheduler since it is the one that presented best performance on previous scenarios and it is sensible to both bandwidth and latency information. Experiments of L3/L7 interaction have been done for the other network aware schedulers as well and lu/pa always performed better; we do not include results of other schedulers because we think their inclusion would not bring added vale to our study.

6.4.1   Measurements Errors

P2P-TV systems need to implement measurement techniques in order to successfully implement both topology management policies and network-aware scheduling algorithms such as $lu/\{la, ba, pa\}$. In a real deployment, both latency-related (e.g., one-way delay or RTT-latency) and capacity-related (e.g., bottleneck capacity, available bandwidth) measurements will be affected by some degree of errors, that are either intrinsic to the measurement techniques, or depend on temporary network conditions: our aim is thus to evaluate the robustness of P2P-TV systems to such errors.

6.4.1.1   Latency measurement errors

Consider latency first, which is generally simple to estimate, and focus on the minimum RTT estimation, which is especially simple since it does not need clock-synchronizaton. In case RTT measurements can passively exploit the continuous transmission of data/acknowledgment pairs, this yields to many samples and thus to robust estimates. However, there are cases where RTT measurements are needed *prior* that any data transmission happened: in this case, active measurement are needed which yields to fewer samples and thus to possibly biased the RTT estimation. Specifically, we consider that RTT can only be *over-estimated* (e.g., since the acknowledgment packet may be delayed due to cross-traffic, self-induced congestion at the access, sustained CPU load in the host machine running the P2P application, etc.), so that a nearby peer can be mistaken as a faraway one. Formally, denote by $\Delta(p, p')$ the actual round trip time latency between p and p', and consider that peer p will measure a $\tilde{\Delta}(p, p') \simeq \Delta(p, p') + \alpha$, where α represent the error intensity. $\tilde{\Delta}(p, p')$ is modeled by a random variable, that follows a negative exponential distribution with mean $\overline{M}\alpha$.

Figure 6.7 reports system performance as a function of increasing error intensity α. Specifically, x-axis reports the ration of overestimation error between the measured and the actual latency
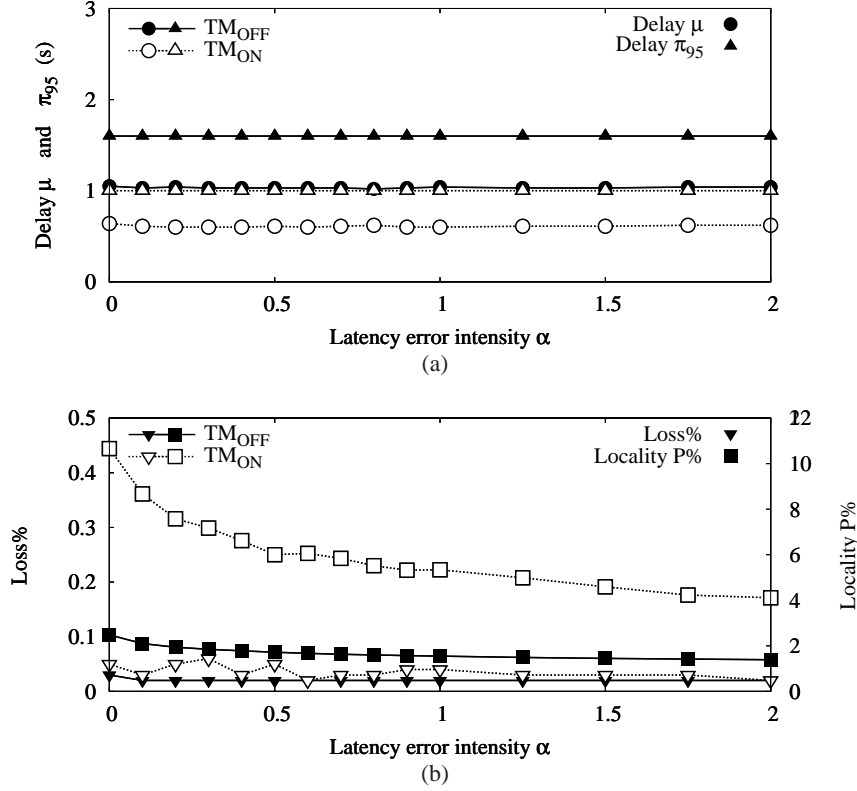
Figure 6.7: L3/L7 Interaction: Impact of latency measurement error. (a) reports chunk delay distribution μ and π_{95}, (b) reports $Loss\%$ and $Locality P\%$.

value, varying from $\alpha = 0$ to $\alpha = 2$ (measured $\tilde{\Delta}$ is, on average, the double of the actual Δ). Pictures report both user-centric (i.e., mean μ and 95th percentile π_{95} of the delay, Loss %) and network-centric metrics (i.e., percentage of local traffic $P\%$). In reason of our previous observation on the lower impact of the propagation delay component, it is not surprising to observe that user-centric performance are not affected by latency errors. Conversely, network-centric performance is deeply affected by measurement errors: as the scheduler knowledge of its neighborhood becomes erroneous, its proximity-aware choices are no longer correct, and the traffic locality index significantly decays. Notice that a qualitatively similar behavior holds both in presence or absence of topology management.

Overall, latency estimation precision has a small impact on P2P-TV performance, limitedly affecting the overlay ability to localize the traffic.

6.4.1.2 Capacity measurement errors

Let us now consider capacity measurement. In this case, we expect measurement errors to have a possibly larger impact on the system performance, since the transmission delay component (which depends on the uplink capacity) plays a major role in determining the chunk delay. At the same time, capacity measurement are notoriously difficult, as several techniques typically yield rather different measurement [103]. Also, unlike the previous case, capacity can be either under-estimated or over-estimated (depending on the considered technique, due to cross traffic, etc.). In the case of P2P, this gets further complicated as concurrent measurements have mutual influence [25], which further adds to the error. Finally, in case of P2P-TV, the uplink capacity
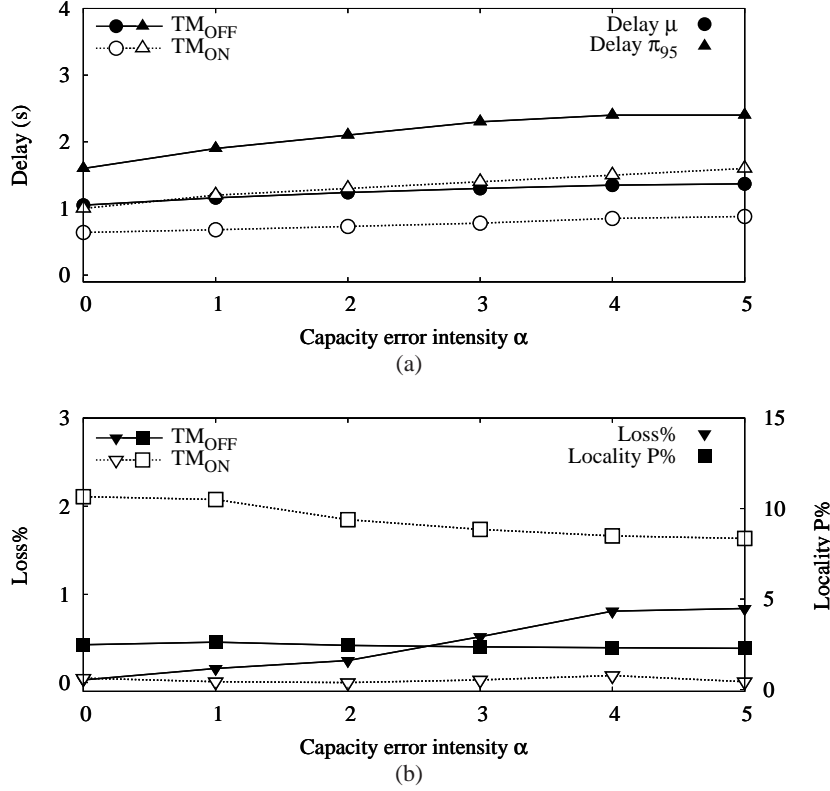
Figure 6.8: L3/L7 Interaction: Impact of capacity measurement error. (a) reports chunk delay distribution μ and π_{95}, (b) reports $Loss\%$ and $LocalityP\%$.

measure of the *receiver peer* is generally needed, which is not straightforward since the measuring peer cannot rely on the chunk transmission process. For the above reasons, we can expect capacity measurement errors to be larger in magnitude with respect to latency errors. Formally, we denote by $C(p', p)$ the actual bottleneck capacity in the path from p' to p; we consider that peer p will erroneously estimate $C(p', p)$ as $\tilde{C} \sim \mathrm{N}(C(p', p), \alpha C)$: in other words, peers capacity estimate is a normally distributed variable, with mean equal to the actual bottleneck capacity $C(p', p)$ and a variance equal to αC.

In Fig. 6.8 we explore errors ranging in $\alpha \in [0, 5]$: despite we consider such a large range, we observe that performance is rather robust: indeed, only a marginal increase of latency (and of losses, in case topology management is disabled) can be observed. This can be explained with the fact that, provided that measurement errors still allow to clearly separate the peers in classes [5], performance of bandwidth-aware algorithms remains consistent. Given the challenges in capacity and bandwidth measurement, this intrinsic robustness is a very advantageous feature, as a rough binary discrimination capability in high-capacity vs low-capacity peers may be enough for network-aware algorithms.

Overall, capacity estimation precision has a small impact on P2P-TV performance: indeed, provided that peer classes are sufficiently separated, it is always possible to differentiate among classes even in presence of measurement errors.

---

[5]Measurements are correct with a very coarse granularity, from Tab. 6.1 we have that $\frac{BW_U^i}{BW_U^{i+1}} > 2 \forall i$ which means that we may correctly separate peers into classes even when the measurement precision is rather poor.

6.4.2 Signaling errors

We now investigate the effect of signaling errors on the system performance. Recall that, in order to send chunks that are *useful* for the receivers, each peer must have a precise knowledge of its neighbor buffer-maps. This can be accomplished by periodically exchanging buffer-map status in control packets, or by piggybacking buffer maps in data packets. In order for this knowledge to be as up-to-date as possible, any peer p should inform all of its neighbors as soon as it receives a new chunk c. Still, even in this "perfect signaling system", due to the unavoidable latency of the signaling process (both propagation and transmission delay), it is still possible that p schedules the transmission of a chunk c to a peer p' that has just received it (but not sent its buffer map $B(p')$ out yet), thus generating a collision. Furthermore, loss of signaling messages can happen in the L3 network, further degrading the quality of peers knowledge. Clearly, frequent buffer-map exchange has a high cost in terms of overhead: as several new chunks are generated at each second, the signaling process would thus need to be continuous in order for peers to have an up-to-date view of their neighborhood. However, lowering the signaling rate to reduce the messaging overhead also increases the chance for collisions to occur.

6.4.2.1 Impact on L7

Up to now, we have evaluated network-aware P2P-TV systems performance by assuming that peers have a perfect instantaneous knowledge of the buffer maps of their neighbors – a rather unrealistic assumption. We therefore model the impact of low signaling rates (or signaling messages losses at L3) as a quality degradation of system state knowledge in the distributed P2P system. In more detail, we model imprecision of system state knowledge as "usefulness" errors: in other words, with a given probability P_{err} a peer p can take a scheduling decision of chunk c toward p' which he believes to be useful (i.e., $c \notin B(p')$) despite it is not (i.e., $c \in B(p')$), which generates a collision.

Conversely, we do not consider the opposite kind of errors (i.e., p believes $c \in B(p')$ despite actually $c \notin B(p')$), as this would indeed model a somewhat unlikely case of *misconfigured* peers sending erroneous updates (i.e., advertising chunk $c \in B(p')$ to be available while it is not).

Fig. 6.9 shows the mean μ and 95th percentile π_{95} delay, along with chunk loss statistics. Notice that while the mean delay is roughly unaffected by signaling error probability P_{err}, a counterintuitive phenomenon characterizes the π_{95} measure. Indeed, the 95th delay percentile increases until $P_{err} = 1/400$, and afterward starts decreasing: this behavior is strongly correlated to the chunk loss rate, which starts rising roughly at $P_{err} = 1/400$. What happens is that for increasing P_{err}, peers indeed receive chunks with higher delay, which in turns raises the probability that chunks arrive beyond the playout delay (i.e., delay larger than 5s), and are thus marked as lost: as lost chunks are not accounted in the delay curve, the peak is thus an artifact due to the playout deadline.

Traffic locality exhibits a non-straightforward behavior as well: indeed, it can be seen from Fig. 6.10 (left y-axis) that locality increases as buffer-map errors increase as well, which is especially visible in case of topology management. This can be explained by considering that the lu/pa scheduler preferentially selects nearby high-capacity peers. When the error probability is low, these peers will be fed first, but then, as peers rarely fail in estimating the usefulness of their decisions, other lower-capacity higher-latency peers get successfully served during the remaining upload slots. Conversely, when error probability is high, the scheduler will keep on sending chunks to close high-capacity neighbors, despite they likely already have received that chunk from other peers (notice that we forbid peers sending the same chunk to the same peer multiple times).
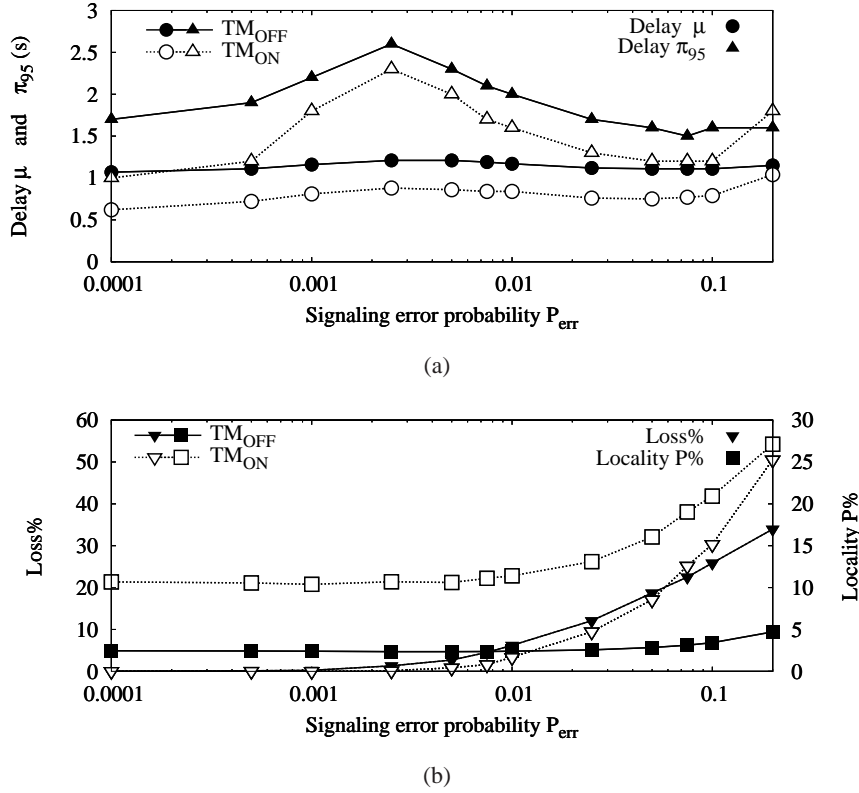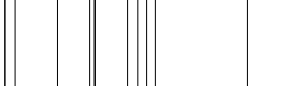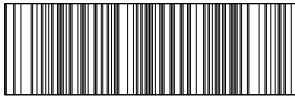
(a)

(b)

Figure 6.9: Delay and chunk losses as a function of signaling errors. (a) reports chunk delay distribution μ and π_{95}, (b) reports $Loss\%$ and locality $P\%$.

Overall, system performance are extremely sensitive to errors due to stale signaling: effects are noticeable on the tail of the delay distribution for error rates as low as $P_{err} = 1/1000$ and loss probability become excessive for error rates as low as $P_{err} = 1/100$.

