Diffusion de contenu en pair-à-pair

Mathieu Goessens

Rapport de stage
Diffusion de contenu en pair-à-pair

Master II - Informatique - 2011/2012

Mathieu Goessens
Sous la direction de Davide Frey
5 Juin 2012

Table des matières

Résumé

L'utilisation grandissante du réseau Internet et du Web comme moyen privilégié de diffusion de l'information fait émerger de nouveaux défis, pour lesquels le modèle client-serveur classique tend à monter ses limites en terme de robustesse et de monté en charge. Nous proposons un nouveau système : behaveCDN. Celui-ci permet la diffusion de contenu Web, en utilisant les algorithmes pair-à-pair non structurés pour construire un réseau basé sur les modèles sociaux. Ce modèle, nous permet de proposer un système de cache distribué, capable de tenir compte des intérêts des utilisateurs. Il permet d'augmenter le potentiel de montée en charge des infrastructures existantes et de préserver l'expérience utilisateur lors d'éventuelles surcharges. Une implémentation est disponible afin d'évaluer ce modèle, et peut être réutilisée comme plate-forme d'expérimentation réelle pour les problématiques liées aux modèles pair-à-pair ou au respect de la vie privée.

Remerciements *L'auteur tient à remercier les personnes suivantes, pour l'aide qu'elles lui ont apporté, leur soutien, leurs conseils ou simplement pour, d'une manière ou d'une autre, avoir rendu tout ceci possible : Anne-Marie, Antony, Arnaud, César, Cecile, Davide, Delphine, Erwan, Frédéric, Gwendal, Heverson, Jérome, Julien, Laurent, Stephane, Stephane, Sonny, Sylvain, Yassine.*

1 Introduction

L'utilisation grandissante du réseau Internet et du Web comme moyen privilégié de diffusion de l'information fait émerger de nouveaux défis, pour lesquels les modèles classiques tendent à montrer leurs limites.

Le modèle client-serveur est le modèle classique utilisé sur Internet. Celui-ci repose principe simple : les utilisateurs voulant accéder à un contenu (les clients) vont se connecter à un serveur afin d'obtenir celui-ci. Bien que simple et efficace cette approche tend à montrer ses limites. Sa nature fortement centralisée réduit fortement sa robustesse et ses possibilité de montée en charge.

Durant la fin des années 1990 et le début des années 2000, et devant l'émergence de contenus toujours plus riches, au regarde des connexions de l'époque, de nombreux projets ont proposé des solutions, visant à augmenter les performances du modèle client-serveur, via l'utilisation de serveurs relais (proxy) [8, 52], ou de système de distribution de contenus en pair-pair [58]. La popularisation de connexions haut débits, disponibles aussi bien pour les utilisateurs que pour les fournisseurs de contenu ont toutefois diminué l'intérêt pour ces travaux, à l'image du projet renater cache abandonné en 2000.

Aujourd'hui, la croissance quasi exponentielle du réseau, et l'émergence de nouveaux usages comme la diffusion de contenu de plus en plus riches (images, fichiers sonores, vidéos...) font émerger à nouveaux ces problématiques. Alors que le poids des pages augmente de manière constante depuis le début des années 2000, il convient de s'interroger sur la possibilité de montée en charge et de passage à l'échelle de ce modèle centralisé, tout comme son aptitude à concilier le maintient d'une bonne expérience utilisateur et la maîtrise des coûts.

Bien que de nombreux projets aient proposé la mise en place de systèmes alternatifs pour la diffusion des contenus les plus riches (vidéos et fichiers sonores), via l'utilisation de technologie spécifiques, les fournisseurs de contenus délaissent les systèmes spécifiques, au profit de modèles entièrement basés sur le Web via l'utilisation des technologies HTML5, augmentant ainsi la pression sur les technologies et l'intérêt de concevoir un modèle de diffusion de l'information adapté à celui-ci.

Pour répondre à ces problématiques de robustesse du réseau vis à vis de la montée en charge et de résistance aux attaques. De nouveaux modèles de diffusion de l'information ont été proposés. Les Content Distribution Networks ou CDN sont une approche industrielle visant à résoudre les problèmes de montée en charge via l'utilisation de plusieurs serveurs, placés au plus près des utilisateurs (réutilisant ainsi d'une certaines manières les concepts de cache locaux évoqués précédemment). Les modèles pair-à-pair sont une seconde approche, reposant sur un modèle décentralisé et permettant ainsi la création de réseaux à très large échelle. Ceux-ci ouvrent la voie vers une diffusion de l'information

par les utilisateurs eux même et la création de CDN participatifs.

À L'image du projet CoralCDN, nous pensons que seule la mise en place de tels CDN, réutilisant les modèles pair-à-pair permet de concilier les problématiques de monté en charge et de maîtrise des coûts des systèmes actuels, avec le maintient d'une expérience utilisateur. Toutefois, à l'inverse de coralCDN, nous pensons que ces CDN, ne sauraient atteindre une masse critique, sans la mise à contribution des utilisateurs eux mêmes.

Pour cela, nous proposons behaveCDN, un mécanisme original, de CDN participatif, réutilisant les ressources des utilisateurs pour permettre un partage de contenu en pair-à-pair, par les utilisateurs eux mêmes. Afin de rester compatible avec le modèle existant notre CDN fonctionne comme un comme un cache Web, respectant les politiques de cache définies par les sites Web.

Pour éviter les problèmes de "slow-start" bien connus des systèmes pair-à-pair, et à même d'entraver l'expérience utilisateur, nous proposons de réutiliser les modèles sociaux, pour construire un réseau où les utilisateurs seraient sélectionnés par leurs centres d'intérêt voisins, permettant ainsi une mise à disposition directe des contenus, sans nécessité de nouvelles connexions, ou d'effectuer une recherche dans l'ensemble du réseau.

Les contributions principales de ces travaux sont les suivantes :

- La création d'un système original de distribution de contenu pour le Web, utilisant les modèles pair-à-pair non structurés afin de reproduire des modèles sociaux, via l'utilisation de profils utilisateurs implicites construits au travers de analyse les données de navigation.

- La conception d'une implémentation de ce modèle, réutilisable comme plate-forme d'expérimentation pratique pour les algorithmes pair-à-pair, les problématiques de respect de la vie privée ou encore les systèmes de recommandation distribués.

- L'analyse, des problématiques ouvertes dans notre modèle et des pistes potentielles de résolution de ces problématiques, permettant d'envisager de futurs travaux.

Le reste de ce document est organisé comme ceci :
la première section revient sur quelques notions décrivant les CDN, les modèles pairs à pair, et les défis ouverts dans l'utilisation de modèle pair-à-pair. La deuxième section, décrit quelques un des pré-requis que notre système souhaite adresser. Les deux sections suivantes décrivent en détail le modèle utilisé et l'implémentation réalisée. Enfin, les deux dernières sections décrivent en détail les méthodes d'analyse de résultats et les problématiques encore ouvertes pour notre système de distribution de contenu en pair-à-pair.

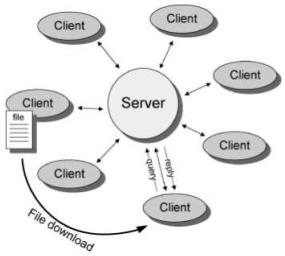


FIGURE 1 – Modèle de réseau hybride [16].

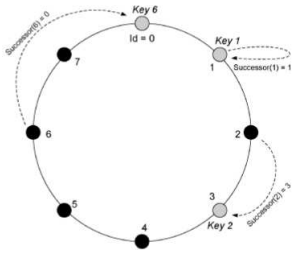


FIGURE 2 – Modèle de réseau structuré [64].

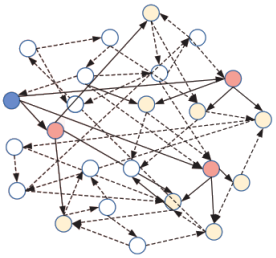


FIGURE 3 – Modèle de réseau non-structuré [41].

2 Contexte

2.1 Les technologies pair-à-pair.

Le pair-à-pair (ou P2P) est un modèle original de communication dans les réseaux numériques. Il est souvent défini comme une extension du modèle client-serveur, où tous les nœuds du réseau sont à la fois clients et serveurs. Il est aussi parfois défini comme un retour aux idées fondatrices d'Internet réseau de communication de bout en bout dans laquelle les notions de clients et serveurs ne sont apparues qu'a posteriori (à l'image de la toute première RFC [34] mentionnant des «hôtes » sans distinction entre d'éventuels clients et serveurs).

Les technologies pair-à-pair ont été utilisées dans de nombreux domaines, de la conception des premiers réseaux de services au dessus d'Internet (Usenet, IRC...), au routage Internet (BGP), en passant par le partage de fichiers (BitTorrent) ainsi que le calcul (BOINC) et le stockage distribué OnceanStore [62] . En 2009, près de 70% dû trafic Internet était du aux technologies pair-à-pair [87] (malgré qu'aux vu des dernières études celui-ci semble décroître au profit du streaming web) .

Ces modèles sont particulièrement intéressants de part la montée en charge qu'ils permettent : chaque nœud rejoignant le réseau viendra potentiellement augmenter la quantité de ressources disponibles. Il existe différents modèles de réseaux pair-à-pair. Ceux-ci définissent la manière dont le réseau est construit et organisé, ainsi que la manière dont les nœuds communiquent entre eux :

Les modèles centralisés, ou hybrides, (figure 1) regroupent les modèles où la construction du réseau et son maintien, reposent sur un ou plusieurs serveurs centraux, appelé parfois super-pairs ou super-nœuds. Ce sont ces serveurs qui indexent les ressources. Ces modèles permettent une gestion facilitée et donc moins coûteuse en ressources. Toutefois, ils se heurtent aux limites des systèmes centralisés classiques : faible tolérance aux fautes, difficultés de montée en charge. L'utilisation d'un grand nombre de serveurs centraux et leur désigna-

tion de manière plus ou moins automatique permet de palier ces inconvénients sans toutefois les corriger totalement. L'exemple du réseau Napster, premier réseau à populariser ce concept, fermé suite à la suppression de son serveur central, illustre bien ces problèmes de robustesse.

Les modèles structurés, illustrés par la figure 2, regroupent les modèles où le réseau et l'ensemble des nœuds qui le composent sont fortement structurés. Les nœuds, comme les ressources, sont identifiés et accessibles via un adressage global dont la gestion est partagée entre tous les nœuds. Ce type de réseau permet donc un passage à l'échelle facilité, et offre de bonnes performances dans pour les recherches et l'accès au ressources, au prix d'une plus grande complexité dans la gestion du réseau, et donc d'une augmentation des coûts liés à celle-ci. Parmi ces modèles ont trouve les réseaux Chord [91], Pastry [86] ou encore Kademelia [65]. Ils reposent sur une table de hachage distribuée ou DHT, indexant à la fois la contenus et les nœuds eux mêmes. L'utilisation d'une table de routage contenant des raccourcis vers les différentes extrémités du réseau permet un routage efficace : le pire cas en terme de nombre de sauts est de l'ordre de $O(log(n))$.

Les modèles non structurés regroupent les modèles de réseau pair-à-pair ne reposant pas sur une structure ferme, à l'image de l'exemple en figure 3. Les nœuds sont organisés de manière plus ou moins aléatoire et la diffusion de messages intervient par inondation (flooding) où chaque nœuds renvoi les messages reçus à ses voisins. La gestion de ces réseaux est réduite à son strict minimum et est par conséquent peu coûteuse. La diffusion des messages peut être décrite comme épidémique ou par rumeur (gossip [41]) et peut être exprimée de manière probabiliste fonction du nombre de nœuds et du nombre de fois qu'un message reçu par un nœuds est renvoyé (fanout). Ces réseaux offrent de bonnes propriétés : un fanout moyen de l'ordre de $O(log(n))$ (n étant la taille du réseau) permet de s'assurer de manière probabiliste qu'un message sera reçu par l'ensemble du réseau après $\frac{log(n)}{log(log(n))} + O(1)$ itérations [23].

Exemple : Le protocole BitTorrent est un protocole développé au début des années 2000 et couramment utilisé sur Internet. Celui-ci offre des fonctionnalités de partage de fichiers et repose sur une stratégie dite gagnant-gagnant [33, 77, 79] («tit for tat») et sur une diffusion multi-sources afin de permettre un téléchargement efficace.

Le protocole BitTorrent fut tout d'abord conçu comme un modèle hybride : si les données sont stockées et échangées directement entre les pairs, l'accès au réseau, tout comme l'indexation et la recherche de méta-données, reposent sur l'utilisation de super-pairs appelés ici trackers. Afin d'améliorer la robustesse du protocole et son aspect décentralisé, diverses extensions ont été proposées : L'utilisation d'une table de hachage distribuée basée sur le protocole Kademelia [65], indexant à la fois les nœuds et les méta-données, permet au pairs de s'échanger

directement des contenus en utilisant un modèle structuré. Une seconde modification repose sur un système d'échanges de pairs et un système permettant aux pairs de s'échanger les méta-données directement entre eux. Enfin, une troisième extension, Tribler [78], complète la seconde et permet d'utiliser un modèle par inondation où pairs transmettent les requêtes les uns des autres à l'intérieur du réseau. Ces différentes extensions ont permis d'augmenter la robustesse du protocole et d'en améliorer les performances [97].

Le protocole BitTorrent repose donc actuellement à la fois sur les modèles centralisés, structurés et non structurés [85].

2.2 Les Content Delivery Networks (CDN)

Les Content Delivery Networks ou CDN sont une réponse industrielle aux problématiques de diffusion de contenu sur Internet. Pour faire face aux problématiques de montée en charge et conserver une bonne expérience utilisateur, les contenus ne sont plus distribués par un unique serveur, mais par un ensemble de serveurs sur lesquels sont répartis les différents clients. Ceci afin de permettre un partage de la charge entre les différents serveurs et la conservation d'une bonne expérience utilisateur, tout en assurant une réduction des coûts en augmentant la localité du trafic réseau.

Techniquement, on pourrait définir les CDN comme un ensemble de serveurs offrant du cache de données, mise en place sous la forme d'un arbre : un ou des serveurs sources alimentent des serveurs intermédiaires, qui fournissent les contenus aux utilisateurs , [100]. Afin d'optimiser l'expérience utilisateur, les serveurs finaux sont le plus souvent placés au plus proche des clients. Le choix du serveur final auquel accédera l'utilisateur peut reposer sur une combinaison de diverses techniques, documentées dans une RFC [20] : DNS, redirections HTTP, anycast.

De nombreux fournisseurs industriels, tels qu'Akamai [1], LimeLight [9] ou Amazon CloudFront [2], proposent de tel services. Toutefois, le coût à ces réseaux [3] les rend difficiles d'accès, réduisant ainsi leur adoption à quelques gros sites ayant d'importants moyens financiers. Par ailleurs, cette approche, bien qu'efficace, conserve un aspect fortement centralisé, et reste sensible aux attaques, même si le grand nombre de serveurs utilisés et leur forte répartition géographique permettent de réduire les risques.

Le projet CoralCDN [47] est une initiative visant à développer un CDN en pair-à-pair. Celui-ci est constitué de serveurs de cache répartis sur plusieurs niveaux (local, régional, et global) fortement connectés et partageant une table de hachage distribuée à des fin d'indexation. Coral se veut compatible avec les modèles et infrastructures existantes et ne repose sur aucune modification des serveurs ou des clients. Pour «coraliser » une URL il suffit de rajouter le suffixe *nyud.net* à la fin du nom de domaine. La requête sera alors redirigée vers les

serveurs de Coral qui agiront comme un proxy se plaçant entre le client et le serveur. Des mécanismes permettent de choisir le serveur le plus proche via une redirection DNS ou HTTP [48]. Cette approche originale par sa simplicité permet l'utilisation du CDN aussi bien par les utilisateurs eux même que par les fournisseurs de contenu. Toutefois, elle ne permet pas aux utilisateurs de rejoindre le réseau et de contribuer à celui-ci, réduisant quelque peu l'intérêt de l'utilisation des concepts pair-à-pair.

Coral est déployé de manière ouverte sur Internet depuis 2004 (via le réseau Planetlab [32]). Dans un article de 2010 [46], Freedman et al. proposent un retour d'expérience basé sur ces quelques années de déploiement du réseau. Ils montrent comment celui-ci a permis de servir plusieurs millions de requêtes par mois, permettant d'endiguer les pics de charge et à aidant la diffusion d'informations à très large échelle (listes noires du plug-in firefox AdBlock). En se basant sur cette expérience, Freedman et al. reviennent sur quelques éléments de conception à prendre en compte dans le design d'un CDN et le déploiement d'un réseau à grande échelle.

L'expérience de Coral tend de plus à démontrer tout l'intérêt que peuvent avoir les concepts de pair-à-pair dans la conception de CDN : la nature et le volume de données traitées en font d'excellents candidats pour les techniques pair-à-pair. Dans une étude de 2005 [61], Karagiannis et al. démontrent également les intérêts que peuvent présenter les systèmes pair-à-pair et leur nature décentralisée pour les opérateurs.

On retrouve ainsi deux grandes approches dans la conception de CDN en pair-à-pair : les systèmes de distribution de contenu assistés par le pair-à-pair (et reposant partiellement sur des serveurs centraux) et les solutions pleinement pair-à-pair.

2.2.1 Les Content Delivery Networks assistés par le pair-à-pair

En 2006, Xu et al. proposent une étude [99] démontrant les intérêts de solutions utilisant à la fois les modèles pair-à-pair et les CDN pour de la diffusion de flux multimédia. Cette étude propose un fonctionnement par palier : les premiers clients cherchant à utiliser le système sont servis par le CDN, tandis que les clients suivants sont servis par le réseau pair-à-pair constitué par les clients ayant rejoint le réseau précédemment. Ceux-ci étudient plus particulièrement les seuils possibles des paliers de passage du CDN au pair-à-pair, et comment ceux-ci peuvent être optimisés afin de réduire au mieux les coûts (en utiliser en maximum le pair-à-pair) tout en maintenant une bonne qualité d'expérience pour l'utilisateur.