6.4.2.2   Impact on Quality of Experience (QoE)

We then dig the user Quality of Experience (QoE) by evaluating an objective video quality metric, namely the Peak Signal to Noise Ratio (PSNR). We consider the standard Soccer sequence (H624 format, CIF resolution, 300 frames @30Hz, looped for the whole simulation duration), and record for each peer the list of lost chunks. We then make use of Evalvid [49] to evaluate video quality, by feeding the tool with the video sequences where we take into account the chunk loss pattern for each peer. To give the reader an intuition of PSNR value, and how it roughly corresponds to other system performance, we report in Tab. 6.3 a few examples. Each row refers to either the video source, a peer of class-II, or a peer of class-IV, and reports different performance metrics such as delay (mean and 95th percentile) and loss statistics,.with a graphical representation of the loss pattern, where each vertical bar corresponds to a loss. As PSNR evaluation is very time consuming and due to the size of our system, we resort to stratified sampling: specifically, we rank peers according to the amount of losses and select a 10-peers sample (corresponding to different loss amounts) out of the total $N_H = 2000$ peer population. Right y-axis of Fig. 6.10 reports the PSNR averaged over the 10-peers sample (bars report the standard deviation over the sample), which due to stratification is however representative of the whole population. It can be seen that PSNR drops as soon as a losses occur in the system: notice further that, since a PSNR<24 dB is

Table 6.3: PSNR values for the Source, a class-III and a class-IV peer

Class	PSNR	Delay $\mu(\pi)$	Lost chunks	Loss pattern
Source	39.4	–	–	
II	16.1	1.18s (4.3s)	45 (2.2%)	
IV	12.1	1.28s (3.7s)	487 (24.3%)	
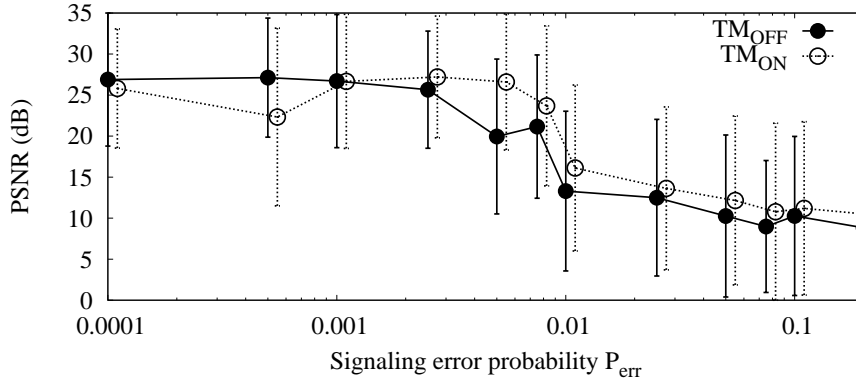
Figure 6.10: PSNR evaluation as a function of signaling errors. Bars represent the standard deviation of PSNR values gathered over the 10-sample population. PSNR of the original video sequence at the source is 35.6 dB.

generally considered as an indicator of extremely bad video quality, this suggest that buffer-map errors should be kept below $\mathrm{P}_{err} < 1/100$.

Overall, as in our evaluation buffer maps hold 50 chunks and a new chunk is generated every 100ms, this suggests that the signaling rate should be about as high as the chunk generation rate.

6.4.2.3 Impact on Peer Class

For the sake of completeness, we analyze how system performance vary across the different classes, comparing ideal (P_{err}=0%) and harsh (P_{err}=5%) settings, so to gather performance bounds. To quantify the impact of P_{err}, we define the *intra-class degradation factor* as:

$$\mathcal{D}(X) = X(\mathrm{P}_{err} = 5\%|c)/X(\mathrm{P}_{err} = 0\%|c) \tag{6.2}$$

where X is any metric considered in the table and c the considered class: intuitively, $\mathcal{D}(X)$ is a compact indicator of the performance loss from ideal to realistic settings.

To quantify the fairness of the results among classes, we define a *inter-class fairness factor* as:

$$\mathcal{F}(X) = \max_c X(c|\mathrm{P}_{err})/\min_c X(c|\mathrm{P}_{err}) \tag{6.3}$$

Table 6.4: Per-class performance breakdown

	Ideal $P_{err} = 0\%$			Harsh $P_{err} = 5\%$			$\mathcal{D}$		
	μ	π_{95}	L%	μ	π_{95}	L%	μ	π_{95}	L%
I	0.4	0.6	10^{-3}	0.8	1.3	0.2	*1.9*	*2.1*	*92.2*
II	0.6	1.0	10^{-2}	0.8	1.1	4.4	*1.2*	*1.1*	*263.9*
III	0.7	1.1	10^{-2}	0.8	1.2	8.5	*1.1*	*1.2*	*1421.3*
IV	0.9	1.3	0.2	1.1	1.4	33.9	*1.2*	*1.1*	*198.6*
$\mathcal{F}$	*2.2*	*2.2*	*68.2*	*1.3*	*1.1*	*146.9*			

intuitively, $\mathcal{F}$=1 corresponds to comparable performance across classes, while the larger the value of $\mathcal{F}$>1, the larger the unfairness.

Results are reported in Tab. 6.4. Notice that, already on ideal settings, class-IV peers experience a delay twice than that of class-I peers, and about 68 times more losses (loss $L\%$ unfairness exceeds a factor of 146 in harsh settings). Concerning the degradation due to signaling error, we see that class-I experiences a larger delay degradation than other classes but a limited loss increase: in other words, average delay increases but not enough to exceed the playout deadline (and to cause losses). The opposite happens instead for class-II and III, whose delay . The extent of class-IV degradation is instead smaller, however the absolute amount of losses exceed 33%, which likely makes video quality unbearable.

We conclude that, even in absence of signaling errors, performance breakdown is unfair with respect to peer belonging to different classes. Under harsh signaling errors, delay become more fair among classes, with however extremely large loss rates for peers belonging to the poorer classes.

6.5   Conclusions

In this work, we compare different state-of-art "network-aware" P2P-TV systems, i.e., systems whose main algorithms (such as chunk selection and topology management) are based on informed decisions concerning the status of the network. We define a flexible framework, able to accommodate further aspects beyond to the one we focus on in this work, and perform a thorough simulation campaign: our purpose is to understand what are the main factors that affect P2P-TV performance, and to what extent performance degrades under realistic settings.

Our main findings can be summarized as follows. First, we find the impact of the L3 underlay network model to be modest, with a small performance gap between simple (e.g., constant and fixed delay) and realistic models (e.g., meridian latency or dynamic latencies). This owes to the fact that the propagation delay has a smaller impact with respect to transmission delay, especially considering the relative low-capacity of current scenario. At the same time, we can expect that as the access capacity increases, the impact of propagation latency may need to be reconsidered.

Second, we find that system performance is rather robust to errors in the measurement of peer properties. More precisely, provided that peers capacity is clearly separated, the ability to roughly discriminate high-capacity from low-capacity peer is sufficient to guarantee a good level of performance. Similarly, errors in the latency estimation only affect the traffic locality, but system performance are otherwise unaltered.

Finally, we find the impact of signaling errors to be, by far, the most important factor able

to significantly degrade the quality of P2P-TV services and is able to severely impact overlay performance already from very low intensities. This suggests P2P-TV protocol designers to pay special attention to signaling logic, in order to gather reliable estimate of the achievable system performance. As future work, we aim at studying the interplay between chunk size and buffer maps update rate and exploring the trade-off between the overhead caused by high signaling rate and the outdated knowledge of neighborhood buffer-maps due to low signaling rate.

Chapter 7

Emulation Analysis

It is desirable that P2P algorithms and protocols are tested before they can be deployed at large scale. Beta testing usually involves either (i) the deployment of small to large-scale testbeds, such as Grid5000 [36], where the environment is fully under control but not representative of real world dynamics, or (ii) the use of large-scale testing facilities, such PlanetLab [88] or OneLab [74], that benefit of the realism of the wild Internet, but lacks however of control.

Researchers face thus the following dilemma. On the one hand, their testbed results may be easily reproducible, but hardly representative of real-world performance: in this case, the large development and deployment effort invested in the testbed does not payoff, since the offered level of realism only slightly exceeds the one achievable by simulation. On the other hand, carrying on experiments over the wild Internet allows to gather realistic results, though in this case the experimental scenario is not under control and generally hardly reproducible. Loss of *control* means that it may be very hard to correlate the observed performance with their root cause, so that experimental results become hard to interpret. Loss of *reproducibility* –which has been a requirement of experimental science since Hipparchus (ca.190 BC – 120 BC) measurement on Earth axial precession– can further hinder cause-effect relationships, and is therefore not a favorable environment for beta testing.

Efforts such as ModelNet [113] offer a third way, enabling the control of the *core network topology* that is instead precluded in Grid5000, PlanetLab and OneLab. In this sense, Model-Net does not try to fully substitute to these existing experimental facilities, but rather to complement them. Indeed, ModelNet stands between the two approaches for being more realistic than Grid5000 or smaller testbeds and, at the same time, more controllable than PlanetLab; furthermore experiments on ModelNet can be reproducible (L3 topology, traffic condition, etc.) as in Grid5000 and unlike in PlanetLab. These capabilities make it a valuable complementary tool for P2P application developer to test their systems. ModelNet is however only capable of shortest-path IP routing, which represents its major drawback. This limitation makes it is not suitable for research in Traffic Engineering (TE), nor completely realistic as emulation environment, since no source-routing or load-balancing techniques, though widely used as of today [8], are available for testing.

In this chapter, we present ModelNet-TE, an extension of ModelNet that enables TE emulation and experiments. Colleagues in our group ported the original ModelNet core code from BSD to Linux, making it available to the scientific community [70] and we slightly modified it for stability reasons and for the integration with the traffic engineering tool that will be discussed next. The ModelNet-TE tool is interoperable, scalable and flexible. Interoperability and scalability are directly inherited from the original ModelNet code, that allows to run possibly thousands of unmodified application instances (provided that certain constraints are met, which we detail in the

following). Flexibility is instead a key of ModelNet-TE, as we took special emphasis in the design of a reusable toolbox, where researchers can easily integrate their own TE algorithms beside those that we already provide [31, 72][1].

We use ModelNet-TE to evaluate the uncoordinated interaction between Traffic Engineering (TE) at the network layer (L3) and end-to-end control policies applied by P2P systems at the application layer (L7). Indeed, though a number of work have studied the issue of selfish routing [45, 47, 58, 75, 90, 99] most of these work adopt a theoretical approach, which is especially true for the case of the uncoordinated interaction of routing dynamics at different levels [45, 47, 58]. On the other hand, while several experimental studies of popular P2P applications exists [16, 29, 87, 91, 101, 106, 123] they nevertheless neglect the interaction with the underlying network. While their approach is necessary to understand application dynamics, it does not allow ISPs to understand the impact of TE on the traffic of their users; nor it allows P2P developer to assess how do their algorithms perform over a reactive network.

Aiming at filling this gap, we study the L3/L7 interaction via ModelNet-TE. To prove the flexibility of ModelNet-TE, and to gather a full blown set of results, we carry on an experimental campaign that, as sketched in Fig. 7.1, considers a rich set of (i) L3 topologies and routing algorithms and of (ii) L7 applications and peer population models. At L3, we consider both a simplistic pure overlay model, where the bottleneck is only at the access, as well as the popular Abilene topology spanning across the US, in which any link can become a bottleneck (depending on the traffic matrix induced by the P2P application). As reactive L3 Traffic Engineering we use a multi-path load balancing algorithm [31], that we compare to standard shortest path IP routing. At L7, we consider two reactive P2P applications, namely BitTorrent [15], the most popular file-sharing application nowadays, and WineStreamer [14, 55], an open source live streaming application[2]. Furthermore, we consider both a uniform peer population across the network, or a skewed population, that reflects the actual number of citizen in major US urban areas.

Summarizing, the main contribution of this chapter is to carry the first thorough campaign, exploiting an experimental methodology, that focuses on the interaction of P2P dynamics with the underlying L3 network. Experimental results yield the following interesting insights: (i) bottleneck in the network (which recently arose in case of popular applications and content [23,67]) may have a profound impact on the P2P application performance; (ii) the peer population model, other than shaping the traffic perceived by the L3 network, may significantly contribute in determining P2P performance; (iii) traffic engineering may ameliorate network-centric ISP performance (e.g., equalize traffic on links) to the detriment of user-centric P2P performance (e.g., due to unexpected interactions with TCP transfers or P2P trading logic).

The rest of the chapter is organized as follows. In section 7.1 we present related work, in 7.2 we present a system-level view of the original ModelNet emulator, describing the TE extension and the load balancing algorithm we use, providing an initial assessment of its scalability as well. 7.3 provides a detailed description of the emulated scenarios, describing the P2P applications used, the different network and population models, and the evaluation metrics. Results of our experimental campaign are then reported in 7.4. Finally, conclusive remarks are drawn in 7.5.

---

[1]The traffic engineering algorithm we present in this chapter has originated from a collaboration in which we provided the network emulator and the interfaces to hook in while authors of [31,72] provided the load balancing algorithm.

[2]We justify the choice of these particular applications in section 7.3.2
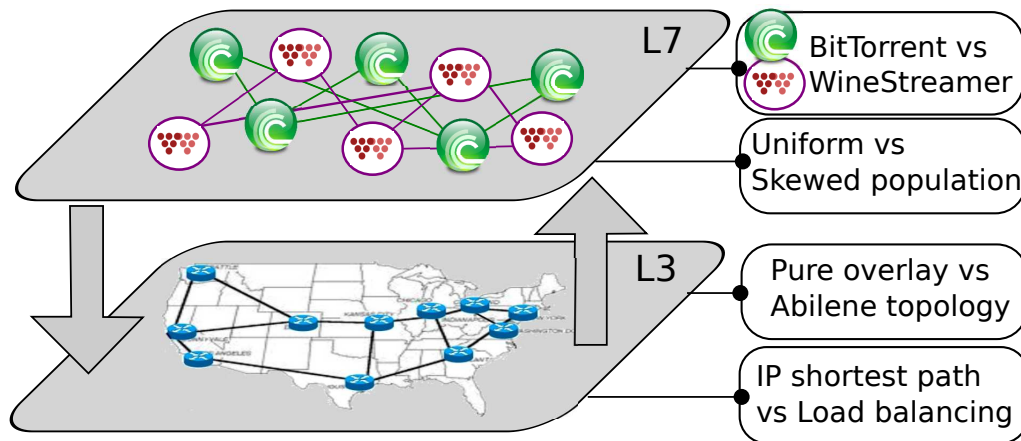
Figure 7.1: Synopsis of the elements taken into account in this chapter
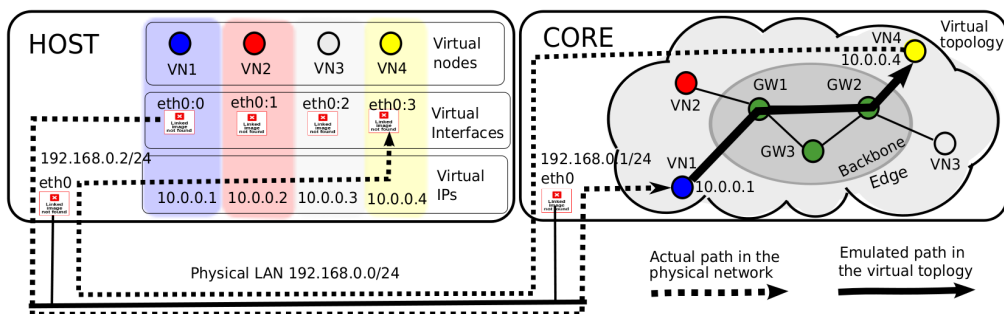
Figure 7.2: ModelNet low-level architecture

7.1 Related work

Two bodies of work are related to ours. On the one hand, there is work focusing on the experimental evaluation of P2P applications by means of testbeds or large scale-experiments. On the other hand, work exists that focuses on the interaction of both layers.

7.1.1 Experimental evaluation

As far as experimental evaluation is concerned, we find two main class of methodologies: the first employs controlled testbed [13, 16, 91], the second world-wide infrastructures [63, 84, 123].