Les systèmes LiveSky [100],et PcubeCast [55] sont construits sur ce principe. Ils proposent des retours d'expérience basés sur le déploiement de tels modèles. PcubeCast montre ainsi comment sur un flux d'environ 62Gb/s diffusé auprès de 85000 clients, près de 52Gb/s a pu être servi en pair-à-pair. LiveSky propose

de plus un système de palier modifié afin de permettre une transition plus douce entre le CDN et le réseau pair-à-pair. Cette solution offre également de bonnes performances et évite sans doute le clustering que pourrait causer des paliers trop fermes.

Cette approche de CDN assisté par le pair-à-pair, permet de combiner les avantages des deux modèles, les CDN de par la qualité de service qu'ils offrent permettent un démarrage rapide, une reprise rapide en cas d'erreur, et peuvent jouer le rôle de super-pair organisant les clients au travers d'un réseau structuré, prenant en compte la localité réseau. Les technologies pair-à-pair viennent en complément et permettre de décharger le CDN en cas de problème de montée en charge ou simplement pour réduire les coûts.

Le protocole BitTorrent permet l'utilisation de telles technologies au travers d'une extension [68] permettant le téléchargement de fichiers en parallèle en pair-à-pair et via des systèmes de diffusion classiques. Cette technologie est notamment utilisé par Blizzard pour diffuser les mises à jours de ses jeux vidéos ou pour la mise à disposition de distributions Linux.

Depuis 2010, une expérience est conduite sur Wikipedia [19] afin de permettre la diffusion des contenus vidéos, assistée par les mécanismes pair-à-pair. Celle-ci repose sur le protocole BitTorrent et plus particulièrement sur Tribler [78].

2.2.2 Les Content Delivery Networks entièrement pair-à-pair

FlowerCDN [40] est une seconde approche orientée vers la conception d'un cache Web global distribué sur un réseau pair-à-pair. Celle-ci reprend donc en partie le modèle du réseau CoralCDN [46, 47], mais permet potentiellement d'intégrer les clients eux même au réseau de diffusion pair-à-pair ainsi construit. FlowerCDN repose sur une structure originale appelée D-ring et utilisant une table de hachage distribuée afin d'indexer des réseaux prenant en compte les intérêts des utilisateurs (par site Web) et leur localité.

Des simulation tendent à montrer les bons résultats de FlowerCDN vis à vis à des modèles concurrents comme Squirrel [58]. Toutefois, si l'utilisation d'une table de hachage distribuée pour indexer des «mini-réseaux » par intérêt et par localité peut sembler intéressante, celle-ci peut présenter quelques inconvénients. Ainsi, l'utilisation d'un grand nombre de réseaux d'intérêts pour chaque couple $site, localité$ suggère un fort clustering, inhérent à l'utilisation de la localité (voir paragraphe 2.3.2) couplé à une granularité potentiellement trop fine dans la prise en compte des intérêts. Un tel niveau de granularité ne permet par exemple pas de profiter des liens pouvant exister entre des sites traitant des mêmes sujets. Enfin, cette solution ne semble pas pleinement prendre en compte quelques-un des défis inhérents à la conception de systèmes de diffusion de contenu en pair-à-pair.

2.3 Défis par la diffusion de contenu en pair-à-pair

La conception de systèmes de distribution à large échelle, basée sur des modèles pairs-à-pairs potentiellement décentralisés ouvrent de nouveaux problèmes. Dans une étude de 2006 [83], Rodriguez et al. en décrivent quelques-uns. L'expérience de CoralCDN [46] vient compléter cette liste par une vue opérationnelle basée sur un retour d'utilisation. Nous décrirons ici quelques unes de ces problématiques.

2.3.1 Le NAT

Le Nat ou Network Address Translation [90], est un système couramment utilisé sur Internet. Celui-ci permet de ré-écrire les adresses sources et destinations d'un message et permet donc de partager une adresse IP entre plusieurs machines. Toutefois celui-ci casse le modèle de communication de bout en bout d'Internet et peut poser des problèmes dans les systèmes pair-à-pair, ceux-ci reposant principalement sur des connexions de bout en bout entre les pairs, réduisant de fait les performances du système [101]

Pour pallier ces problèmes divers solutions de contournement existent :

- Le protocole IPv6 se veut une évolution du protocole IPv4 actuellement utilisé sur Internet. De part le grand nombre d'adresses qu'il propose (2^{128} au lieu de 2^{32} pour IPv4), celui-ci permettrait de réduire la nécessité des mécanismes de NAT et de restaurer le principe de communication de bout en bout. En 2012, le déploiement d'IPv6 n'est pas encore totalement effectif et seule une minorité des nœuds ont un accès IPv6 [10,56]. Toutefois, des efforts de déploiements sont en cours, à l'image du World IPv6 Launch [13] prévu le 6/6/2012 qui vise à déployer IPv6 de manière synchronisée auprès de plusieurs centaines d'opérateurs, fournisseurs de contenus comme fournisseurs d'accès.
- Des mécanismes intégrés aux systèmes de NAT permettant également de restaurer le modèle de communication de bout en bout, c'est le cas d'uPnP [12] qui permet aux nœuds situés derrière un NAT d'autoriser de possibles connexions entrantes.
- Des mécanismes permettent également de contourner les systèmes de NAT via des serveurs intermédiaires [84] ou via la création ou la réutilisation de connexions existantes dans le but de permettre une communication entre deux pairs. [44, 69, 89].

2.3.2 La prise en compte de la topologie réseau

La prise en compte de la topologie réseau est un élément important dans la conception de systèmes pair-à-pair. Celle-ci peut regrouper plusieurs choses : la prise en compte des NAT, et le choix des pairs de manière à optimiser l'efficacité du protocole et l'utilisation des ressources réseaux.

La prise en compte de la capacité des nœuds en complément ou à la place de la localité réseau peut également être un élément intéressant dans la conception de systèmes pair-à-pair. Celle-ci peut permettre d'organiser le réseau en conséquence afin d'augmenter les performances globales. Jin et al., dans une étude [60] de 2011 discutent de l'intérêt de la prise en compte de cette capacité réseau vis-à-vis des informations de localité. Ils concluent que l'utilisation excessive des informations de localité dans la sélection des pairs tend à augmenter les clustering et diminuer globalement les performances. A l'inverse, l'utilisation de la capacité permet une augmentation significative des performances sans augmenter le clustering.

Diverses recherches montrent comment il est possible d'estimer au mieux la capacité des nœuds. C'est le cas de [92] qui propose un modèle basé sur une corrélation entre des mesures obtenues via les ressources échangées et l'injection régulière de trafic afin d'obtenir une estimation de la bande disponible. Heterogeneous gossip [49] montre comment l'utilisation d'une telle information permet d'augmenter de manière significative les performances dans les réseaux épidémiques (bien que le concept semble également applicable aux réseaux structurés).

La prise en compte de la localité est également intéressante dans la conception de systèmes pair-à-pair. Celle-ci, permet d'augmenter l'efficacité des protocoles par en privilégiant des nœuds plus proches et donc potentiellement plus rapides. De plus, celle-ci permet une utilisation plus raisonnée des ressources d'un opérateur en augmentant la localité du trafic à l'intérieur de son réseau et vers les réseaux proches. Ce mécanisme visant à augmenter la localité du trafic, très faible (moins de 5%) par défaut [95], permet entre autre d'éviter aux opérateurs d'avoir recours à des mécanismes de blocage ou de limitation du pair-à-pair tels que ceux déployés par Comcast en 2007 dans le but de limiter ses coûts [39]. Cette problématique étant particulièrement importante tant du point de vue des utilisateurs que des opérateurs, de nombreuses solutions ont été proposées. Le groupe de travail Alto [4] de l'IETF et de l'IRTF étudie actuellement la normalisation de ces travaux et en résume quelques uns dans une RFC [82] :

- L'utilisation d'oracles telle que proposée par les systèmes Idmaps [45], GNP [72]et Ono [31] est une première approche permettant de mesurer la localité entre deux nœuds via l'utilisation d'un tiers ayant une connaissance plus globale du réseau. Ono [31] est un exemple d'une telle solution. Celle-ci fonctionne par triangulation : afin de vérifier la proximité de deux pairs, ceux-ci comparent les résultats des redirections effectués par des CDN : le fait que deux pairs soient redirigés au même endroit ou non donne une bonne heuristique de leur relative proximité. De telles approches offrent de bons résultats ; ainsi Ono permet amélioration de l'ordre de 30% de la localité du trafic et des performances.

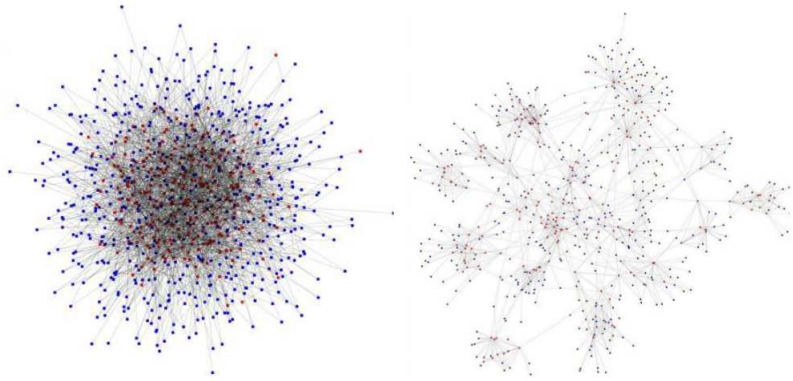
- Une coopération P2P et opérateur : D'autres modèles étendent le principe de l'oracle et proposent une coopération entre systèmes pair-à-pair et opérateurs, dans un modèle gagnant gagnant permettant aux opérateurs d'augmenter la localité du trafic et aux utilisateurs d'obtenir de meilleurs performances. C'est le cas de P4P [98] ou de [14]. Une expérimentation à grande échelle de P4P, initiée par l'opérateur Comcast et documentée dans une RFC [51] montre de bons résultats tant du point de vue des utilisateurs (augmentation du débit allant jusqu'à 80% dans certains cas) que de l'opérateur (diminution du trafic entrant et sortant respectivement de l'ordre de 80% et 30%).
  Piatek et al. [76] démontrent toutefois que cette approche peut être limitée de par les conflits d'intérêts des opérateurs entres eux. L'idée, déjà développée par Karagiannis et al. [61], est que certains opérateurs ayant le rôle d'intermédiaire entre fournisseurs de contenus et fournisseurs d'accès (opérateurs dits«de transit ») ont tout à gagner à maintenir une faible localité du trafic afin de pouvoir facturer les échanges entre opérateurs et sont donc peu incités à participer à une telle coopération.

- Enfin, il existe des modèles plus décentralisés : Vivaldi [35] est une première approche, développée pour le système Chord [91]. Celle-ci crée une cartographie locale du réseau, basée sur la latence, cette cartographie étant ensuite échangée lors des communications entre eux les nœuds afin de permettre à ceux-ci d'obtenir une vision plus globale du système. TopBt [81] est une autre approche reposant sur une vue locale du système (potentiellement assistée d'informations globales telles que les tables routages) afin de sélectionner les pairs les plus proches en fonction de différentes métriques : latence, nombre de sauts, informations de routage. Cette solution offre de bon résultats en augmentant la localité du trafic, et les performances en téléchargement de l'ordre de 15 à 20%. L'utilisation d'une vue locale est également particulièrement intéressante car elle permet aux nœuds de profiter de bonnes performances quand bien même leurs pairs ne bénéficieraient pas de la technologie (à l'inverse d'Ono [31] par exemple).

Ces différents modèles offrent donc de bonnes résultats de part les gains de performances et l'augmentation de la localité du trafic qu'ils permettent. Toutefois, la prise en compte des informations de localité n'est pas sans inconvénients. Celle-ci peut augmenter le clustering et réduire la robustesse du réseau face aux défaillances [63]. Aussi, si l'on cherche à augmenter les performances des nœuds, peut être est il plus intéressant d'essayer de mesurer directement les capacités de ceux-ci, plutôt que d'utiliser la localité comme heuristique.

2.3.3 Le respect de la vie privée

La protection des informations sensibles des utilisateurs est une problématique croissante sur Internet. Cette problématique peut sembler simple à



(a) Topologie sans oracle

(b) Topologie biaisée par l'introduction d'un oracle

FIGURE 4 – Topologie réseau modifiée par l'introduction d'un oracle [14].

adresser dans les systèmes centralisés, toutefois l'expérience montre que l'utilisation de telles informations, leur fuite ou leur revente à des fins commerciales est devenue courante, à l'image du moteur de recherche Google [5, 6].

La plupart des systèmes pair-à-pair actuellement utilisés sont tout aussi peu respectueux de la vie privée : la diffusion des informations de présence et les contenus partagés sont accessibles publiquement, permettant ainsi de tracer les utilisateurs de ces réseaux. Divers études s'intéressent à cette problématique de montrent comment il est facile de récupérer des informations sur les utilisateurs de BitTorrent, que ce soit au travers de trackers, comme [75] ou par l'utilisation de la DHT [42, 96].

A l'inverse, il existe des systèmes pair-à-pair fortement orientés vers le respect de la vie privée c'est le cas de Tor [38] ou de freenet. Tor repose sur un routage en oignon : afin de communiquer au sein du réseau un nœud construit un circuit à partir de la liste des autres nœuds du réseau. Chaque nœud intermédiaire n'aura de vision que sur le prédécesseur et le successeur d'un message, sans pour autant pouvoir en identifier l'origine ou la destination, de même que les autres nœuds intermédiaires. L'utilisation d'un chiffrement en couche appliqué à chaque saut dans le réseau permet de s'assurer de la confidentialité des données à l'intérieur de celui-ci.

Une des limite principale d'une telle approche concerne les performances de ce type de réseau. Certains choix de conception pouvant dégrader très fortement ces performances : l'utilisation de chiffrement à chaque saut, l'absence de mécanismes incitatifs dans la participation au réseau, l'utilisation de nœuds sans prise en compte de la localité, ou encore le choix d'en-capsuler du TCP et donc de ne permettre qu'une diffusion mono-chemin sont quelques un des choix ré-

duisant l'intérêt pratique de Tor lorsque l'expérience utilisateur est importante.

OneSwarm [57] se veut une autre tentative de développement d'un système pair-à-pair, prenant en compte les problématiques de vie privée tout en maintenant un bon niveau de performances. Celui-ci repose sur une série de mécanismes originaux, proposés à partir de l'analyse des problèmes des solutions précédentes : l'utilisation de chiffrement de bout en bout et non à chaque saut, l'utilisation de mécanismes de diffusion multi-chemin ou encore l'utilisation de mécanismes incitatifs basés sur ceux proposés par BitTorrent. D'autres mécanismes originaux sont proposés afin de préserver la vie privée des utilisateurs. Ainsi, la recherche et la diffusion de contenu s'effectue par inondation en utilisant un routage en oignon. Un jeu astucieux sur les délais permet de ne pas mettre de TTL dans les messages. Pour se préserver de certaines attaques, comme les attaques par analyse de délais [70] ou par collusion, la diffusion des messages est rendue probabiliste et des délais artificiels sont ajoutés. Enfin, si une DHT est une utilisée pour le bootstrap, chaque entrée y est chiffrée afin de n'être lisible que par un seul et unique pair : un nœud voulant établir des connexions avec n pairs devra insérer n entrées.

Une expérimentation sur le réseau Planetlab tend à comparer les performances de celui-ci vis à vis de Tor et montre de nets gains , en particulier lors de l'utilisation de la diffusion multi-chemin. Toutefois, cette solution étant basée sur des compromis entre la sécurité et la performance, le niveau de sécurité fourni est moindre que celui proposé par des solutions comme Tor, plus orientées vers le respect de la vie privée et la sécurité.

2.3.4 Un réseau «social»

Si l'on s'intéresse à la conception d'un réseau pair-à-pair orienté vers la distribution de contenu et performance, l'utilisation d'un réseau basé sur les concepts sociaux peut également être intéressante :

- L'utilisation d'une dimension sociale peut ainsi permettre la prise en compte des intérêts afin de sélectionner les pairs ayant le plus d'intérêts communs et donc potentiellement plus de contenu à échanger. Ceci permettant de conduire à de potentielles améliorations de performances.

- Aussi, l'exemple de OneSwarm [57] tend à montrer que l'utilisation de données issues de réseaux sociaux permet un bootstrap plus rapide du réseau, permettant d'atteindre plus rapidement une masse critique autant en terme d'utilisateurs que de liens entre ceux-ci.

- De plus, l'utilisation de concepts sociaux permet de mettre en avant d'éventuels liens de confiances entre deux pairs, permettant ainsi de réduire certains mécanismes visant à préserver la vie privée, à l'image de OneSwarm [57].

- Enfin, la construction d'un réseau «social » permet de revenir aux bonnes

propriétés des réseaux dit small-world, ainsi, les exemples de Facebook [94]et de Last.fm [57] montrent en moyenne 4,3 et 7.1 sauts entre les utilisateurs du réseau.

Gossple [49] est un exemple d'un tel réseau, construit autour d'algorithmes épidémiques, et réutilisant les concepts de routage en oignon. Celui-ci vise à construire un réseau, basé sur les intérêts de ses utilisateurs tout en préservant les contraintes de respect de la vie privée.

2.3.5 L'authentification des contenus

Pour être pleinement utilisable un système de diffusion de contenu en pair-à-pair devrai conserver les fonctionnalités offertes par le modèle client serveur existant. L'authentification des contenus au travers de mécanismes comme HTTPS est un de ces mécanismes, celui-ci si repose sur une infrastructure à clef publique (ou PKI) centralisé.

- En se basant sur l'expérience de CoralCDN [46], Terrace et al. proposent une approche [93] compatible avec les infrastructure actuellement utilisée sur Internet, afin d'authentifier les données. Celle-ci repose sur une vérification de l'authenticité des données via une consultation du serveur d'origine. Toutefois, cette approche, bien que simple et efficace reste centralisée exerçant une dépendance non négligeable vers le serveur d'origine.