Controlled experiments are run in dedicated infrastructures, such as Grid5000 [16, 91] or ad hoc testbeds [13], where clusters of several coordinated machine, which are usually connected through LAN, run P2P clients. As these infrastructures are general purpose (i.e., not tailored for network experiments) experimental setup can be a burden. Besides, latency and packet drops must be artificially enforced by external tools and it is impossible to carry out studies on L3/L7 interaction. Nonetheless, [91] concludes that BitTorrent experiments performed on cluster are realistic and that wide area network latency and packet losses impact for less than 15% of the download time. If we agree that this precision is realistic enough for elastic file-sharing applications, such error margin cannot be tolerated for interactive live-streaming applications – where chunk losses or delayed arrivals heavily impact the quality of experience.

World-wide infrastructures such as PlanetLab [88] and OneLab [74], have long been used to

test and benchmark distributed applications [63, 84, 123]. Currently, PlanetLab provides about 1000 nodes at 500 different sites scattered around the globe. Yet, one of the perception is that PlanetLab is not suitable for P2P experiments since it is composed mainly by high capacity nodes and few DSLs [108]. Another potential problem arises from the fact that access to nodes is shared among users, each of which is getting different "slices": yet, as multiple concurrent experiments can be run on the same infrastructure, there is no control on CPU load[3] nor other traffic (e.g., two P2P experiments running concurrently, or measurement between PlanetLab nodes) that can both alter experiment results[4].

7.1.2   Routing layer interaction

The study of the interaction of several routing layers is instead motivated by findings in [75,90,99]. Briefly, while selfish routing may be highly unoptimal in general settings [75, 99], in practice it performs reasonably well in Internet-like environment [90], which justify and confirms its interest. At the same time, an important observation is that local optimization entailed by selfish overlay routing may counter actions taken by the underlaying network, overall resulting in poor system stability.

As a consequence, there has been a recently increased attention [45, 47, 58, 127] on the potential issues on uncoordinated, uncontrolled interaction of two routing paradigms. Different studies consider different levels of interaction such as a P2P overlay network and the underlying IP network routing [47, 58], IP routing and the underlying MPLS/GMPLS network [127], multiple P2P overlays routing, coexisting on a given underlay network [45].

7.1.3   Advances with respect to the State of Art

This work extends and complements both bodies of work. On the one hand, though we use an experimental methodology, to the best of our knowledge P2P traffic has been studied with a pure overlay model [13, 16, 29, 53, 63, 87, 91, 101], neglecting thus the mutual impact with lower-layer network. On the other hand, most of the work focusing on interaction between different routing layers exploits a Game Theoretical approach [30, 58, 127], with a simulative approach limitedly used in [47].

As such, the research community still lacks more realistic and practical studies, which is precisely what we address in this work: thanks to the ModelNet-TE framework, we encompass both classes of work, by proposing the first study of routing layer interaction that exploits an experimental methodology.

Moreover, ModelNet-TE tool sits between controlled and wild testbed infrastructures, trading off between the realism of PlanetLab, and the scalability of Grid5000, and adding the control over the network topology and routing algorithm. Yet, the scale of the experiments that can be performed is not compromised: to prove this, Tab. 7.1 reports a comparison of different closely related work, highlighting the scalability aspects for the methodologies discussed so far.

Notice that most works scale up to a few hundreds peers, i.e., the same order of magnitude of our ModelNet-TE experiments, confirming the validity and usefulness of the tool. Only two notable exceptions push the experiment scale to 1,000 [13] and 10,000 [16], trading off experimental scale with simplicity of the experimental setup. Still, the size of ModelNet-TE experiments could

---

[3]For instance, [87] points out that "Since most Planetlab machines are usually over-loaded, we limit the overlay size to 160 peers running on machines that report at least 5% idle CPU time."

[4]Notice that this should change with the recent ability in OneLab to reserve resources, similarly to what happens in Grid5000

Table 7.1: Scale of different P2P experiments (this work in bold)

Ref.	Testbed	Nodes	Nodes/ Machine
[16]	Grid5000	10000	100
[13]	Own testbed	1000	5
[29]	PlanetLab	400	1
[87]	Modelnet	320	32
[91]	Grid5000	300	100
[101]	PlanetLab	280	1
	ModelNet-TE	**200**	**35**
[87]	PlanetLab	160	1
[86]	Modelnet	80	8
[53, 63]	PlanetLab	41	1

be scaled up, though a higher number of HOST machines would be needed in that case. Taking into account that the scale of experiments is limited by the core machine, we expect the testbed to scale up by a factor of 4 in the WineStreamer scenario leaving the access capacities unchanged and augmenting the number of HOST machines; more details about scalability issues will be given in 7.2.4. In the BitTorrent case, larger swarms up to a factor of 5 could be instead emulated by considering lower access speeds (such as the 800 Kbps [91] in Grid5000 or 1 Mbps [87] in ModelNet)(i.e., lower access speeds allow a greater number of nodes since the aggregate traffic rate passing through the core is lower).

7.2 ModelNet-TE Emulator

7.2.1 ModelNet Primer

The original ModelNet software [113] is an IP network emulator, which allows to run unmodified applications plugging them into realistic, large-scale networks. ModelNet implements emulated virtual topologies that are independent from the physical testbed interconnection. A synoptic of its architecture is sketched in Fig. 7.2. The ModelNet environment consists of two kind of machines, HOST and CORE, interconnected by a physical LAN (address 192.168.0.0/24 in the figure). The CORE machine emulates the virtual network with an arbitrary topology, while each HOST machine runs multiple instances of the application under test (in our case, BitTorrent or WineStreamer clients, see 7.3.2). Each instance is bounded to a Virtual Node (VN) and a virtual IP address belonging to a private subnet (typically the 10.0.0.0/8 network), dedicated to ModelNet emulation. While in the physical topology VNs runs on HOST machines, in the virtual topology each VN is attached to a Gateway node (GW), that constitutes its ingress/egress point in the emulated network.

Notice that, for the emulation to be successful, each packet generated by any VN application instance must be delivered to the CORE over the physical LAN: this is because, for each packet, IP network emulation takes place at kernel level in the CORE. Emulation tasks can be summarized as follow: using the source and destination virtual IP addresses of packets coming from applications running on HOST machines, the CORE determines a path through the virtual topology and handles the packets accordingly. Each hop on this path has a given bandwidth, queuing, propagation delay, and packet loss characteristics: thus, this hop-by-hop emulation lets IP traffic experience realistic wide area effects, possibly including congestion on core links. Notice that packet emulation occurs in real time, and packet delays are handled with millisecond accuracy.

For the sake of clarity, Fig. 7.2 depicts the case of an application instance bound to a virtual

node VN having IP address 10.0.0.1, that wants to send a packet to a VN having IP address 10.0.0.4. Though both VNs are on the same physical HOST, packets are however delivered to the CORE over the physical LAN, through a kernel level hack happening at the interface of the machine hosting the source VN[5]. The CORE routes then the packet in the emulated topology: once the packet has crossed all path hops (in the emulated topology), it is delivered (again through the physical LAN) to the HOST to which the destination VN is bound.

7.2.2 ModelNet-TE Overview

To overcome the single-path limit, we have modified the original ModelNet kernel module to allow multiple parallel path to be used between any source destination pair: we call the improved emulator ModelNet-TE. We have ported the original BSD code to the Linux kernel, that we make available[6] at [70].

We now briefly describe the improved internal ModelNet-TE structure. As can be seen in Fig. 7.2, the topologies emulated by ModelNet-TE can be logically divided in two sections: a *backbone* part that connects the gateways nodes GW and an *edge* part that comprises the set of access links interconnecting each VN to a single GW node. In practice, we can imagine that each GW to which VNs are attached, acts as WiFi hotspot or an ADSL DSLAM. We point out that the TE algorithms only apply to the *backbone* part of the network, acting thus on aggregated traffic demands coming from the network edge.

The high-level idea of the ModelNet-TE extension is depicted in Fig. 7.3, where all the relevant components are represented, as well as their relationship and their mean of interaction. Basically, the *emulated topology* is described through an XML file, as in the original ModelNet-TE. Topology definition consists in specifying both edge and backbone link, by fully defining the property of each link (such as bandwidth, delay, loss probability, queue size, etc.) and their topological interconnection structure.

Routing tables of nodes in the emulated topology are instead specified in a *source routes* file, representing the Forwarding Information Base (FIB). In ModelNet, FIB is a text file containing, for each VN couple, the list of hops that each packet coming from source VN_S and destined to VN_D has to cross. In ModelNet-TE, the FIB is extended in order to handle multiple routes between each VN pair: more specifically, a *probability* is associated to each of the multiple paths connecting each VN couple. The kernel-level forwarding module applies then this probability for each packet: i.e., on each new packet arrival, one of the multiple paths is chosen at random according to the specified probability.

Notice that the forwarding module only applies per-path probabilities, but expects an external *L3 TE routing module* to set them: this way, routing optimization is *decoupled* from low-level forwarding, making it easy to integrate new algorithms. To prove the flexibility of the mechanism, ModelNet-TE already implements two different TE algorithms [31,72] (although in this work, we use only one of them, that we briefly describe in 7.2.3).

Centralized TE algorithms can easily run on ModelNet-TE. Notice that, given that all GW traffic transits through the CORE machine, the TE optimization algorithms running on the CORE benefit of the knowledge of the Traffic Matrix (TM), and of the load on each link. TM is continuously updated by the CORE: more precisely, at a configurable periodic interval, the CORE exports

---

[5]ModelNet-TE flips a bit of the virtual destination address, which forces packets to exit the HOST (instead of being "captured" by the loop-back interface), and be directed to the CORE (which is set as HOST default gateway). The same bit of the IP destination address is then flipped again at packet reception in the CORE.

[6]As our patch applies only to specific versions of the Linux kernel (namely `2.6.18` or `2.6.22`), and so as to reduce the startup time for new users, we directly provide full ready-to-use system *images* of the patched CORE and HOST machines, containing the source code as well.
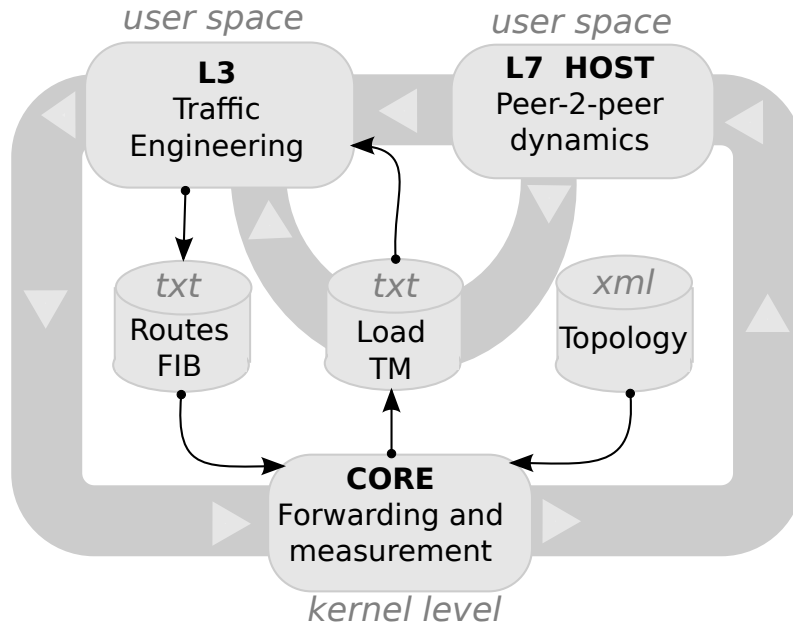
Figure 7.3: ModelNet-TE high-level architecture

TM information from the kernel, writing it in an output text file containing the *load* of each link on the network (expressed as the sum of the Protocol Data Unit (PDU) lengths that crossed each link during that timeframe). TM information can then be used as input by the TE toolbox running in the user space, so to compute the FIB to be used by the CORE in the kernel-level forwarding process, as illustrated in Fig. 7.3.

The routing/forwarding decoupling is not only a natural choice, as it follows from standard operation in IP networks, but also simplifies the integration of new modules by (i) providing a simple, clean and natural interface for the Linux environment (i.e., a file to read TM statistics from the kernel, a file to write FIB information for the kernel) and (ii) avoiding constraints on the time-scale of TE optimization module (which asynchronously runs in user space). To better grasp the advantages of this design choice, let us consider which one between the (i) TE optimization and (ii) FIB update process may constitute a bottleneck. Consider the ideal case of an instantaneous optimization algorithm: then, update rate could only be limited by the time it takes ModelNet-TE to read the new FIB from disk. As, in our experience, loading the update routing tables takes less than a second (for moderate size networks of 10-50 nodes), this poses no constraint on the choice of TE timescale. Indeed, the bottleneck in the FIB reconfiguration rate is more likely tied to the TE algorithm running time, that depends on the algorithm complexity, and is generally tied to the solution of an optimization problem.

With respect to our L7 vs L3 routing interaction study in a P2P vs TE scenario, notice that once the source routes are updated by the TE module, the CORE will use the updated routes in the emulated topology, possibly triggering in turn changes at L7 due to P2P traffic dynamics, as depicted in Fig. 7.3. This feedback happens naturally, i.e., without requiring any modification at the application level, which is thus unaware of the L3 dynamics. In turn, changes in the L7 traffic matrix translate into different loads at L3, which possibly triggers a new update of the source routes by TE, closing the feedback loop.

7.2.3 ModelNet-TE Minimum Congestion Load Balancing

As already said in the introductory section, we did not participate in the development of the load balancing algorithm but we present its description here to ease the understanding of the following sections.

The L3 TE algorithm we consider in this chapter is the classic minimum congestion load balancing problem, probably first introduced in [27]. For each link l, we define a convex increasing function $f_l(\rho_l)$, where ρ_l is the load on link l, and the problem objective is to minimize the sum over all links of $\sum_l f_l(\rho_l)$. The rationale is that this function represents the congestion on the link, and that TE should strive to minimize the total congestion on the network. Convexity is intuitively justified by the fact that at higher loads, an increase in load generates more congestion than at lower loads. This objective function has become very popular, to the point that [125] defines TE as the procedure through which the network operator minimizes $\sum_l f_l(\rho_l)$.

Regarding the link congestion function $f_l(\rho_l)$ we chose the resulting mean queue size of a M/M/1 queue:

$$f_l(\rho_l) = \frac{\rho_l}{c_l - \rho_l} \tag{7.1}$$

The influence of the particular choice of $f_l(\rho_l)$ on different performance indicators is studied in [31]: as long as (7.1) is convex, increasing and diverges as ρ_l reaches c_l, the exact choice is unimportant in what regards path available bandwidth and link utilization.

The first input to the algorithm is the TM information. If every GW is ordered by an index, TE traffic matrix contains in its ij-th entry the mean traffic demand from node i destined to node j, usually called Origin-Destination (OD) pair. In addition to the TM, the algorithm also requires a set of paths that each OD pair may use. By specifying this set *a priori*, the resulting optimization problem is convex, which simplifies its solution (note that paths in this set are only *potentially* used in the solution, i.e., the amount of traffic sent along some paths may be zero). In particular, we bound the length of alternate paths $|A|$ with respect to the length of the shortest path $|S|$ found by Dijkstra, so that $|A| < |S| + 3$. In other words, we take alternate paths that exceed the shortest path by at most two hops, so to be able to route around a congested link or node, without incurring the load overhead of longer paths.

All in all, given the topology, the $f_l(\rho_l)$ associated to each link on the network, the traffic matrix and the paths that each OD pair may use, an algorithm is needed to find the amount of traffic that each GW should send along each path, so as to minimize $\sum_l f_l(\rho_l)$. With this respect, several choices are possible since the problem is convex. For instance, a classical approach to this kind of problems is the gradient descent method [26]. However, most of gradient based algorithms include a parameter that controls convergence speed, which may be very tricky to assign. Although for each algorithm there exists a range for this parameter that makes the solution converge, in turn these values may result in slow convergence in certain situations. Conversely, larger values for the descent parameters may translate into faster convergence, but can possibly also trigger oscillations.

To avoid this reactivity-stability tradeoff, we resort to the use of so-called no-regret algorithms: in particular, ModelNet-TE implements the Incrementally Adaptive Weighted Majority (iAWM) algorithm [7], that presents the advantage of self-regulation. For instance, its convergence speed is automatically set, depending on previously observed values of $f_l(\rho_l)$. For lack of space, we invite the reader to [7] for a thorough algorithm description, and to [31] for extensive simulations results.