- A l'inverse Datta et al. proposent un modèle [37] basé sur une PKI pair pair.

- Enfin, une dernière approche, consisterait à utiliser les fonctionnalités classiques des réseaux de confiance (Web of Trust) basés sur les concepts de réputation afin d'authentifier la source des contenus diffusés sur le réseau. Cette approche est décrite en section 7.3.

2.3.6 Des mécanismes d'incitation adaptés

Si le succès de BitTorrent vis à vis des autres modèles de pair-à-pair tend à démontrer une chose, c'est bien l'importance et l'intérêt que peuvent avoir l'utilisation de mécanismes incitatifs afin de favoriser les échanges dans le réseau. BitTorrent repose sur un principe simple [33, 79], celui d'une approche dite «gagnant-gagnant », visant à privilégier les échanges équitables et à rendre profitable la contribution au réseau : ainsi le comportement le plus profitable devient le comportement le plus altruiste. Par exemple un nœud BitTorrent cherchera à récupérer en priorité les pièces les plus rares, s'assurant ainsi de leur disponibilité sur le réseau.

Toutefois, cette approche, bien qu'éfficace n'est, de par son coté générique, pas forcément adaptée à tous les cas d'usages, et reste perfectible [74]. D'autres

solutions sont ainsi proposées :

- Aperjis et al. proposent une solution [17] basée sur les notions économiques de prix, d'offre et de demande. Celle-ci repose sur des mécanismes incitatifs explicites (le «prix » d'un contenu est annoncé). Elle vise à augmenter l'aspect multilatéral des échanges et permet par exemple de prendre en compte les cas des nœuds contribuant beaucoup ou très peu au réseau ainsi que la possible prise en compte des contraintes réseaux sous-jacentes.

- 2FAST [50] est une autre solution visant à favoriser les échanges dans le cadre d'un réseau basé sur les principes sociaux comme Tribler [78]. Celle-ci repose sur l'idée d'utiliser les amis d'un pair pour l'aider dans ses téléchargement, et se veut donc une extension du concept altruiste développé par BitTorrent.

- OneSwarm [57], utilise également le concept de «tit for tat » de BitTorrent. Toutefois, afin d'inciter les nœuds à participer au routage en oignon celui-ci tend à privilégier les nœuds ayant un ratio *upload/download* proche de 1 ou supérieur. En effet, la transmission d'un message ayant un impact nul sur ce ratio, un nœud ayant un ratio proche de 1 ou supérieur sera un nœud ayant fortement contribué au réseau, que ce soit par le routage ou par la diffusion de contenu.

Loin d'être un index exhaustif de toutes les problématiques à prendre en compte dans la conception de systèmes de diffusion de contenu en pair-à-pair, cette étude, espère toutefois en avoir cerné les principales, permettant ainsi de guider de futurs travaux.

Les contraintes de montée en charge et de robustesse du modèle actuel de diffusion de l'information sur Internet représentent une grande opportunité pour les systèmes pair-à-pair. Si ceux peuvent être vus comme un moyen de résoudre ces problématiques, ils permettent également d'envisager des solutions plus ouvertes, plus décentralisées, mais aussi plus participatives et par conséquent plus proches des utilisateurs et de leurs préoccupations. C'est, afin de prendre au mieux en compte ces préoccupation que nous choisissons de fonder notre modèle sur une série de pré-requis.

3 Pré-requis

Afin de concevoir un modèle pertinent de distribution de contenu web en P2P, plusieurs éléments nous semblaient importants à adresser : la compatibilité avec l'infrastructure existante, la conception d'un système auto-adaptatif, le maintient d'une bonne expérience utilisateur, ou encore le respect de la vie privée.

3.1 Compatibilité avec l'infrastructure existante

L'une des principales problématiques que rencontre le déploiement d'un système pair-à-pair est l'obtention d'une masse critique. En effet, la loi de Metcalfe définit l'intérêt d'un réseau comme le carré du nombre d'utilisateurs [67]. Concrètement cela signifie qu'un réseau ayant un faible nombre d'utilisateurs sera moins performant, offrira moins de contenus et aura des difficultés à obtenir une masse critique tant que cet état de fait perdurera.

Nous avons choisi de prendre en compte cette problématique des la conception du système, en permettant à celui ci de bénéficier d'une compatibilité avec l'infrastructure existante. Ainsi, quand bien même une masse critique d'utilisateurs ne serai pas encore atteinte, la quantité de contenu comme les performances du système ne se trouveraient que très peu dégradée. Ceci afin de permettre au système d'attirer suffisamment d'utilisateur pour obtenir une masse critique et envisager ainsi un déploiement incrémental [1].

3.2 Un système auto-adaptatif

Une seconde problématique nous semblait importante à adresser, celle de concevoir un système adaptatif, dans lequel le réseau se réorganiserai de lui même. Ceci afin de permettre une utilisation optimale des ressources, conduisant entre autre à de bonnes propriétés de montée en charge mais aussi à un déploiement aisé nécessitant le moins d'intervention humaine possible, à l'inverse par exemple des CDN pour les fournisseurs de contenus ou de fermes de proxy pour les fournisseurs d'accès.

3.3 Expérience utilisateur

Une troisième problématique importante dans la conception d'un système pair-à-pair est le maintient d'une bonne expérience utilisateur. Nous avons choisi de placer celle-ci au cœur du design de notre modèle, afin de proposer une expérience utilisateur au moins aussi bonne que celle proposée par les modèles existants. En effet, nombreux sont les systèmes de distribution de contenu en pair-à-pair, toutefois, ceux-ci, à l'image de BitTorrent, privilégient le débit brut à la réactivité. Ce problème, connu sous le nom de "slow-start" nous semble un élement primordial à adresser dans le but de préserver l'expérience utilisateur, dans le cadre de la distribution de contenu web. Une étude menée par Google en 2009 et pointant l'impact sur le nombre de clients des délais de recherche [28] l'illustre bien.

1. Le succès du réseau Tor, reposant sur un modèle ouvert et totalisant près de 400000 utilisateurs [11] vis à vis du réseau freenet reposant sur un modèle fermé et ne totalisant que 20000 utilisateurs [7], nous conforte dans ce choix.

3.4 Vie privée

Enfin, une dernière problématique que nous souhaitions prendre en compte est la problématique du respect de la vie privée. En effet, bien que des solutions existent, la plupart des systèmes pair-à-pair ne prennent pas ou peu en compte cette problématique et reposent sur la mise à disposition publique d'informations tels que les contenus échangés, ou encore l'adresse réseau des utilisateurs, permettant ainsi de dresser des profiles complets des utilisateurs [22,71,80]. Nous avons donc choisi de définir un modèle prenant en compte ces problématiques de vie privée en réduisant les informations publiquement disponibles et préserver un niveau de confidentialité au moins équivalent au modèle client serveur actuel.

4 Éléments de conception

4.1 Présentation générale

La compatibilité avec le modèle existant de diffusion de contenu actuellement utilisé sur internet étant au cœur de nos problématiques, nous avons choisi, à l'image du système coralCDN [47], de définir un modèle reposant sur le moins de modifications possibles de ce système. Toutefois, à l'inverse de ce projet, nous proposons de permettre aux clients eux même de contribuer au réseau. En effet, il nous est apparu que seul cette solution permettait de concevoir un système réellement adaptatif. Ainsi plus une ressource serai demandée plus celle ci se retrouverai facilement disponible dans le réseau. Ceci permettant d'obtenir de bonnes propriétés de montée en charge, tout comme une utilisation optimale du réseau.

4.1.1 Un cache web distribué...

Pour cela, nous proposons une modification des nœuds terminaux que sont les client web, afin d'utiliser un système de cache partagé en pair-à-pair. Un tel système permet aux clients de mettre à disposition les contenus web déjà visités. Cette approche, déjà étudiée par les projets SquirrelCache [58] et FlowerCDN [40], nous semblait une manière élégante de répondre aux deux problématiques que sont la réalisation d'un système à la fois rétro-compatible et auto-adaptatif.

En outre, une tel approche permet de d'envisager une intégration aisée aux solutions déjà existantes et un déploiement incrémental sous la forme d'un système de diffusion assisté par le pair-à-pair. Elle permet également d'envisager à terme le déploiement d'un système pleinement pair-à-pair via la mise en place de nœuds "super-pairs" déployés par les fournisseurs de contenus ou les fournisseurs d'accès.

Si le fait de ne proposer qu'un système de diffusion de contenu assisté pair-à-pair en obligeant par exemple, les clients à re-valider les contenus via le serveur

d'origine, peut être vu comme une faiblesse de notre modèle, nous pensons à l'inverse qu'il s'agit s'agit de l'approche la plus élégante pour permettre une compatibilité avec l'infrastructure existante. De plus, cette approche nous permet de préserver la qualité d'expérience de l'utilisateur, en autorisant notre système à consulter le serveur d'origine lorsqu'un contenu n'est pas directement disponible, évitant ainsi une recherche coûteuse dans l'ensemble du réseau, et la consultation depuis un pair potentiellement lointain. Ainsi, nous choisissons de privilégier cette expérience utilisateur au détriment de l'utilisation de notre modèle dans les cas où l'utilisation de celui-ci ne serait pas avantageuse.

Nous pensons qu'une telle approche, visant dans tous les cas à privilégier l'expérience utilisateur, est de nature à encourager le déploiement de notre modèle, et de permettre ainsi à celui-ci de montrer pleinement son potentiel.

4.1.2 ... utilisant un réseau pair-pair "social"

Afin d'obtenir un système auto-adaptatif, nous nous proposons de réutiliser le réseau Gossple, basé sur les algorithmes pair-à-pair non structurés décrit en section 2.1. Ces algorithmes, prennent en effet en compte cet aspect auto-adaptatif par design, en permettant aux nœuds de se réorganiser régulièrement pour sélectionner les meilleurs voisins, évitant ainsi les problèmes liés aux churn, tout comme la nécessité de dépendre d'une quelconque infrastructure.

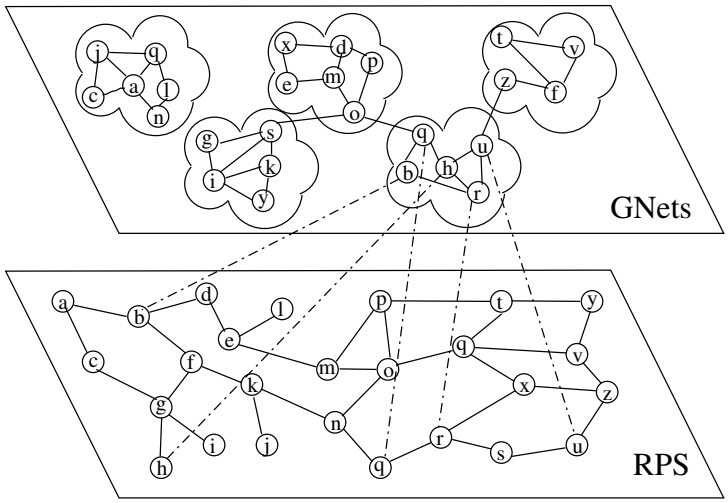
De plus, l'utilisation du réseau Gossple, permet de prendre en compte les modèles sociaux en permettant de sélectionner les voisins sur la base d'une proximité de centre d'intérêt. Notre intuition est qu'un ensemble de pairs partageant une base d'intérêts communs ont plus de chance d'avoir de contenus susceptibles d'intéresser les uns les autres, permettant une consultation directe de ces contenus via l'interrogation des nœuds voisins, sans nécessité de mécanismes de recherche pouvant conduire à un parcours de l'ensemble du réseau et donc à une latence accrue.

L'utilisation de modèles sociaux basé sur des profils utilisateurs nous semble également un bon moyen de tirer profit de l'éventuelle proximité de sujet entre les sites visités. Ainsi, si deux utilisateurs ont visité les même sites, traitant d'un même sujet donné, il parait judicieux que ceux-ci sélectionnent leurs voisins sur la base de ce centre d'intérêt commun, plutôt que sur la base des seuls sites visités.

Cette approche, visant à sélectionner les voisins sur une base d'intérêts communs, et permettant de tirer pleinement profit de l'éventuelle proximité entre les sites visités, nous semble idéale, dans le but d'augmenter les performances de notre modèle, vis à vis des systèmes existants comme FlowerCDN [40] ou Squirrel [58].

Pour compléter celle-ci, nous proposons de prendre en compte la topologie

FIGURE 5 – Le réseau Gossple [22].



réseau sous-jacente afin que les voisins d'un nœud soit également sélectionnes par leur proximité réseau, ceci afin de permettre des temps d'accès courts dans le but de maintenir une expérience utilisateur acceptable.

4.2 Le réseau Gossple

Nous avons choisi de réutiliser les travaux ayant eu lieu sur le réseau Gossple [22], décrit en figure 5. Comme évoqué lors de la section précédente, celui-ci permet la mise en place d'un réseau auto-adaptatif, basé sur les concepts sociaux, dans lequel les nœuds sélectionnent leurs pairs par proximité d'intérêt, via un système d'échange de profils. Pour ce faire, Gossple repose sur les éléments suivants :

- Un réseau contenant un ensemble de pairs aléatoires
- Un réseau contenant un ensemble de pairs choisi par intérêts
- Un modèle de calcul de similarité

4.2.1 Un réseau aléatoire

Le réseau aléatoire, ou RPS pour Random Peer Sampling [59], permet à chaque nœud d'obtenir un ensemble de nœud aléatoire. Celui-ci repose sur une procédure d'échange de vue : régulièrement, chaque nœud du réseau va envoyer à ses voisins une partie des nœuds auquel il est connecté : sa vue. Au cours de cet échange chaque nœud du réseau conservera un sous-ensemble de sa vue

originale, mais obtiendra également de nouveaux nœuds. Ce procédé permet à chaque nœud de renouveler régulièrement sa liste de voisins.

4.2.2 Un réseau par intérêt

Au dessus du réseau aléatoire, est construit est réseau d'utilisateurs partageant des intérêts communs. Ce *Gnet* [22], récupère les nœuds du RPS et imite son fonctionnement : régulièrement, chaque nœud du réseau échange avec ses voisins une partie de sa vue et en conserve un sous-ensemble.

Toutefois, à l'inverse du RPS, les pairs ne sont pas gardés aléatoirement, mais de manière à ne conserver que des nœuds partageant des intérêts communs. Ce mécanisme repose sur deux éléments principaux :

- Un système d'échange de profil, disponible dans le réseau Gossple. Nous réutilisons celui-ci afin d'échanger des profils implicites, basés sur des données obtenues durant le processus de navigation des utilisateurs, afin de refléter leurs habitude de navigation et les données associées. Le tout, dans le but et d'effectuer un clustering reflétant aussi bien les sites visités que les sujets traités par ces sites. Le détail des profils créés et les données utilisées pour le peupler seront décrites en section 4.3.1.

- Un modèle de calcul de similarité permettant de calculer la proximité des intérêts des utilisateurs. Celui-ci sera décrit dans la section suivante.

4.2.3 Un modèle de calcul de similarité

Afin de pouvoir calculer la proximité d'intérêts entre deux utilisateurs donnés, à partir de leur profils divers méthodes sont disponibles :

Le modèle de Jaccard est utilisé pour obtenir la part d'intérêts communs entre deux entités. Il est défini ainsi :

$$Jaccard(A, B) = \frac{A \cap B}{A \cup B}$$

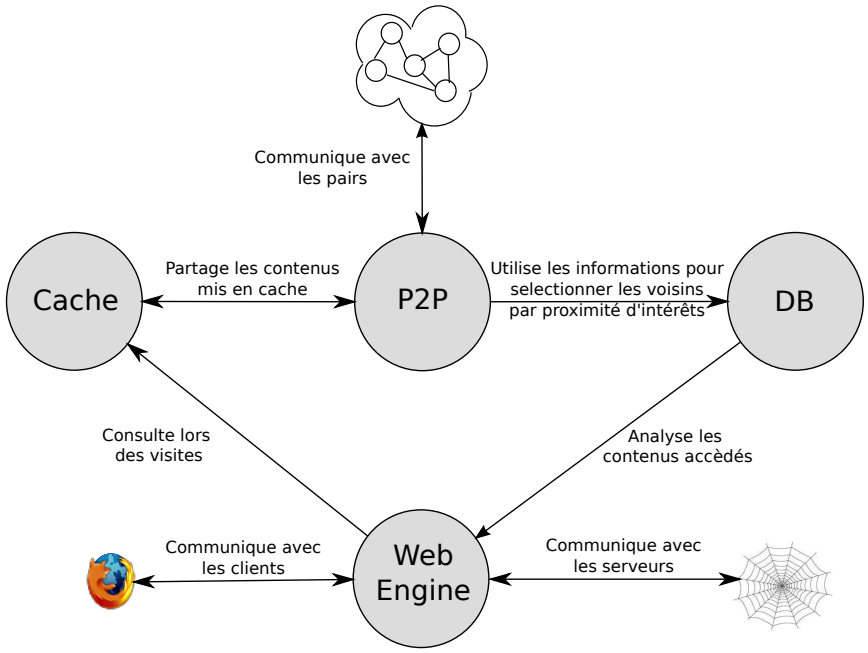
La similarité cossine est utilisée pour obtenir l'angle commun entre deux vecteurs. Celle-ci est définie de la manière suivante :

$$cossine(A, B) = \arccos\left(\frac{A.B}{\|A\|.\|B\|}\right)$$

4.3 Notre modèle

Notre modèle réutilise le réseau Gossple afin de constituer un réseau social par proximité d'intérêt entre les utilisateurs. Afin de réutiliser Gossple pour notre modèle de diffusion de contenu web en pair-à-pair les éléments suivants

FIGURE 6 – Architecture du système de cache pair-à-pair.



sont adaptés :

- La gestion des profils utilisateurs
- Le modèle de calcul de similarité

Au dessus du réseau ainsi constitué, se trouve notre système de cache web en pair-à-pair, permettant aux utilisateurs de s'échanger les contenus déjà visités. Celui-ci est construit au travers de plusieurs éléments, décrits en figure 6 :

- Un cache web local
- Un cache web pair-à-pair
- Un moteur web

4.3.1 Profils utilisateurs

Afin de créer un réseau social basé sur les habitudes de navigation des utilisateurs et leurs centres d'intérêts, il convenait de créer des profils utilisateurs susceptibles de refléter ces habitudes. Ceci, sans pour autant dépendre de calculs coûteux susceptibles de ralentir le processus de navigation. C'est ainsi que, tout

le long du processus de navigation une base de données des centre d'intérêts des utilisateurs est constituée, de manière implicite, à partir des informations de navigation des utilisateurs. Pour cela, les contenus demandés et accèdés sont analysés en tache de fond durant la navigation.