7.2.4 ModelNet-TE Scalability

It should not be forgotten that virtual nodes and virtual topology are ultimately emulated by the over a physical network, and that multiple virtual nodes are run on HOST machines. Hence, certain

constraints must be met so to avoid that LAN capacity, CPU or RAM bottlenecks perturb the experiments. Our setups comprises 6 HOSTs and 1 CORE machines, each equipped with Intel Xeon CPUs (4 cores in hyper-threading running at 1.86GHz) and 4GB RAM, that are interconnected by an Ethernet Foundry EdgeIron 24G-A switch with 1 Gbps ports ports and a 4 Gbps back-plane.

Concerning CPU and RAM bottleneck, we experimentally verified that each HOST is able to run up to 35 P2P clients without incurring CPU penalties (i.e. CPU idle time was always higher than 20%): hence for instance, with 6 machines, we can build overlays whose size reaches $N_P = 200$ P2P clients (which is also a reasonable swarm size for both file-sharing and live-streaming, cfr. 7.3.2). We point out that the number of P2P application instances that can be run on a single HOST machine also depends on the emulated VN uplink $C_{U,i}$ and downlink $C_{D,i}$ access capacities, as these translate into constraint on the physical HOST capacity. In our scenarios, we verify that such safety constraints are met, as the maximum aggregated throughput (measured over 1 second long time slots) generated by the whole set of 6 HOSTs never exceeds 377Mbps (BitTorrent) or 140Mbps (WineStreamer).

An even more stringent constraint applies however to CORE capacity: indeed, in ModelNet-TE each packet needs to traverse the core twice, and although packets are sent on virtual interfaces, they enter the CORE through the same physical interface. As emulation happens in real time, at any time the overall traffic sent by all HOSTs in the physical network (or, equivalently, by all VNs in the emulated network) must not exceed the capacity of the CORE as otherwise unwanted queuing and drop effects may arise in the physical LAN, perturbing thus the experiments. In other words, it must be ensured that the sum of uplink and downlink traffic does not exceed the CORE capacity, translating into $C_U + C_D < 1$ Gbps, where C_U and C_D represent the aggregated uplink $C_U = \sum_{i=1}^{N_{VN}} C_{U,i}$ and downlink $C_D = \sum_{i=1}^{N_{VN}} C_{D,i}$ capacities respectively. Our setup can therefore be considered conservative since, as we just saw, both applications generate an aggregate traffic which is lower than the 1 Gbps threshold. Hence, we can infer that, at least theoretically, our testbed could scale-up by a factor of 2.5 and 7 respectively for BitTorrent and WineStreamer. A more conservative estimate that takes into account [86] experience (hinting to a degradation of ModelNet core precision for aggregated traffic greater than 600Mbps), would still support a 4 times WineStreamer swarm.

7.3 Scenario and methodology

We now describe the scenarios emulated in our experimental campaign, providing motivation and detailed information concerning our choice of (i) network topologies, (ii) TE algorithm details, (iii) population models, (iv) P2P applications.

7.3.1 L3 Network

7.3.1.1 Topology

Irrespectively of the P2P application, we consider two network topologies: namely, (i) a realistic Abilene topology and (ii) a simplified pure overlay model.

Often indeed, network topology is not considered due to studies such as [4], showing the bottleneck to be sited *at the edge* of the network. However, this assumption holds for scenarios with a majority of low-capacity access technologies, such as e.g., ADSL lines, whose upload capacity is significantly limited. Furthermore, this may no longer hold in case of fast FTTx Internet access (i.e., Fiber To The Curb/Home), which is in the agenda of all major developed countries, and that reinforces the need of studying more realistic network scenarios.
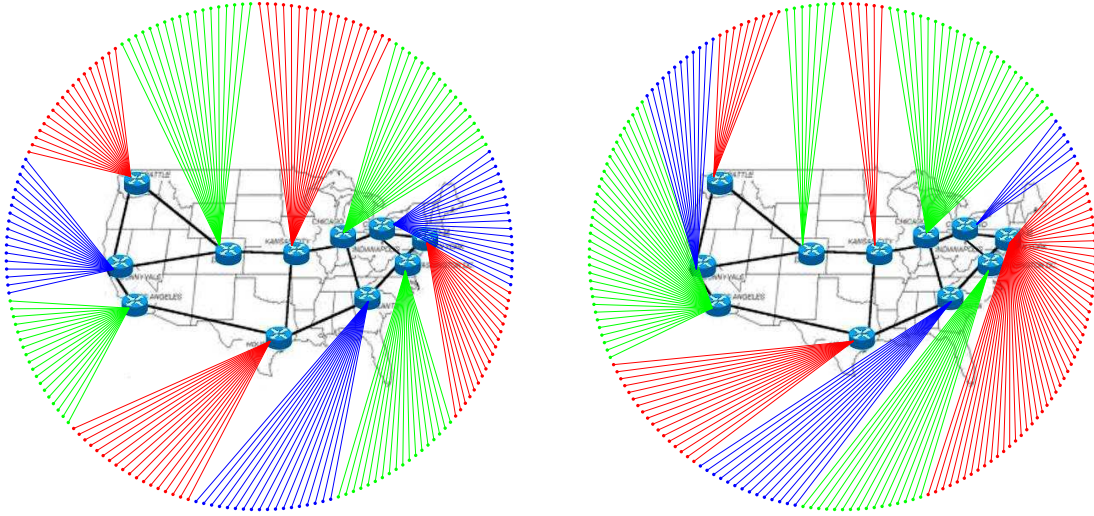
Figure 7.4: L3 Abilene topology and L7 swarm: uniform (left) vs skewed (right) population models.

Second, signals of the fact that P2P (or other user-level applications) are already causing congestion to ISPs can be inferred by recent issues such as (i) the throttling of BitTorrent connections by Comcast in the US [23] or (ii) the throttling of Megaupload by France Telecom in Europe [67]. The above examples show that, actually, ISPs are *already struggling* with the amount of data in their networks as of today, i.e., even when FTTx represent a minority of access technologies.

We take into account the above observation while building an emulation scenario. Due to the scale of our testbed, and to the physical limits of the interconnection (i.e., Ethernet transceivers, switches back-plane, number of HOST described in 7.2.4), it is however clearly impossible to emulate a full speed Internet core. Rather, we observe that problems may arise when the aggregated traffic generated by the user may cause congestion in the network, and decide thus to *jointly* scale access and core capacities so to produce situations similar to [23, 67]. Notice also that while, the BitTorrent vs Comcast case has already hit the media, P2P-TV application may represent a similar threat due to the forecast ed growth of Internet video [21].

The realistic network scenario we design is thus as in Fig. 7.4, with core links interconnected according to the well-known Abilene topology [1], comprising $N_R = 11$ routers spanning over the US country. In our scaled setup, we consider core links capacities in $C = \{5, 10\}$ Mbps and we model peer access capacity as loose symmetric FTTH with $C_{D,i} = C_{U,i} = 5$Mbps. It is worth pointing out that the aggregate traffic of each peer is on average between 720Kbps (WineStreamer) and 1.5Mbps (BitTorrent), and that since part of that traffic is directed toward other peers behind the same GW, it will thus not consume core link capacity C. Hence, while $C = 5$ Mbps represents an under-provisioning scenario (i.e., the core is not able to transport all the aggregate traffic and we expect losses on core links), the $C = 10$ Mbps scenario models a fairly well provisioned network. These values have been chosen empirically since is difficult to exactly size a network in function of access capacities and video rate. Moreover notice that, though realistic, the Abilene topology is also a hard scenario for load balancing, since the level of path diversity may not always allow to route *around* congestion.

To better grasp the impact of the network topology, we compare the Abilene scenario with a simplified model (not shown in the picture) where all peers are interconnected in a star topology to a single network core router. No capacities or delay are emulated in the network core (but only

at the access): hence, due to our physical setup, the backbone runs at 1 Gbps switched Ethernet speed (which is much faster than the Abilene case, and where congestion never arises). Still, in this scenario we may enforce realistic access latencies, depending on the population model (see 7.3.2.1).

7.3.1.2 Traffic Engineering

We now discuss some implementation details of the iAWM algorithm described in 7.2.3, notably the timescale at which the algorithm is run. Let us recall that one of the inputs to the algorithm is the traffic matrix (TM), defined as the amount of aggregated VN traffic each GW node exchanges with each other.

In ModelNet-TE, the TM is sampled over windows of w seconds ($w = 1$ in our case), and ModelNet-TE can perform simple operations[7] (e.g., average, standard deviation, maximum, etc.) over W consecutive time windows. Then, after W consecutive windows, these demands are exported from the kernel to the TE algorithm (see 7.2.2). For the experimental campaign reported in this chapter, we set $W = 30$, and run iAWM periodically after W windows, to set the new routing tables. Notice that the resulting timescale of the L3 traffic engineering decisions is on the order of 30 seconds (which is comparable with the order of the L7 timescale, as we describe in the next section).

7.3.2 L7 P2P Applications

At L7, we build realistic scenarios by considering heterogeneity in the (i) class of P2P applications and (ii) peer population models.

We select two P2P applications, namely BitTorrent [15] and WineStreamer [14, 55], that offer heterogeneous services and have thus a rather dissimilar design. Indeed, BitTorrent and WineStreamer are rather diverse in their constraints (i.e., elastic file-sharing vs minimum rate live-streaming), architectural choices (i.e., TCP vs UDP) and trading logic (i.e., rarest-first vs playout-deadline based). Yet, these applications also share some similarities (i.e., both are built on an unstructured a mesh overlay, with each peer optimizing its neighborhood by preferring high-bandwidth peers) that are a natural result of the evolution of the Internet P2P ecosystem, following the good performance these choices have exhibited [54, 104].

For both applications, we emulate a flash-crowd scenario in which a single source initially provides content (i.e., a file, or a TV channel) to a swarm of $N_P = 200$ peers. Notice that this is a reasonable swarm size for file-sharing applications, that furthermore trades off between observation in [84, 126]: more precisely, [126] observes that only about 1% of the torrents have more than 100 peers, while [84] reports typical sizes of BitTorrent swarms to be around 300-800 peers [84] [8]. As far as live-streaming is concerned, in chapter 3 we observed that the swarm size for the same channel also depends on the application (i.e., which reflects the application popularity rather than the popularity of the content itself), with swarms ranging from 500 peers in TVAnts to about 180,000 peers for PPLive for the most popular content. Hence, $N_P = 200$ can represent an highly popular channel over a mildly popular application, or a mildly popular content over an highly popular application.

For the sake of simplicity, we consider homogeneous swarms capacities (i.e. each node has the same access capacity as described in section 7.3.1.1): notice that the effect of heterogeneous

---

[7]The support for different operations simplifies the implementation of different algorithms, that may rely on different inputs (e.g., average for iAWM [7] or maximum for [72]).

[8]This may be due to the fact that [126] exhaustively explores *all* torrents, while observation in [84] are limited to a smaller torrents catalog.
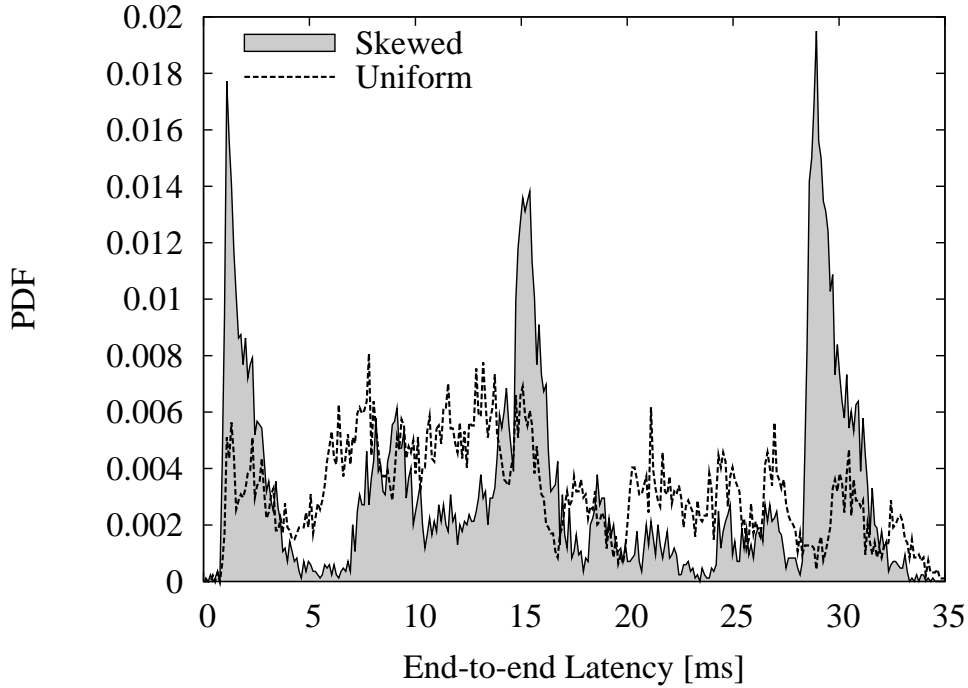
Figure 7.5: Uniform vs skewed population models: end-to-end latency distribution.

swarms with multiple capacities are well-known [53] from a pure L7 standpoint, and may be worth investigating from a joint L7/L3 viewpoint as future work.

7.3.2.1 Population model

Irrespectively of the P2P application, we may consider different swarm population models. In the Abilene topology of Fig. 7.4, each router acts as access router for several peers of the network: since the Abilene network comprises $N_R = 11$ nodes, and since we emulate $N_P = 200$ peers swarms, on average there are about 20 peers per node. In both cases, swarms initially have a single source located in Kansas City (in the middle of US).

However, while emulation studies usually uniformly distribute peers in the network (e.g., by spreading peers at random over PlanetLab nodes), we argue that peer population is more likely to reflect the actual human population in the real world. As the Abilene network spans across the US, we consider US cities of Abilene PoP and distribute peers to routers proportionally to the population of the corresponding urban area [118].

The skew in the population distribution translates into a more clustered swarm population, where several peers (users) may be found behind the same router (city). In turn, this also affect the distribution of the end-to-end[9] latencies, as peers are now more likely to be close.

The difference in the uniform vs skewed population models is pictorially represented in Fig. 7.4, and the corresponding distributions of edge-to-edge latencies are reported in Fig. 7.5. Latency distribution shows the impact of skewed peer population, as three peaks clearly arise:

---

[9]Notice that edge-to-edge latencies are measured between any pair of gateways GW (or IP routers), taking into account the physical distance between US cities. We derive latencies by applying a factor of 5 microseconds per kilometer of optics fibre [52]. End-to-end latencies are emulated by additionally taking into account the local loop network beside the access GW adding a uniformly distributed access latency with mean 1ms.

these correspond to (i) low delay for nearby communication (i.e., behind the same GW or confined in the east cost, west cost, or mid US), (ii) moderate delay for mid-range communication (i.e., between east cost and mid US, or west cost and mid US), and (iii) high delay for faraway communications (i.e., between both coasts). Notice also that high delay PDF peak is pronounced, as the majority of US inhabitants can be found along the eastern and western US coasts. Conversely, uniform peer distribution yields to a completely different latency distribution, which is unrealistic with respect to actual measurements carried on in measurement projects such as [120].

7.3.2.2   P2P Filesharing: BitTorrent

For file-sharing, we use the Python version of BitTorrent-4.0.0-GPL. In file-sharing, the main aim is to let all peers in the swarm download the content in the shortest possible time. Notice that the BitTorrent version we consider employs TCP at transport layer (L4). While we are aware that recently BitTorrent introduced a new application-layer transport protocol based on UDP at L4 [96], we choose TCP filesharing since the new protocol is for the time being implemented only in a specific client (namely, μTorrent, that is estimated to account for varying ratio of BitTorrent clients from 15% [84] to 60% [126]. Besides, our attention is here more focused on the interaction of P2P applications and L3 network, rather than to the performance of BitTorrent under a new congestion control paradigm, which has been investigated in [110].

We point out that providing a survey of BitTorrent is out of the purpose of this work, for which we refer the reader to [15, 53]. Here, we only mention that BitTorrent peers establish and maintain a limited number of connections, over which they download small portions (or chunks) of the file they are interested in obtaining. Periodically (every 20 seconds), peers rank their connections depending on the download rate, keeping only the best connections ("chocking" the least performing ones), and optimistically trying to discover new potential good peers (nicknamed as "optimistically un-choking" in BitTorrent, and performed every 30 seconds). To avoid free-riding, BitTorrent enforces reciprocation of content exchange (tit-for-tat) and, to avoid resources hot-spot, BitTorrent peers try to equalize the chunk availability in the system by downloading the rarest chunk first. The timescale of the L7 application dynamics is on the order of 20 seconds, thus comparable with L3 dynamics.

In a flash crowd scenario, BitTorrent peers behave differently depending on whether they are leecher or seed. Initially, the seed is the unique source of a 100 MBytes file: hence, at the very beginning we expect most of the traffic to be originated from a single VN (i.e., the seed). However, as chunks start spreading in the swarm, exchanges between leechers become prominent, until the seed contribution is no longer necessary [53]. Hence, the traffic matrix offered at L3 by L7 will change during the whole experiment duration, so that the system evolves without ever reaching a stationary state.

7.3.2.3   P2P-TV: WineStreamer

For live-streaming, we use WineStreamer, an application developed in the context of the FP7 Strep Project on Network Aware P2P Applications over Wise Networks (Napa-Wine) [55]. In live-streaming the main aim is to let all peers in the swarm receive the minimum stream rate (similarly to video-on-demand), and to minimize the playout lag with the source (additionally to video-on-demand).

WineStreamer belongs to the last generation of live-streaming applications, and is able to take informed decisions with respect to the network state [28]. Knowledge of the network state is commonly nicknamed as "network awareness", and can either (i) be measured by the application or

(ii) be achieved with ISP cooperation. Examples of (i) include preferring nearby peers to faraway ones based on RTT or IP hop-count measurements, or preferring high capacity peers by means of bandwidth measurements, etc. [28]). An examples of (ii) is represented by IETF ALTO [6], defining ISP servers that acts as "oracles" and participate in the P2P peer selection process with informed suggestion on good candidate peers.

In this work, we only consider L7 measurements performed by the application itself, and turn off WineStreamer ALTO capabilities. Notice that, due to chunk transmission duration over ADSL lines, we expect the bandwidth-aware [104] peer selection criterion to prevail over latency-aware [12] or power-aware [93] (i.e., the ratio of bandwidth over latency) peer selection criteria. In other words, as for slow ADSL peers the chunk transmission time exceeds the propagation delay, in order to keep the overall system latency low, the ability to find high-capacity peers prevails over the ability to find nearby peers as already discussed in chapter II.

In all of the following experiments we stream a 600 Kbps video at 25 fps encoded with H264. In the video diffusion, we map every video frame to a single chunk (while several audio frames are grouped together in a single chunk to reduce the overhead). Video stream is not decoded at destination, but is discarded to avoid too many concurrent blocking IO calls; however, we log chunk-level arrival patterns to later evaluate the quality of user experience.

Again, providing a survey of WineStreamer is out of the purpose of this work, for which we refer the reader to [14, 55]. Rather, here we highlight the complementarity of WineStreamer with respect to BitTorrent. On this regards, we point out that the application exploits UDP and, though it implements a simple retransmission mechanism, the version we use in the testbed does not implement any form of congestion or flow control – hence, it sends out chunks at full speed. Moreover, chunk size is smaller than the one normally used in BitTorrent: as the scheduler performs decisions at a higher rate, hence we expect the P2P neighborhood to be more dynamic. Due to the use of UDP and to the minimum stream-rate requirement, WineStreamer is therefore a non-elastic application, with stringent near real-time requirements, unlike BitTorrent. Also, differently from BitTorrent, WineStreamer source is always providing new content to the swarm, at the same average rate, so that the system tends to a stationary state (although with a varying neighborhood).

7.4   Experimental Results

In this section, we report results of our experimental campaign, adopting two complementary viewpoints. First, we analyze the traffic that the P2P applications induce on the L3 network. Then, we analyze the impact that each simplistic vs realistic parameter choice has on the quality that the user perceives.

7.4.1   Traffic demands and link load

Let us investigate the traffic demands that P2P traffic induces over the whole network, and how these demands translate into individual link load. A pictorial representation of the traffic demands, at L7 and L3, is shown in Fig. 7.6 and Fig. 7.7 respectively.

Fig. 7.6 exploits a matrix representation to compare traffic demands of the applications, measured over the whole experiment duration, where black points indicate a chunk exchange between two peers. Already at a first glance we can observe the difference in matrix density: WineStreamer behavior is much more "loquacious", while BitTorrent contacts a lower number of nodes.

Augmenting the same kind of representation with gray levels proportional to the volume of exchanged data, we analyze load on L3 induced by the L7 application. Considering for the sake of example only the BitTorrent application, we vary the L7 population model and the L3 routing,
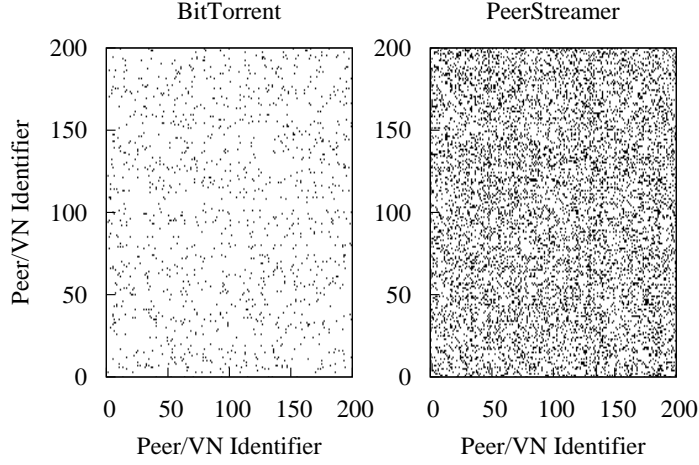
Figure 7.6: L7 traffic demands (Abilene topology, skewed population, IP routing): Swarm adjacency matrix for BitTorrent (left) and WineStreamer (right)

and depict the L3 TM in Fig. 7.7. Comparing top and bottom rows, one can gather the difference between uniform (top) and skewed (bottom) population models: data exchange in the skewed population is much more concentrated around few points (i.e. GW of large US cities as New York or Los Angeles) while in the uniform case, chunks are more evenly exchanged. Comparing instead left and right columns, one can gather the difference between IP shortest-path (left) and TE multi-path (right) routing: as expected, traffic is more spread out under TE load balancing, which is especially visible in the case of skewed population.

Performance of L3 network are reported in 7.8, showing link load and packet loss rate of individual links in the backbone, for both P2P applications and comparing IP vs TE routing. For the sake of simplicity, we only consider a scenario with realistic Abilene topology, 5Mbps core links, and skewed population. The striking differences that the L7 traffic matrix exhibited in 7.6, also entails different impact on L3, which can easily be explained.

Consider first the BitTorrent case in 7.8(a): as the application version we use employs TCP at L4, peers attempt at fully utilizing their uplink bandwidth (provided that they have enough chunk requests). In turn, this yields to a significant utilization of core link, which are mostly above 70% average utilization. Notice also that important links, such as those serving the gateways where the source is located, are not facing severe congestion (i.e., higher load or losses), which is again due to TCP congestion control. Hence, TE only provides marginal changes in the traffic matrix, increasing by about 2% the fairness of the link utilization (measured with Jain fairness index $(\sum_{i=0}^{N} \rho_i)^2 / (N \sum_{i=0}^{N} \rho_i^2)$ with ρ_i load on the i-th link).

Conversely, in the WineStreamer case of 7.8(b), we notice that load is unevenly distributed, with some links being lightly loaded and other carrying significant traffic amount, and experiencing non marginal losses. This striking difference is due to (i) chunk scheduling dynamics and (ii) the transmission of chunks as spurts of back-to-back packets over UDP. As for chunk scheduling, peers need to receive content over small time windows, and as new content is constantly being produced at the source, the source is possibly overwhelmed by chunk requests. Moreover, as no congestion control is implemented by the application, the chunk transmission process can be very
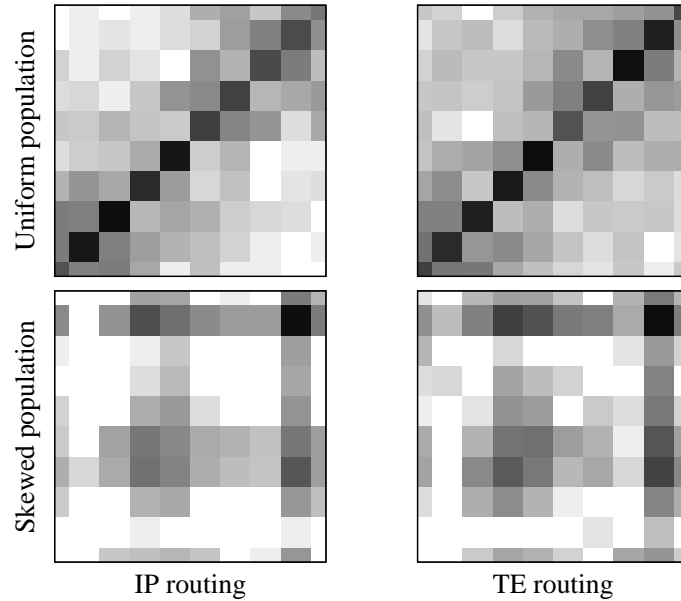
Figure 7.7: L3 traffic demands (Abilene topology, BitTorrent file-sharing application). Plots represent uniform (top) vs skewed (bottom) population models, with IP shortest path (left) and TE multi-path (right) routing.

bursty, so that aggregated traffic load is no longer smooth as in the TCP case, but is more likely to cause drops on some links. It must be said that 7.8(b) depicts a severe congestion scenario due to narrow 5Mbps link, that however let us better grasp some effects: notice indeed that while it can be seen that TE manages to equalize link level load to some extent (fairness increases by about 6%), TE efforts are not sufficient in this severe congestion scenario. Worse yet, use of multi-path TE can seldom overload links (that were only mildly loaded under IP routing), further inducing losses that were absent under IP (mean loss for IP and TE are respectively 3.08% and $4,77\%$). This behavior can be induced by the two uncoordinated control policies at L3 and L7, that happen independently and at the same timescale, and that we can exemplify as follows. Assume that L7 application decides to route content toward a peer whose path is lightly loaded and has never experienced losses. Assume further that, roughly at the same time, L3 realizes that links along the same path are lightly loaded, and decides to reconfigure the FIB. Now, what happens is that links along that path will experience a sudden, unexpected, load increase – that in case of live-streaming will be exacerbated by the use of full-rate UDP chunk spurts.

Notice also that 7.8(b) suggests that not all the network capacity is fully utilized, while a swarm of the same size in 7.8(a) was able to use more resources. This hints to the fact that peers in the WineStreamer application could potentially serve more other peers, thus offloading the source and further ameliorating system performance. Similar observations lately led to the development in WineStreamer of a dynamic aggregated congestion control over UDP, named Hose Rate Control (HRC) [13], showing thus that ModelNet-TE can provide an invaluable help to P2P application developers[10].

---

[10]HRC was however not available at the time of the experimental campaign.

Core links loads and drops

(a) BitTorrent

Core links loads and drops
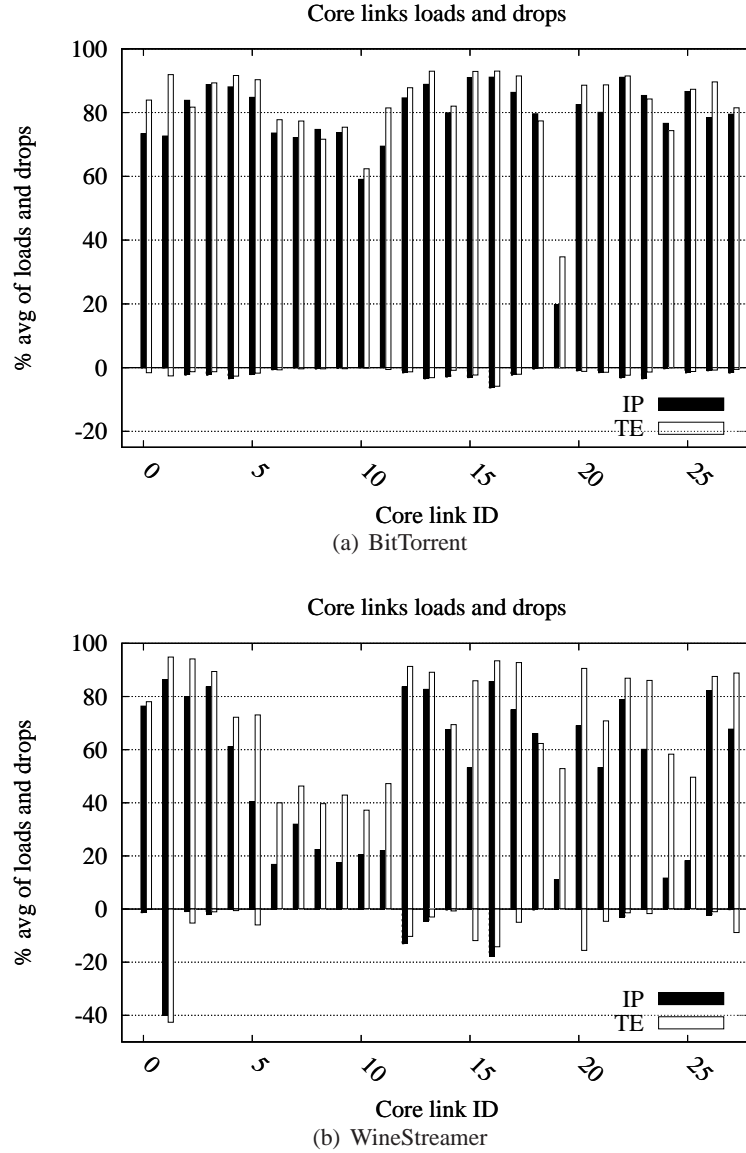
(b) WineStreamer

Figure 7.8: Network link load and packet loss rate, BitTorrent (top) vs WineStreamer (bottom), in the IP vs TE cases, Abilene topology and skewed population

7.4.2 Impact of population model and network capacity

We now focus on the performance of L7 applications, considering (i) the download rate of BitTorrent peers and (ii) the percentage of correctly received chunks for WineStreamer. We argue these to be the most relevant metrics that furthermore intuitively express the quality of user experience: as for (i), download rate is tied to the system efficiency and to the time it takes peers to complete their download; as for (ii), the video quality is badly affected by chunks that are received after the playout deadline (e.g., due to queuing delay at L3) or that are only partially received (e.g., due to packet loss at L3 that WineStreamer retransmission mechanism failed to recover). Both metrics are evaluated over windows of 10 seconds (i.e., the same timescale employed by BitTorrent to rank the active peer set for the choke operation). In the following, we report results gathered over

(a) BitTorrent

(b) WineStreamer
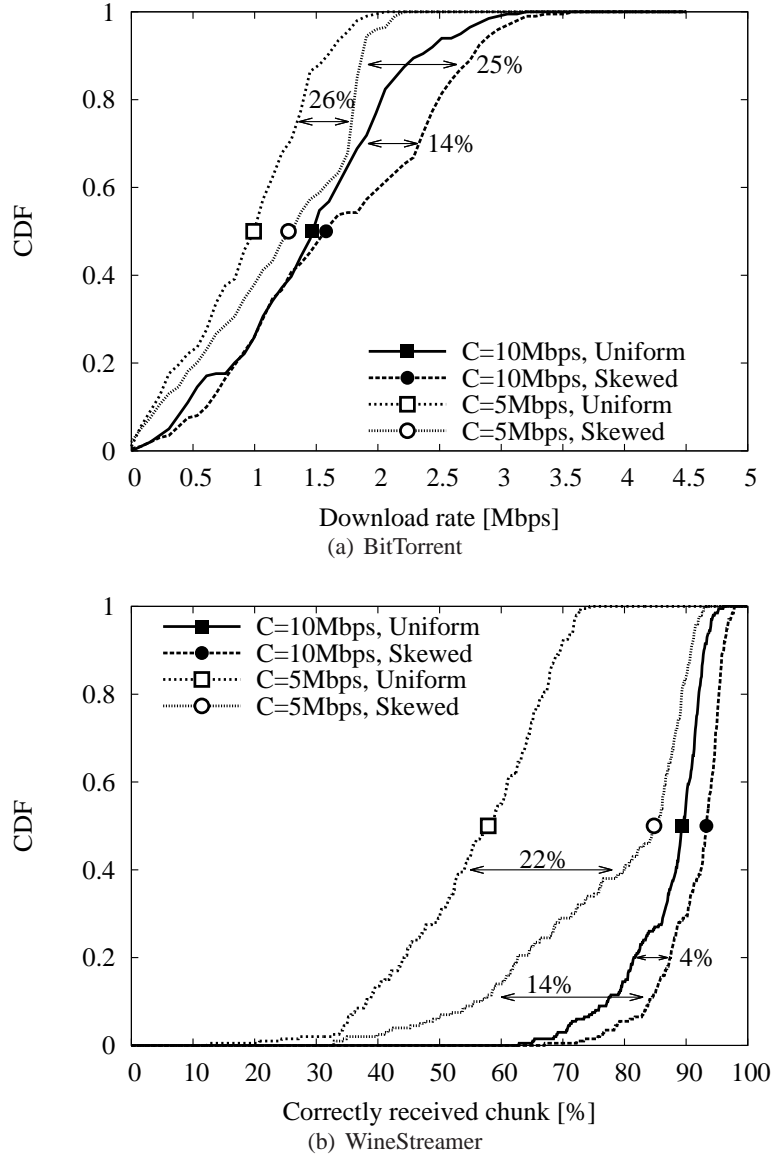
Figure 7.9: Impact of (i) uniform vs skewed population model and (ii) core link capacities C=5Mbps vs C=10Mbps. Arrows are used to highlight the percentage of difference between the *mean values* of the corresponding CDF curves.