Sont ainsi récupérés :

- Des informations sur les contenus accèdés :
  - Les URL demandés
  - Les noms de domaines accèdés
- Des informations extraites des contenus eux-mêmes :
  - Les mots clefs des pages
  - Les descriptions des pages
  - Les titres des pages

Bien que perfectibles, ces heuristiques nous permettent de construire aisément un profil implicite des centres d'intérêts des utilisateurs, tout en évitant la complexité de la conception d'un réel moteur d'analyse sémantique des contenus accèdés.

L'utilisation de ces différents éléments permet, de plus, d'avoir accès à un ensemble d'informations ayant des dégréés de précisions différents. Ainsi, l'URL d'une ressource permettra de connaître de manière très précise les contenus accèdés. À l'inverse, les noms de domaines des sites visités, ou les informations sémantiques extraites des pages nous permettent de dresser un profil plus large des centres d'intérêts des utilisateurs. De cette manière, un profil utilisateur pourra contenir des éléments permettant de cibler précisément ces centres d'intérêts principaux, tout comme des éléments plus large tels que des mots clefs reflétant directement les domaines d'intérêts des utilisateurs ("research", "computer science", "peer-to-peer").

4.3.2 Calcul de similarité

Afin de calculer la similarité d'intérêt entre de profils de navigation ainsi constitué, notre système, utilise la méthode de Jaccard. Toutefois, la réutilisation des autres modèles de calcul (ou d'une combinaison de ceux-ci), est envisagée. Celle ci pourrait faire l'objet de futurs travaux, tout comme et l'évaluation de leur impact sur le réseau générée et les performances du système de cache distribué.

De la même manière, ces calculs sont effectués sur un sous ensemble du profil, les 1000 éléments les plus récurrents, sans tenir compte de leur nombre d'occurrences, ni de leur type (URL, domaine, titre de page...). Des modèles plus évolués utilisant le nombre d'occurrences comme pondération, et/ou un poids différent selon le type d'élément seront analysés prochainement.

En effet, et comme nous l'avons évoqué dans la section précédente, ces éléments sont d'une nature différente, certains pointent de manière précise le type de contenu accédé (les URL), tandis que d'autres sont plus large (les mots clefs par exemple). Notre intuition est que l'utilisation de ces différents éléments et leur éventuelle pondération est de nature à influencer le clustering et par conséquent les performance du système de cache distribué.

En effet, une pondération mettant en avant les informations précises comme les URL conduirai à obtenir des réseau de pairs partageant des intérêts commun très précis et donc susceptibles de profiter du cache distribué pour des contenus ayant un rapport direct avec leur habitudes de navigation.

À l'inverse, l'utilisation d'éléments de profil plus larges (et plus imprécis) tel que les mots clefs des pages visitées conduira vraisemblablement à un clustering plus large, le cache serai moins utilisé pour les contenus au cœur des intérêts des utilisateur, mais celui-ci est susceptibles de contenir des éléments qu'ils seraient potentiellement amené à visiter, permettant ainsi une utilisation du cache plus poussée lors de la recherche d'informations, ou encore la réutilisation du clustering ainsi obtenu pour la conception de systèmes de recommandation distribué.

Ainsi, l'utilisation d'un mécanisme de pondération attribuant des poids aux divers éléments afin d'obtenir un clustering mettant à la fois en avant une part d'éléments précis, et une part d'éléments plus larges semble être un compromis idéal afin de profiter des intérêts cumulés des deux mécanismes.

4.3.3 Un cache web local

Afin de permettre aux utilisateurs de partager les contenus mis en cache, nous créons un cache web local. Celui-ci stocke les contenus accèdés pour les mettre en cache durant le processus de navigation. Afin d'éviter les problèmes de "cache stale" où un utilisateur obtiendrai un contenu depuis le cache, alors qu'une version plus récente est disponible, nous imitons les comportements des navigateurs en réutilisant les politiques de cache définis par les sites web. Ceci, nous permet de respecter la compatibilité avec les infrastructures existantes aussi bien que l'expérience utilisateur, en évitant des changements perceptibles qui pourraient survenir lors de l'utilisation d'une politique de cache trop agressive.

Pour ce faire, notre cache respecte les politiques de cache exportés par les serveur web. Ces politiques, décrites dans le RFC2616 [43] et utilisées aussi bien

par les navigateurs que par les éventuels relais (proxys) intermédiaires permettent aux sites renvoyer aux clients web des informations sur la politique de cache à adopter vis à vis des contenus renvoyés :

- La possibilité ou non de mettre un élément en cache.
- La durée maximum de mise en cache d'un contenu.
- Le caractère privé ou public du contenu mis en cache.
- La nécessité de vérifier ou non la présence d'un contenu plus récent via le serveur d'origine avant d'utiliser la version mise en cache.

Le respect de ces différentes informations, émises par le site web, pour chacun des contenus permettent de préserver l'expérience utilisateur actuelle et donc d'envisager un déploiement progressif de la solution présentée.

Pour les sites web n'exportant pas de politique de cache explicite, une politique de cache basée sur une heuristique est disponible. Cette heuristique définie dans la RFC2616 [43], permet de forcer la mise en cache d'un contenu pour une durée donnée en l'absence de politique explicite. La durée de mise en cache peut être exprimée de deux manières, soit en fonction de l'âge de document, soit via une durée explicitement choisie. Passée cette durée, le serveur sera à nouveau consulté pour vérifier la présence d'un nouveau contenu. Si cette durée n'a pas encore été atteinte ou si le serveur ne propose pas de nouveau contenu, la version en cache sera utilisée.

Nous pensons que l'utilisation de cette heuristique est de nature à améliorer grandement les performances de notre système de cache [54], un choix judicieux de la durée de mise en cache devra toutefois être effectué afin de ne pas perturber le processus de navigation. Celui-ci fera l'objet d'expérimentations futurs.

4.3.4 Un cache web pair-à-pair

Pour permettre aux utilisateurs de s'échanger les contenus mis en cache nous proposons un système de cache pair-à-pair. Celui-ci est construit à partir de la fusion de l'ensemble des caches des voisins. Ceci, dans le but de permettre d'accéder de manière transparente à l'ensemble des caches des nœuds voisins.

Une première version est construite via un système de partage des index de cache des nœuds voisins. Ce fonctionnement, simple, permet à un nœud de connaître l'ensemble des éléments de cache des voisins et de s'assurer de manière immédiate de la présence ou non d'un élément dans le cache pair-à-pair.

Lorsqu'une entrée de cache est demandée au système de cache pair-à-pair, celui-ci vérifie si une entrée est présente dans les caches des voisins. Si c'est le cas,

cette entrée est demandée. Si plusieurs nœuds possèdent cette entrée de cache, un nœud est sélectionné de manière aléatoire afin de répartir la charge. Dans le cas contraire (aucune entrée de cache n'est disponible) une erreur est renvoyée.

Toutefois, ce système de partage des index de cache pose d'évidents problèmes de vie privée. En effet, le fait d'avoir accès à l'ensemble des entrée de cache des voisins, permet d'avoir des informations précises sur les visites et ainsi de dresser un profil précis des utilisateurs, à l'image des recoupements ayant été fait sur les bases de donnée netflix et AOL [71, 80]. L'utilisation de mécanismes permettant de contourner ce problème sera discuté en section 7.2.2.

4.3.5 Un moteur web

Afin d'intégrer notre système de distribution de contenu en pair-à-pair au processus de navigation, il était nécessaire de concevoir un moteur web, destiné à recevoir les requêtes et les traiter.

Celui-ci fonctionne de la manière suivante : Lorsqu'un contenu est demandé, il consulte d'abord le cache local, puis si aucune réponse n'est trouvée, le cache pair-à-pair. Enfin, si aucune entrée de cache n'est disponible en local ou en pair-à-pair, celui ci demande le contenu auprès du serveur d'origine. Dans tous les cas, le contenu récupéré est renvoyé au système d'analyse des intérêts des utilisateurs. Le fonctionnement de ce moteur web est illustré par l'algorithme 1.

5 Implémentation

Afin de valider notre système de partage de cache entre les navigateurs, nous avons choisi d'implémenter celui-ci, pour nous permettre de tester celui-ci en condition réelles, aux travers de d'expérimentations, plutôt que de ne disposer que de simulations. Pour ce faire, nous avons étudié deux possibilités. L'implémentation sous la forme d'un plugin de navigateur qui semblait la voix la plus naturelle, et l'implémentation sous la forme d'un logiciel externe.