5 different runs for any given experimental settings.

We first consider the impact that the capacity C on core links and the peer population model have, and depict the cumulative distribution function (CDF) of the download and chunk reception rates, measured over the whole swarm in Fig. 7.9 (for the time being, we fix the topology to Abilene and limitedly consider IP shortest path routing, whose impact we instead assess in 7.4.3).

Consider the population distribution first. The general consideration is that skewed population is beneficial in that, provided that the application is aware of latency or bandwidth[11], it can estab-

---

[11]Since TCP is advantaged by smaller RTT, application preferring high-bandwidth peers will also likely prefer nearby peers. Even for application such as PPLive, using UDP at L4 and measuring transfer rates at L7, we experimentally

lish neighboring relationships with peers attached to the same GW router, thus confining traffic at the edge and avoiding narrow core links.

Focusing on BitTorrent clients, we see that lowest download rates are achieved by peers on the C=5Mbps uniform population scenario: then, notice that a roughly equivalent performance gain can obtained by either (i) doubling the capacity under the same population model or (ii) considering a skewed population model at the same capacity level. Notice indeed that the average download rate increases by 25% and 26% respectively, as reported in Fig. 7.9-(a).

Considering WineStreamer clients, we see that the impact of the population model remains considerable, although in this case core links capacities plays a determinant role due to streaming constraints. Indeed, considering the under-provisioned C=5Mbps scenario in Fig. 7.9-(b), on average 22% more chunks are received under a skewed population model with respect a uniform one. Yet, change in the population model are not sufficient, as the percentage of received chunks for C=5Mbps is still low for some peers (those that were serviced by the underprovisioned link exhibiting up to 40% packet losses in Fig. 7.8), while situation improves considerably for C=10Mbps.

7.4.3 Impact of network topology and multi-path routing

Let us now consider the impact of the network topology and routing policies, where for the sake of simplicity, we only consider a skewed population model. To assess the impact of the topology, we compare a pure overlay model against a well provisioned Abilene network with core link capacity equal to C=10Mbps. While it is straightforward to foresee that on a pure overlay model both applications will perform better, as we removed any topological bottleneck, it would not be possible to quantify this gain without ModelNet-TE.

As expected, we see that in the case of BitTorrent the performance achieved on a pure overlay model can be significantly higher with respect to the Abilene case. This is because once in network capacity bottleneck are removed, TCP can make better use of the access capacity: on average BitTorrent can download at a 40% faster rate in a overlay-only scenario with respect to shortest-path IP routing. Conversely, as the video stream sent by WineStreamer has a fixed bitrate, and since the capacity of the network is provisioned to transport almost all that traffic, the difference between the overlay-only vs IP routing is limited to 4% (respectively, 95% vs 91% of chunks are received on average).

Finally we analyze the influence that TE techniques can have on P2P systems, by contrasting IP vs TE routing on the Abilene topology: counter-intuitively, we see that TE may lower the performance of both applications.

In BitTorrent, this can be explained by the fact that, recalling Fig. 7.8(a)-(a), nearly all links already operates at a regime close to their capacity. Hence, as TE reroutes the traffic along possibly longer paths, it extends the number of traversed link for each packet: thus, while TE balances the load more evenly across links, it may in turn raise the global network load. Second, since TE operates on a per packet basis, it may alter TCP congestion control: indeed, TCP transmission mechanism is self-clocked on the basis of RTT estimation. As each packet may traverse different paths, of different lengths, with different levels of congestion, this can significantly affects the RTT estimate. Second, as packets can now arrive out of order, this may possibly trigger spurious TCP retransmissions [68]

WineStreamer is a network aware application, that already executes measurement on the underlying L3 network, so to perform informed peer selection and scheduling decisions: in this case, periodical changes in the network topology due to TE are not beneficial to the already complex

---

verified in chapter 4 that bandwidth preference induces a clustering of nearby peers.

(a) BitTorrent L7 performance

(b) WineStreamer L7 performance
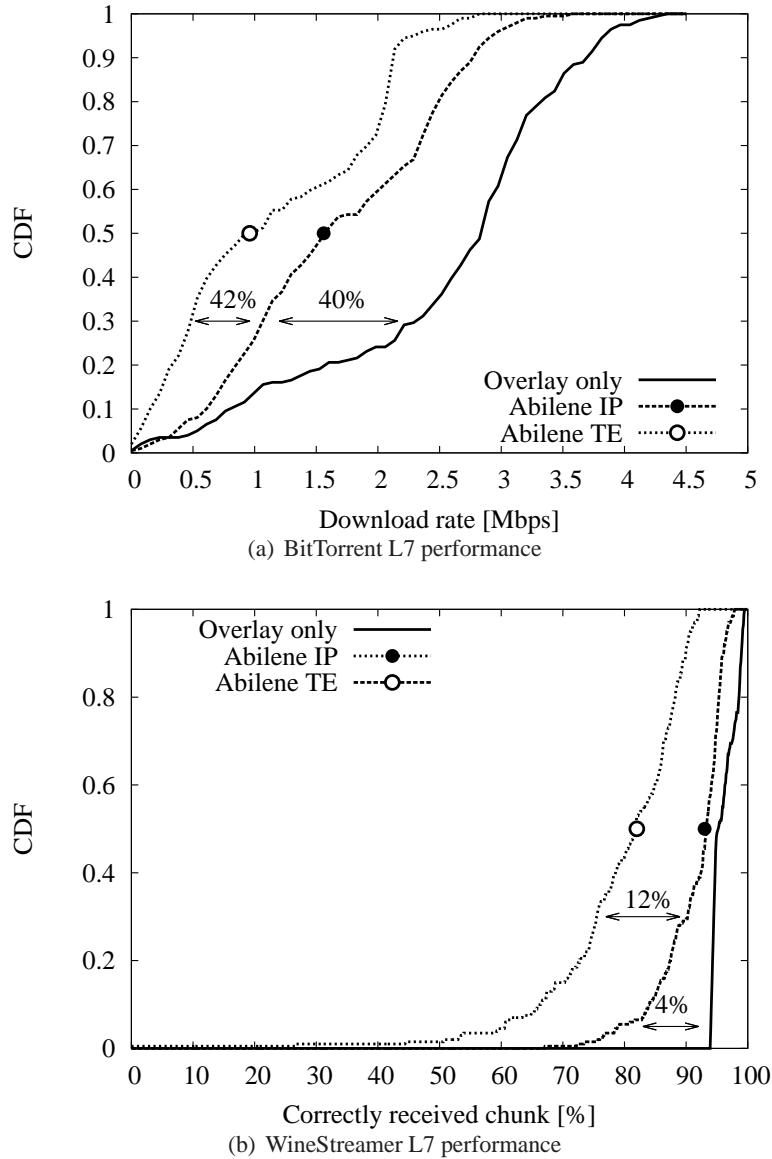
Figure 7.10: Impact of (i) Abilene vs Pure overlay network topology and (ii) IP single-path vs TE multi-path routing. Arrows are used to highlight the percentage of difference between the *mean values* of the corresponding CDF curves.

L7 algorithms. Recalling 7.8, we see that due to the highly bursty chunk transmission process, it seldom happens that independent L7 and L3 decisions increase the loads on some link. However, since this is the result of two uncoordinated decisions, it is impossible to blame a single actor between L3 or L7, as problems arise from the interplay of both. In fact, both L3 TE and L7 algorithm take decisions on the assumption that, respectively, traffic and network topology are static: thanks to ModelNet-TE we see that when this assumption no longer holds, unexpected phenomena may arise.

7.5   Conclusions

This chapter presents ModelNet-TE [70], a open source emulation tool with Traffic Engineering (TE) capabilities: building over the original ModelNet core, which only offers standard IP routing, we added the support of TE and implemented a multi-path load balancing algorithm. At the same time, our purpose was to design a flexible tool, that can be easily integrated with many other TE algorithms beyond the one that we provide.

As a case study, we use ModelNet-TE to analyze the interaction between Traffic Engineering at the network level (L3) and end-to-end control policies implemented at the application-layer (L7) by P2P application such as BitTorrent (one of the most popular file-sharing applications) and WineStreamer (a mesh-based network-aware live-streaming application). We performed a thorough experimental campaign, considering several parameters (such as topology, core link capacities, IP vs TE routing, peer population models, etc.) in the scenario definition. To gather a comprehensive understanding of the system dynamics, we express performance in terms of both network-centric and user-centric metrics: at L3, we measure link load and losses and at L3 we measure the BitTorrent download rate and WineStreamer chunk reception rate.

Our results not only validate ModelNet-TE as a complementary tool to test P2P applications in realistic environment, but also yield several interesting insights on L7/L3 dynamics. Summarizing our main findings, we have that overlay-only models yield an overly optimistic evaluation of P2P application: while the overlay-only model applies to toady's ADSL access, we have increasing evidence [23, 67] that in the near future bottlenecks may no longer sits at the user access link. Second, we observed that the population model heavily impacts overlay performance, as its impact can be of the same order of magnitude of in-network capacity limitations: hence, the ability to localize part of traffic behind the same access gateway, e.g., by means of IETF ALTO servers, seen an interesting option to offload the network and ameliorate the user experience. Third, we see that TE can noticeably worsen L7 performance: this counter-intuitive results is due to the interplay of several factors, among which (i) the impact of per-packet load balancing on TCP performance, and (ii) the uncoordinated reconfiguration of the overlay and underlay networks for unelastic applications.

While this work attempts at analyzing a large spectrum of scenarios, it also leaves many points open. As far as the experimental results are concerned, for example, it would be interesting to assess whether the conclusions gathered in this chapter are more general than the explored settings, i.e., if they continue to hold for different topologies, TE algorithms and P2P applications. As far as the tool itself is concerned, it would instead be interesting to further extend the scale of the achievable experiments, e.g., by allowing the newly introduced TE functionalities to work on multiple parallel CORES as supported by the original ModelNet core for shortest path IP routing.

Chapter 8

Conclusions

In this thesis we studied the importance of network awareness in Peer-to-Peer television: in last years we saw P2P-TV loosing traffic share with respect to the traditional client-server streaming model. However, in a near future where FTTH is widely deployed and users have enough upload capacity, P2PTV could still play a major role in the distribution of video content. In such a scenario is important for both P2P application developer and network operators to know how to deal with the traffic patterns generated by P2P-TV swarms and, at the same time, it is important to optimize as much as possible data exchange between nodes of the swarm.

8.1 Summary

Passive Analysis Our first work aimed at measuring the level of network awareness embedded in the applications that are mainly used today. We did this by setting up an European-scale testbed in which 37 machines running undisturbed P2P clients and passively collected packet traces. We then proposed a methodology to analyze data and highlight which metrics, if any, are exploited by P2P-TV applications to optimize the video delivery. Considering three popular P2P-TV applications, namely `PPLive`, `SopCast` and `TVAnts`, we have shown that only `TVAnts` and `PPLive` exhibit a mild preference to exchange data among peers in the same Autonomous System. However, no evidence of preference versus peers in the same subnet, or having a shorter path, neither the use of incentive mechanism emerge from any of the system under observation. This methodology alone, however, did not permit to analyze the so-called path-wise metric such as Round-Trip-Time, path capacity or loss rate. We thus developed another methodology which exploits the joint use of active (controlled testbed) and passive measurement technique for the analysis of the network awareness of P2P-TV system.

Hybrid Analysis The technique was designed so to consider P2P systems as a black-box, and as such can be applied to future systems as well. It consists of an active testbed in which we have a few controlled nodes running the application. With a mix of firewall rules and network emulation at kernel level we force impairments (latency, capacity, etc) on links and we register application behavior. Besides that, we exploit a passive dataset to correlate results and have a bigger and sharper understanding of the application behavior. As a case study, we applied the methodology to the analysis of `PPLive` gathering interesting results. First of all, by means of active testbed methodology, we found `PPLive` to be extremely sensitive to bandwidth, only mildly sensitive to losses and mostly unaware of IP distance, expressed

in terms of either delay or IP hop count, which is in agreement with [5]. Refining further this picture, we found that actually the peer selection process is continuously updated, with a relative preference among path-wise properties that depends on the actual magnitude of the impairment. Interestingly, by the correlation analysis of peer-wise preference gathered through the passive technique, we showed that the very same bandwidth sensitivity of PPLive, seems to induce a desirable side-effect: namely, a moderate geo-clusterization of peers within the same AS and CC.

Sherlock We then developed a general framework for the characterization of any P2P application based on a black-box measurement and analysis of the traffic they generate, coupled to an expressive data representation exploiting Kiviat graphs. We used Sherlock to analyze a number of file-sharing, VoIP, VoD and live-streaming P2P applications that are popular nowadays, further presenting two case studies, namely P2P anomaly detection and P2P network awareness. As emerges from the results, Sherlock has a number of desirable properties, which makes it a valuable tool for P2P traffic analysis. First of all, it allows a very compact representation of rather heterogeneous features and metrics, which can be furthermore easily customized as we shown. Moreover, the representation is flexible in the space domain, which is suited to express not only individual peers behavior, but also generalizes well to express the aggregated peer behavior (e.g., mean) and its variability (e.g., standard deviation). The representation is also flexible in the time domain, which allows to observe not only the long-term behavior of P2P applications, but the temporal system evolution as well. Finally, Sherlock is generally applicable, in virtue of its black-box approach, which is important in reason of both the varying popularity of Internet applications and the closeness of popular P2P applications. Notably we implemented Sherlock in a demonstration software (P2PGauge) that measures and displays the network awareness using multiple representation (maps, probability distributions, kiviat charts). Starting from the understanding that no conclusions can be drawn without jointly leverage passive and active measurement, P2PGauge computes statistics exploiting both peer-wise (Autonomous System, Country, IP) and path-wise (RTT, hop count, path capacity). We applied our method to SopCast which showed a greedy preference toward high capacity nodes coupled with a negligible preference towards topologically close peers.

In the second part of the thesis we focused on the testing of network aware algorithms and applications in controlled environment: particularly we wanted to assess (i) how much measurement errors on network property impact overlay performance and (ii) which are the interactions of L7 and L3 when traffic engineering algorithms (TE) change periodically the underlying topology configuration.

P2PTV Simulation In Chapter 6 we compared different state-of-art "network-aware" algorithms by means of a simulation campaign. We defined a flexible framework, which can be easily extended with new components: our purpose was to understand what were the main factors that affect P2P-TV performance, and to what extent performance degraded under realistic settings. We modified P2PTV-SIM a chunk level simulator conceived in the NapaWine project; P2PTV-SIM takes into account several factors such as heterogeneous class of peers, access link capacity, link latency and exposes to diffusion algorithms information about L3 network and nodes; moreover this information can possibly be changed by an error model. Our main findings can be summarized as follows. First, we found the impact of the L3 underlay network model to be modest, with a small performance gap between simple (e.g., constant and fixed delay) and realistic models (e.g., meridian latency or dynamic latencies).

This owes to the fact that the propagation delay has a smaller impact with respect to transmission delay, especially considering the relative low-capacity of current scenario. At the same time, we can expect that, as the access capacity increases, the impact of propagation latency may need to be reconsidered. Second, we found that system performance was rather robust to errors in the measurement of peer properties. More precisely, provided that peers capacity was clearly separated, the ability to roughly discriminate high-capacity from low-capacity peer was sufficient to guarantee a good level of performance. Similarly, errors in the latency estimation only affected the traffic locality, but system performance were otherwise unaltered. Finally, we found the impact of signaling errors to be, by far, the most important factor able to significantly degrade the quality of P2P-TV services and was able to severely impact overlay performance already from very low intensities. This suggests P2P-TV protocol designers to pay special attention to signaling logic, in order to gather reliable estimate of the achievable system performance.

P2P Emulation Finally we presented ModelNet-TE [70], a open source emulation tool with Traffic Engineering (TE) capabilities: building over the original ModelNet core, which only offers standard IP routing, we added the support for Traffic Engineering (TE) and implemented a multi-path load balancing algorithm. At the same time, our purpose was to design a flexible tool, that could be easily integrated with many other TE algorithms beyond the one that we provide. As a case study, we used ModelNet-TE to analyze the interaction between Traffic Engineering at the network level (L3) and end-to-end control policies implemented at the application-layer (L7) by P2P application such as BitTorrent (one of the most popular file-sharing applications) and WineStreamer (a mesh-based network-aware live-streaming application). We performed a thorough experimental campaign, considering several parameters (such as topology, core link capacities, IP vs TE routing, peer population models, etc.) in the scenario definition. To gather a comprehensive understanding of the system dynamics, we expressed performance in terms of both network-centric and user-centric metrics: at L3, we measured link load and losses and at L3 we measured the BitTorrent download rate and WineStreamer chunk reception rate. Our results not only validated ModelNet-TE as a complementary tool to test P2P applications in realistic environment, but also yielded several interesting insights on L7/L3 dynamics. Summarizing our main findings, we found that overlay-only models yielded an overly optimistic evaluation of P2P application: while the overlay-only model applies to today's ADSL access, we have increasing evidence [23, 67] that in the near future bottlenecks may no longer sits at the user access link. Second, we observed that the population model heavily impacts overlay performance, as its impact can be of the same order of magnitude of in-network capacity limitations: hence, the ability to localize part of traffic behind the same access gateway, e.g., by means of IETF ALTO servers, seen an interesting option to offload the network and ameliorate the user experience. Third, we saw that TE could noticeably worsen L7 performance: this counter-intuitive results was due to the interplay of several factors, among which (i) the impact of per-packet load balancing on TCP performance, and (ii) the uncoordinated reconfiguration of the overlay and underlay networks for unelastic applications.

8.2   Future Work

Despite we did our best to make this work as complete as possible, there are inevitably points that we did not deal with for lack of time, or which would require supplementary year(s) to be discussed. In the following, we present issues that we think are still open and we would like to

pursue in the future.

Next generation applications Frameworks and analysis methods for the measure of network awareness presented in first part of this thesis have been applied to P2P-TV applications that were the state of the art in 2008 and 2009. During the last years however, P2P developers improved their existing solutions and different network aware approaches have been studied by other project as well (P2PNext [77] or P4P [124]). Our opinion is that a comparative study assessing the improvement done in recent years is necessary. Also, it could be interesting to monitor applications, for instance by means of Sherlock, over a period of time. In fact, studies as the one presented in chapter 3 are useful but their main limit is that they refer to a snapshot of time; it would be more valuable to apply the very same methodology continuously over the years, to monitor the evolution and usage of P2P applications.

Churn Chapters 6 and 7 analyze swarm performance in realistic scenarios and on stable conditions but do not take into account one of the issues in P2P streaming: peer churn. Although it is true that population is almost constant during popular programs, still sudden departures or flash-crowd arrivals could mine the stability of the system and degrade user-perceived quality of experience. The problem is complex and worth to be studied, e.g. by measuring real life churn behavior (channel switch, mean duration time) and then embed a derived model in the simulator of Chapter 6.

Signaling dynamics Chapter 6 concludes that signaling errors are the biggest cause of quality degradation; this result depends on certain parameters such as chunk-size, buffer-map size and buffer-map update rate. An interesting future work is to study the interplay between those parameters and exploring the trade-off between the overhead caused by high signaling rate and the outdated knowledge of neighborhood's buffer-maps due to low signaling rate; eventually develop novel signaling strategies exploiting buffer-map compression or prediction techniques. Furthermore, since the signaling model used in simulation was oversimplified, it would be interesting to implement actual signaling protocols for cross-check.

Parallel Modelnet Core As far as ModelNet-TE is concerned, it would be useful to further extend the scale of the achievable experiments, e.g., by allowing the newly introduced TE functionality to work on multiple parallel CORES. Although this is supported by the original ModelNet core for shortest path IP routing, in the case of TE algorithm the issue is more complex as, at each computation interval, the master core must query all slaves for traffic load information, re-compute forwarding tables and dispatch them to slaves. This operation however should be done carefully since control information is exchanged in-bandwidth and should not deteriorate application performance.

New Topologies and TE algorithms In Chapter 7 we showed that traffic engineering, whereas reduced losses and congestions on highly loaded link, most of the times it did not ameliorate P2P-TV performance. In fact, an improved quality of service (QoS), e.g., link load/loss fairness, is not enough to have higher quality of experience (QoE); load balancers or frequent route changes, while ameliorating global network performance, could annoy applications by reordering packets or, if application is preforming network measures, by interfering with its decisions. Besides that, we know from our experiments that there are particular topologies in which the action of TE can aid improving both the QoE and the QoS. This leads to the need of a sensitivity analysis to understand which TE algorithm/topology performs better and give developers and operator general guidelines (e.g. frequency of network measures,

connectivity degree of L3 network, creation of critical backup links) about development of new generation applications and good deployment of network infrastructure.

Web Integration It is well known that nowadays YouTube is generating huge amount of video traffic and it is the second search engine in the US. If we imagine a future in which video is only in High Definition, the amount of video data served by YouTube servers and the surrounding Content Distribution Network could be too high to support. A possible solution would be to implement and standardize P2P trading logic in future browsers (for instance through HTML5 `<VIDEO>` tag) so that each user could, at least, cache very popular contents and lighten the load on servers.

Appendix A

List of publications

We report here the list of publications and papers under submission related to this thesis.

A.1 Published

- O. Morandi, F. Risso, P. Rolando, S. Valenti and P. Veglia, Creating Portable and Efficient Packet Processing Applications, In *Design Automation for Embedded Systems, Volume 15, Number 1, 51-85,*, Mar. 2011.

- D. Rossi and P. Veglia, Assessing the impact of signaling on the QoE of push-based P2P-TV diffusion algorithms, In *Proc. of IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Paris, France, Feb. 2011.

- D. Ciullo, M. A. Garcia, A. Horvath, E. Leonardi, M. Mellia, D. Rossi, M. Telek and P. Veglia, Network Awareness of P2P Live Streaming Applications: A Measurement Study, In *IEEE Transactions on Multimedia, Volume 12, Issue:1, 54-63,*, Jan. 2010.

- D. Rossi, E. Sottile and P. Veglia, Black-box analysis of Internet P2P applications, In *Peer-to-Peer Networking and Applications, Volume 4, Number 2, 146-164,*, Jun. 2010.

- D. Rossi and P. Veglia, A Hybrid Approach to Assess the Network Awareness of P2P-TV Applications, In *International Journal of Digital Multimedia Broadcasting, Volume 2010, Article ID 826351,*, Nov. 2010.

- D. Ciullo, M. Garcia, A. Horvath, E. Leonardi, M. Mellia, D. Rossi, M. Telek and P. Veglia, Network Awareness of P2P Live Streaming Applications, In *Proc. of Sixth International Workshop on Hot Topics in Peer-to-Peer Systems,* Rome, Italy, May. 2009.

- D. Rossi, E. Sottile, S. Valenti and P. Veglia, Gauging the network friendliness of P2P applications, In *SIGCOMM demo session,* Barcelona, Spain, Aug. 2009.

- D. Rossi, S. Valenti, P. Veglia and D. Bonfiglio, Pictures from the Skype, In *ACM Performance Evaluation Review,* Sep. 2008.

- O. Morandi, F. Risso, S. Valenti and P. Veglia, Design and implementation of a framework for creating portable and efficient packet-processing, In Proc. of International Conference on Embedded Software, Atlanta, USA, Oct. 2008.

A.2   Under Review

- P. Veglia, D. Rossi, Performance evaluation of P2P-TV diffusion algorithms under realistic settings, *submitted to Springer Peer-to-Peer Networking and Applications*.

- P. Veglia, D. Rossi, ModelNet-TE: An emulation tool for the study of P2P and Traffic Engineering interaction dynamics, *submitted to Springer Peer-to-Peer Networking and Applications*.

Bibliography

[1] Abilene network. `http://www.internet2.edu/network/`.

[2] S. Agarwal, J. P. Singh, A. Mavlankar, P. Baccichet, , and B. Girod. Performance and Quality-of-Service Analysis of a Live P2P Video Multicast Session on the Internet. In *IEEE IwQoS*, Enschede, NL, Jun. 2008.

[3] V. Aggarwal, A. Feldmann, and C. Scheideler. Can ISPS and P2P users cooperate for improved performance? *ACM SIGCOMM Computer Communication Review*, 37(3):29, luglio 2007.

[4] A. Akella, S. Seshan, and A. Shaikh. An empirical evaluation of wide-area internet bottle-necks. In *Proc. of the 3rd ACM SIGCOMM Conference on Internet Measurement (IMC'03)*, Miami, FL, USA, Oct. 2003.

[5] E. Alessandria, M. Gallo, E. Leonardi, M. Mellia, and M. Meo. P2P-TV systems under adverse network conditions: a measurement study. In *In Proc. of IEEE INFOCOM*, Rio de Janeiro, Brazil, Apr. 2009.

[6] Alto ietf working group. `http://datatracker.ietf.org/wg/alto/charter/`.

[7] P. Auer, N. Cesa-Bianchi, and C. Gentile. Adaptive and self-confident on-line learning algorithms. *Elsevier Journal of Computer and System Sciences*, 64(1):48 – 75, 2002.

[8] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding traceroute anomalies with paris traceroute. In *Proc. of ACM SIGCOMM Internet Measurement Conference, (IMC'06)*, Rio de Janeiro, Brazil, Oct. 2006.

[9] Bandwidth-test.net. Bandwidth test statistics across different countries. `http://www.bandwidth-test.net/stats/country/`.

[10] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *In Proc. of ACM SIGCOMM*, Pittsburgh, PA, USA, Aug. 2002.

[11] A. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509, 1999.

[12] R. Bindal, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang. Improving traffic local-ity in bittorrent via biased neighbor selection. In *In Proc. of IEEE International Conference on Distributed Computing Systems, (ICDCS'06)*, Lisboa, POR, Jul. 2006.

[13] R. Birke, C. Kiraly, E. Leonardi, M. Mellia, M. Meo, and S. Traverso. Hose rate control for P2P-TV streaming systems. In *Proc. of IEEE International Conference on Peer-to-Peer Computing (P2P'11)*, Kyoto, Japan, Sep. 2011.

[14] R. Birke, E. Leonardi, M. Mellia, A. Bakay, T. Szemethy, C. Kiraly, R. L. Cigno, F. Mathieu, L. Muscariello, S. Niccolini, J. Seedorf, and G. Tropea. Architecture of a network-aware P2P-TV application: the napa-wine approach. *IEEE Communication Magazine*, 49, Jun. 2011.

[15] BitTorrent home page. `http://www.bittorrent.com`.

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

[17] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, and A. Twigg. Epidemic live streaming: optimal performance trade-offs. In *ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, Annapolis, USA, 2008.

[18] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth multicast in a cooperative environment. In *In Proc. of ACM SOSP*, Lake Bolton, NY, Oct. 2003.

[19] H. Chang, S. Jamin, and W. Wang. Live streaming performance of the Zattoo network. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*. ACM, 2009.

[20] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang. Enabling conferencing applications on the internet using an overlay multicast architecture. In *In Proc. of ACM SIGCOMM*, San Diego, CA, USA, Aug. 2001.

[21] Cisco visual networking index: Forecast and methodology, 2010–2015. `http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf`.

[22] D. Ciullo, M. A. Garcia, H. Akos, E. Leonardi, M. Mellia, D. Rossi, M. Telek, and P. Veglia. Network awareness of P2P live streaming applications: A measurement study. *IEEE Transaction on Multimedia*, 12(1):54 –63, Jan. 2010.

[23] Comcast discloses throttling practices — bittorrent targeted. `http://www.wired.com/threatlevel/2008/09/comcast-disclos/`.

[24] Pearson product moment correlation coefficient. `http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient`.

[25] D. Croce, M. Mellia, and E. Leonardi. The quest for bandwidth estimation techniques for large-scale distributed systems. *ACM SIGMETRICS Performance Evaluation Review*, 37(3):20–25, 2009.

[26] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[27] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, 1987.

[28] L. Dai, Y. Cao, Y. Cui, and Y. Xue. On scalability of proximity-aware peer-to-peer streaming. *Elesevier Computer Communications*, 32(1):144 – 153, 2009.

[29] C. Dale, J. Liu, J. Peters, and B. Li. Evolution and enhancement of bittorrent network topologies. In *In Proc. of ACM/IEEE International Workshop on Quality of Service (IWQoS'08)*, Twente, The Netherlands, Jun. 2008.

[30] D. DiPalantino and R. Johari. Traffic engineering vs. content distribution: A game theoretic perspective. In *Proc. of IEEE INFOCOM*, Rio de Janeiro, Brazil, Apr. 2009.

[31] Federico Larroca. *Techniques d'Ingénierie de Trafic Dynamique pour l'Internet*. PhD thesis, Telecom ParisTech, 2009.

[32] A. Finamore, M. Mellia, M. Meo, and D. Rossi. Kiss: Stochastic packet inspection classifier for udp traffic. *IEEE Transactions on Networking*, 18(5):1505 – 1515, October 2010.

[33] A. Finamore, M. Mellia, M. Meo, D. Rossi, and S. Valenti. Kiss to Abacus: a comparison of P2P-TV traffic classifiers. In *Traffic Measurement and Analyis (TMA'10), LNCS*, Zurich, Switzerland, April 2010.

[34] A. Finamore, M. Mellia, M. Meo, D. Rossi, and S. Valenti. Peer-to-peer traffic classification: exploiting human communication dynamics. In *Globecom'10 Demo session*, Miami, USA, December 2010. IEEE.

[35] G. Fox and S. Pallickara. The Narada event brokering system: Overview and extensions. In *Parallel and Distributed Processing Techniques and Applications, PDPTA '02*, Las Vegas, USA, June 2002.

[36] Grid5000 home page. `http://www.grid5000.fr/`.

[37] I. W. Group. Application-layer traffic optimization (alto). `http://www.ietf.org/html.charters/alto-charter.html`.

[38] K. Gummadi, S. Saroiu, and S. Gribble. King: Estimating latency between arbitrary Internet end hosts. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*, Marseille, FRA, 2002.

[39] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava. PROMISE: peer-to-peer media streaming using CollectCast. In *In Proc. of the eleventh ACM international conference on Multimedia*, page 54. ACM, 2003.

[40] N. Hegde, F. Mathieu, and D. Perino. On optimizing for epidemic live streaming. In *Proc. of IEEE International Conference on Communications (ICC)*. IEEE, 2010.

[41] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross. A measurement study of a large-scale P2P IPTV system. *IEEE Transactions on Multimedia*, 4(8):1672–1687, Dec. 2007.

[42] X. Hei, Y. Liu, and K. Ross. Inferring network-wide quality in P2P live streaming systems. *IEEE JSAC, special issue on P2P Streaming*, 25(9):1640–1654, Dec. 2007.

[43] K. M. Hendrik Schulze. Internet study 2008/2009. Technical report, ipoque, 2008-2009.

[44] The internet after dark. `http://asert.arbornetworks.com/2009/08/the-internet-after-dark/`.

[45] W. Jiang, D.-M. Chiu, and J. C. S. Lui. On the interaction of multiple overlay routing. *Elesevier Performormance Evaluation*, 62(1-4):229–246, 2005.

[46] R. Kapoor, L.-J. Chen, L. Lao, M. Gerla, and M. Y. Sanadidi. Capprobe: a simple and accurate capacity estimation technique. *SIGCOMM Comput. Commun. Rev.*, 34(4):67–78, 2004.

[47] R. Keralapura, C.-N. Chuah, N. Taft, and G. Iannaccone. Can ISPs take the heat from overlay networks? In *In Proc. of ACM Workshop on Hot Topics in Networks (HotNets-III)*, San Diego, CA, USA, November 2004.

[48] Internet traffic classification demo webpage. `http://perso. telecom-paristech.fr/~drossi/index.php?n=Software. ClassificationDemo`.

[49] J. Klaue, B. Rathke, and A. Wolisz. Evalvid – a framework for video transmission and quality evaluation. In *Computer Performance*, volume 2794 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2003.

[50] K. Kolence and P. Kiviat. Software unit profiles and kiviat figures. *ACM SIGMETRICS Performance Evaluation Review*, 2(3):2–12, Sep. 1973.

[51] S. Kullback and R. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[52] Latency in fibre optics. `http://en.wikipedia.org/wiki/Latency_ (engineering)#Fibre_Optics`.

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

[54] A. Legout, G. Urvoy-Keller, and P. Michiardi. Rarest first and choke algorithms are enough. In *Proc. of ACM SIGCOMM Internet Measurement Conference, (IMC'06)*, Rio de Janeiro, Brazil, Oct. 2006.

[55] E. Leonardi, M. Mellia, A. Horvath, L. Muscariello, S. Niccolini, and D. Rossi. Building a cooperative P2P-TV application over a wise network: the approach of the european fp-7 strep napa-wine. *IEEE Communication Magazine*, 64(6), April 2008.

[56] B. Li, Y. Qu, Y. Keung, S. Xie, C. Lin, J. Liu, and X. Zhang. Inside the new coolstreaming: Principles, measurements and performance implications. In *IEEE INFOCOM*, Phoenix, AZ, USA, Apr. 2008.

[57] Y. Liu, L. Guo, F. Li, and S. Chen. A case study of traffic locality in internet P2P live streaming systems. In *In Proc. of IEEE International Conference on Distributed Computing Systems. (ICDCS '09)*, Montreal, Quebec, Canada, Jun. 2009.

[58] Y. Liu, H. Zhang, W. Gong, and D. Towsley. On the interaction between overlay routing and underlay routing. In *Proc. of IEEE INFOCOM*, Miami, FL, USA, Mar. 2005.

[59] R. J. Lobb, A. P. Couto da Silva, E. Leonardi, M. Mellia, and M. Meo. Adaptive overlay topology for mesh-based P2P-TV systems. In *In Proc. of International Workshop on Network and Operating Systems Support for Digital Audio and Video, (NOSSDAV'09)*, Williamsburg, USA, 2009.

[60] N. Magharei and R. Rejaie. Mesh or multiple-tree: A comparative study of P2P live streaming services. In *Infocom 2007*, Anchorage, Alaska, May 2007. IEEE.

[61] N. Magharei and R. Rejaie. Prime: Peer-to-peer receiver-driven mesh-based streaming. *IEEE/ACM Transaction on Networking*, 17(4), 2009.

[62] D. Manzato and N. da Fonseca. Incentive mechanism for the CoopNet network. *Peer-to-Peer Networking and Applications*, 1:29–44, 2008.

[63] P. Marciniak, N. Liogkas, A. Legout, and E. Kohler. Small is not always beautiful. In *Proc. of 7th International workshop on Peer-To-Peer Systems (IPTPS'07)*, Tampa, FL, USA, Sep. 2008.

[64] F. Mathieu. Heterogeneity in data-driven live streaming: Blessing or curse? In *IEEE International Symposium on Parallel Distributed Processing (IPDPS)*, pages 1–8, April 2010.

[65] Maxmind geolite. `http://www.maxmind.com/`.

[66] A. McGregor, M. Hall, P. Lorier, and J. Brunskill. Flow clustering using machine learning techniques. In *In Proc. of Active and Passive Measurement Conference*, Antibes Juan-les-Pins, France, Apr. 2004.

[67] Megaupload accuse orange de ralentissements. `http://info.france2.fr/sciences-tech/megaupload-accuse-orange-de-ralentissements-66896673.html`.

[68] M. Mellia, M. Meo, L. Muscariello, and D. Rossi. Passive analysis of tcp anomalies. *Elsevier Computer Networks*, 52(14), October 2008.

[69] Meridian project. `http://www.cs.cornell.edu/People/egs/meridian/`.

[70] ModelNet-TE Web page. `http://perso.telecom-paristech.fr/~drossi/index.php?n=Software.ModelNet-TE`.

[71] F. D. R. C. F. K. R. Morris. Vivaldi: a decentralized network coordinate system. In *In Proc. of ACM SIGCOMM*, Portland, USA, 2004.

[72] L. Muscariello, D. Perino, and D. Rossi. Do next generation networks need path diversity? In *Proc. of IEEE International Conference on Communications, (ICC'09)*, Dresden, Germany, Jun. 2009.

[73] Network-aware P2P-TV application over wise networks. `http://www.napa-wine.eu`.

[74] Onelab home page. `http://www.onelab.eu`.

[75] A. Orda, R. Rom, and N. Shimkin. Competitive routing in multiuser communication networks. *IEEE/ACM Transactions on Networking*, 1(5):510–521, 1993.

[76] P2PGauge home page. `http://www.infres.enst.fr/~drossi/P2PGauge`.

[77] P2PNext home page. `http://www.p2p-next.org/`.

[78] P2PTV-Sim Home Page. `http://www.napa-wine.eu/cgi-bin/twiki/view/Public/P2PTVSim`.

[79] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking. In *In Proc. of ACM NOSSDAV*, Miami, FL, May 2002.

[80] V. Pai, K. Kumar, K. T. andy Vinay Sambamurthy, and A. E. Mohr. Chainsaw: Eliminating trees from overlay multicast. *Lecture Notes in Computer Science*, 2005.

[81] K. Pearson. *On Lines and Planes of Closest Fit to Systems of Points is Space*.

[82] F. Pianese, D. Perino, J. Keller, and E. Biersack. PULSE: an adaptive, incentive-based, unstructured P2P live streaming system. *IEEE Transactions on Multimedia*, 9(8):1645–1660, 2007.

[83] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do incentives build robustness in BitTorrent? In *Symposium on Networked System Design & Implementation, NSDI'07*, Cambridge, USA, 2007. USENIX.

[84] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do incentives build robustness in bittorrent. In *Proc. of 4th USENIX Symposium on Networked Systems Design & Implementation (NSDI'07)*, Cambridge, MA, USA, Apr. 2007.

[85] M. Piatek, H. V. Madhyastha, J. P. John, A. Krishnamurthy, and T. Anderson. Pitfalls for isp-friendly P2P design. In *ACM Workshop on Hot Topics in Networks (HotNets-VIII)*, New York City, USA, 2009.

[86] F. Picconi and L. Massoulié. Is there a future for mesh-based live video streaming? In *Proc. of IEEE International Conference on Peer-to-Peer Computing (P2P'08)*, Aachen, Germany, Sep. 2008.

[87] F. Picconi and L. Massoulié. Isp friend or foe? making P2P live streaming isp-aware. In *In Proc. of IEEE International Conference on Distributed Computing Systems (ICDCS'09)*, Montreal, Quebec, Canada, 2009. IEEE.

[88] Planetlab home page. `http://www.planet-lab.org`.

[89] PPLive. `http://www.pplive.com`.

[90] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker. On selfish routing in internet-like environments. In *in Proc. of ACM SIGCOMM*, Karlsruhe, Germany, Aug. 2003.

[91] A. Rao, A. Legout, and W. Dabbous. Can Realistic BitTorrent Experiments Be Performed on Clusters? In *IEEE International Conference on Peer-to-Peer Computing (P2P'10)*, Delft, Netherlands, Aug. 2010.

[92] R. Rejaie and A. Ortega. PALS: peer-to-peer adaptive layered streaming. In *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video, NOSSDAV '03*, Monterey, USA, 2003. ACM.

[93] D. Ren, Y. Li, and S. Chan. On Reducing Mesh Delay for Peer-to-Peer Live Streaming. In *In Proc. of IEEE Infocom*, Phoenix, AZ, USA, Apr. 2008.

[94] D. Rossi, E. Sottile, S. Valenti, and P. Veglia. Gauging the network friendliness of P2P applications. In *In Proc. of SIGCOMM demo session*, Barcelona, Spain, Aug. 2009.

[95] D. Rossi, E. Sottile, and P. Veglia. Black-box analysis of internet P2P applications. *Peer-to-Peer Networking and Applications*, 4:146–164, 2011.

[96] D. Rossi, C. Testa, S. Valenti, and L. Muscariello. Ledbat: the new bittorrent congestion control protocol. In *Proc. of International Conference on Computer Communication Networks (ICCCN'10)*, Zurich, Switzerland, Aug. 2010.

[97] D. Rossi and P. Veglia. An hybrid approach to assess the network awareness of P2P-TV applications. *International Journal of Digital Multimedia Broadcasting, SI on Network-Aware Peer-to-Peer (P2P) and Internet Video*, 2010.

[98] D. Rossi and P. Veglia. Assessing the impact of signaling on the QoE of push-based P2P-TV diffusion algorithms. In *Conference on New Technologies, Mobility and Security, NTMS'11*, Paris, FRA, 2011. IEEE.

[99] T. Roughgarden and Éva Tardos. How bad is selfish routing. *Journal of the ACM*, 49:236–259, 2002.

[100] S.Ali, A.Mathur, and H.Zhang. Measurements of commercial peer-to-peer live video streaming. In *In Proc. of Workshop on Recent Advances in Peer-to-Peer Streaming*, Waterloo, ON, Aug. 2006.

[101] J. Seibert, D. Zage, S. Fahmy, and C. Nita-Rotaru. Experimental comparison of peer-to-peer streaming overlays: An application perspective. In *IEEE Conference on Local Computer Networks, LCN'08*, Montreal, Canada, 2008.

[102] R. Sherwood, S. Lee, and B. Bhattacharjee. Cooperative peer groups in NICE. *Computer Networks*, 50(4):523–544, 2006.

[103] A. Shriram, M. Murray, Y. Hyun, N. Brownlee, A. Broido, M. Fomenkov, and K. Claffy. Comparison of public end-to-end bandwidth estimation tools on high-speed links. In *In Proc. of Passive and Active Measurement conference, (PAM'05)*, Boston, USA, 2005.

[104] A. P. C. D. Silva, E. Leonardi, M. Mellia, and M. Meo. A Bandwidth-Aware Scheduling Strategy for P2P-TV Systems. In *Peer to Peer computing, P2P'08*, Aachen, GER, 2008. IEEE.

[105] T. Silverston and O. Fourmaux. Measuring P2P IPTV systems. In *ACM NOSSDAV*, Urbana-Champaign, IL, USA, Jun. 2007.

[106] T. Silverston, O. Fourmaux, K. Salamatian, and K. Cho. On Fairness and Locality in P2P-TV through Large-Scale Measurement Experiment. In *Proc. of IEEE GLOBECOM*, Miami, FL, USA, Dec. 2010.

[107] SopCast. http://www.sopcast.com.

[108] N. Spring, L. Peterson, A. Bavier, and V. Pai. Using planetlab for network research: myths, realities, and best practices. *SIGOPS Operating Systems Review*, 40:17–24, January 2006.

[109] Synacast. Synacast corporation. http://www.synacast.com/en/.

[110] C. Testa and D. Rossi. The impact of uTP on bittorrent completion time. In *Proc. of IEEE International Conference on Peer-to-Peer Computing (P2P'11)*, Kyoto, Japan, Sep 2011.

[111] D. Tran, K. Hua, and T. Do. A peer-to-peer architecture for media streaming. *IEEE Journal on Selected Areas in Communications*, 22(1):121 – 133, Jan. 2004.

[112] TVAnts. `http://www.tvants.com`.

[113] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostic, J. Chase, and D. Becker. Scalability and accuracy in a large-scale network emulator. In *Proc. of 5th USENIX Symposium on Operating Systems Design and Implementation (OSDI'02)*, 2002.

[114] S. Valenti, D. R. amd M. Meo, M.Mellia, and P. Bermolen. Abacus: Accurate behavioral classification of P2P-TV traffic. *Elsevier Computer Networking, to appear*, 2010.

[115] L. Vu, I. Gupta, J. Liang, and K. Nahrstedt. Measurement of a large-scale overlay for multimedia streaming. In *In Proc. of the 16th International Symposium on High Performance Distributed Computing*, Monterey, CA, USA, Jun. 2007.

[116] F. Wang, J. Liu, and Y. Xiong. Stable peers: Existence, importance, and application in peer-to-peer live video streaming. In *IEEE INFOCOM*, Phoenix, AZ, USA, Apr. 2008.

[117] D. Watts and S. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393(6684), 1998.

[118] Table of united states metropolitan statistical areas. `http://en.wikipedia.org/wiki/Table_of_United_States_Metropolitan_Statistical_Areas`.

[119] Winestreamer web page. `http://napa-wine.eu/cgi-bin/twiki/view/Public/WineStreamer`.

[120] B. Wong, A. Slivkins, and E. G. Sirer. Meridian: A lightweight network location service without virtual coordinates. In *Proc. of ACM SIGCOMM*, Philadelphia, Pennsylvania, USA, Aug 2005.

[121] C. Wu, B. Li, and S.Zhao. Exploring large-scale peer-to-peer live streaming topologies. *IEEE Transaction on Multimedia Computing, Communications and Applications*, 4(3), 2008.

[122] C. Wu, B. Li, and S. Zhao. Multi-channel live P2P streaming: Refocusing on servers. In *IEEE INFOCOM*, Phoenix, AZ, USA, Apr. 2008.

[123] D. Wu, P. Dhungel, X. Hei, C. Zhang, and K. Ross. Understanding peer exchange in bittorrent systems. In *Proc. of IEEE International Conference on Peer-to-Peer Computing (P2P'10)*, Delft, Netherlands, Aug. 2010.

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

[125] D. Xu, M. Chiang, and J. Rexford. Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering. *IEEE/ACM Transactions on Networking*, PP(99):1, 2011.

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

[127] H. Zhang, Y. Liu, W. Gong, and D. Towsley. On the interaction between overlay routing and mpls traffic engineering. In *In Proc. of ACM SIGCOMM, Poster Session*, Portland, OR, USA, 2004.

[128] J. Zhang, L. Liu, L. Ramaswamy, and C. Pu. PeerCast: Churn-resilient end system multi-cast on heterogeneous overlay networks. *Journal of Network and Computer Applications*, 31(4):821–850, 2008.

[129] X. Zhang, J. Liu, B. Li, and T. Yum. CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming. In *proceedings of IEEE Infocom*, volume 3, pages 13–17. Citeseer, 2005.

[130] B. Zhao, J. Lui, and D. Chiu. Exploring the optimal chunk selection policy for data-driven P2P streaming systems. In *Peer to Peer computing, P2P'09*, Seattle, USA, 2009. IEEE.