5.1 Piste envisagées

5.1.1 Un plugin les navigateurs web

L'implémentation sous la forme d'un plugin pour le navigateur semblait la meilleure solution :

En effet elle permettait de profiter d'une grande intégration avec le navigateur et donc de tirer partie d'un ensemble de briques déjà existantes telles que la gestion d'un cache, l'exploitation des méta-données des pages consultées, ou encore l'implémentation d'une API de gestion des connections directes entre les

Algorithme 1 Traitement des requêtes web par le cache pair-à-pair.

```
1  handle(request_url){
2    CacheElement element
3
4    if(localCache.contain(request_url)){
5      element = localCache.get(request.url);
6    }
7
8    else if (p2pCache.contain(request_url)){
9      element = p2pCache.get(request_url);
10     localCache.put(request_url, element);
11   }
12
13   else {
14     element = httpConnection.get(request_url);
15     localCache.put(request_url, element);
16   }
17
18   InterestDataBase.parse(element.copy());
19   return element;
20 }
```

navigateurs : webRTC.

Toutefois, cette solution avait comme principal inconvénient d'entraîner une dépendance forte au navigateur utilisé et au système d'exploitation sous-jacent, obligeant à adapter l'implémentation aux multiples navigateurs et systèmes d'exploitations existants.

En outre elle obligeai à dépendre d'un navigateur dans la mise en place d'expérimentation ce qui aurai pu augmenter les difficultés de mise en place.

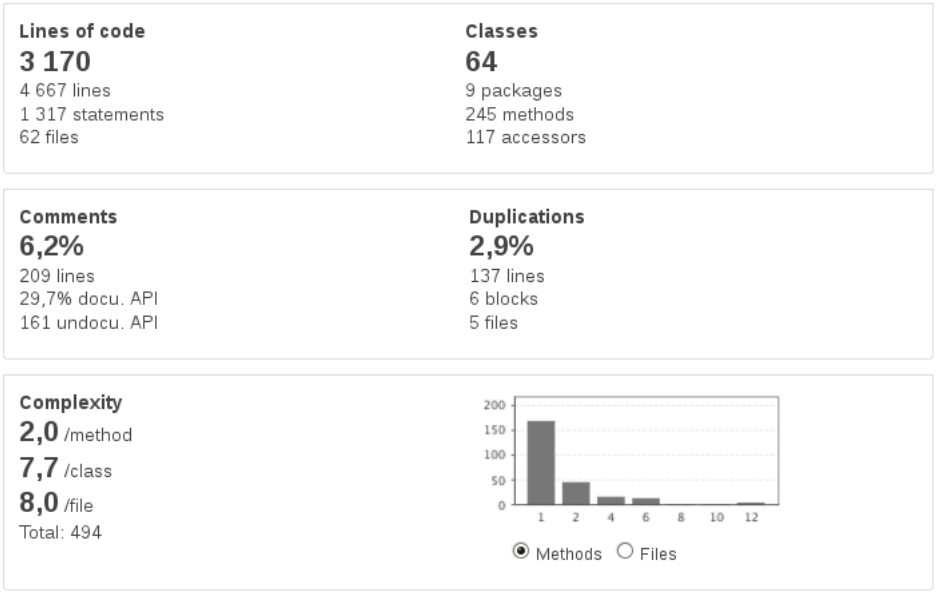
5.1.2 Un proxy web externe

Une autre piste envisagée a été l'implémentation sous la forme d'un logiciel externe, chargé d'intercepter les requêtes des navigateurs pour les faire passer au travers du système de cache pair-à-pair. Cette solution, bien que moins naturelle que la précédente au premier abord, offrait plusieurs intérêts :

Celle ci permettait d'envisager l'écriture d'un logiciel portable, utilisable au travers de nombreux client web, sur de nombreux systèmes d'exploitation.

Cette portabilité, permettait également d'envisager une phase d'expérimentation simplifiée, puisse ce que réalisable au travers de n'importe quel client

FIGURE 7 – Métriques de notre implémentation



web, y compris des clients web simplifiés et en mode texte.

Enfin, cette approche offrait également l'intérêt de pouvoir être déployée pour un ensemble d'utilisateurs en une seule instance, permettant ainsi de profiter par exemple, des boxes fournis par les fournisseurs d'accès pour déployer notre système [30].

À l'inverse cette approche, obligeait à un développement plus conséquent, pour imiter voir contourner, certaines fonctionnalités des navigateurs, telles que la gestion d'un cache, ou encore l'exploitation des meta-données contenus dans les pages.

5.2 Une première implémentation

Attentifs aux problématiques de portabilité, tout comme à la nécessité de pouvoir utiliser notre implémentation pour piloter des expérimentations, c'est finalement la solution du développement d'un proxy web que nous avons retenu. Afin de disposer d'un système portable, nous avons choisi de développer cette implémentation en Java. Pour pouvoir réutiliser aisément différentes librairies nous avons choisi d'utiliser Maven, outil de compilation pour Java qui permet entre autre une gestion automatique des dépendances. Cette implémentation est construite autour de plusieurs éléments :

La configuration est gérée par la librairie apache commons-configuration. Celle-ci nous permet de définir, pour chaque élément un ensemble de paramètres dont la valeur peut être lue depuis un fichier de configuration, comme en ligne de commande. Ceci nous permet de pouvoir définir des valeurs par défaut pour le logiciel, tout en ayant la possibilité de revenir sur celles-ci pour adresser des cas spécifiques telles que les expérimentations.

Des logs sont générés tout au long du fonctionnement de l'application. En utilisant la librairie log4j, nous pouvons définir différents niveaux de logs (message de deboggage, erreur..) pour chacun des différents éléments de notre application. Ceci pour nous permettre d'analyser un à un les différents composants de notre implémentation et extraire les informations importantes en cas de besoin.

Un proxy web est utilisé pour recevoir les requêtes des utilisateurs. Celui-ci est construit autour de la librairie apache HTTP Components. Afin de pouvoir gérer un grand nombre de connexions simultanées, nous utilisons un modèle par événements non-bloquant, ainsi qu'un "pool" de threads. Ainsi, pour chaque requête utilisateur reçue, un nouveau thread est instancié (ou réutilisé si disponible). Ce thread lance un client web (construit là encore à l'aide la librairie HTTP Components) chargé d'effectuer la requête, après validation d'éventuelles données disponibles en cache. Une fois la réponse reçue, les données sont copiés pour être analysées en tâche de fond, permettant ainsi de les renvoyer au client au plus vite, sans nécessité d'attendre la fin de l'analyse.

Un moteur d'analyse des intérêts des utilisateurs est créé. Là encore, celui ci utilise un "pool" de thread pour pouvoir effectuer de nombreux traitements simultanés et pouvoir traiter les données reçues au fil de l'eau. Celui-ci parcours les méta-données des réponses reçues (URL demandée, nom de domaine du serveur, code HTTP reçu...) ainsi que les contenus renvoyés. Ces contenus sont analysés pour en extraire les informations décrites en section 4.3.1 à l'aide d'un "parseur" XML spécifique, le parseur HTML5 utilisé dans les navigateurs mozilla. En effet, La plupart des pages web étant des documents XML mal formatés, il convenait d'utiliser un parseur spécifique capable de traiter de tels documents.

Une base de donnée de profils utilisateurs est ainsi constituée à partir des contenus analysés. Afin de pouvoir être traités facilement ces informations de profil sont stockés dans une base SQL. Le moteur de base de données utilisé, H2, a été choisi pour son empreinte mémoire réduite et le fait d'être 100% Java (lui permettant ainsi d'être portable). Afin de pouvoir être traités de manière transparente, ces différentes informations constituant les profils utilisateurs sont accèdés via un mécanisme de translation entre objets et base de donnée : JPA.

Un cache web local est créé, en utilisant la libraire EhCache. Cette librairie nous permet de mettre en place facilement un cache local, stockant les éléments

en mémoire et sur disque, consultable par le client web décrit plus haut.

Un cache web pair-à-pair est construit comme une extension (décorateur) au cache web local, celui-ci vérifie le contenu du cache local comme décrit dans l'algorithme 1. Si un contenu est présent dans le cache local, celui-ci est renvoyé, sinon, ce contenu est demandé au moteur pair-à-pair.

Un moteur pair-à-pair est construit autour de l'implémentation de Gossple [22], et des divers librairies sous jacentes (converties à Maven pour l'occasion). Celui-ci utilise les informations issues du profil pour calculer la similarité entre deux nœuds et ainsi décider ou non de conserver un nœud comme voisin. Pour tous les voisins, les index de cache sont régulièrement demandés, afin d'avoir accès de manière précise à l'ensemble des contenus disponibles via le système de cache en pair-à-pair. C'est, ce moteur pair-à-pair qui est chargé de toutes les communications avec les pairs, et en particulier la mise en place du cache pair-à-pair : Lorsqu'une demande d'un pair est reçue, le cache local est consulté et le contenu (ou une erreur information de l'absence de contenu) est renvoyé. De la même manière lorsqu'une demande est reçue en local, le contenu est demandé à un pair en utilisant l'index de cache (comme décrit en section 4.3.4) et la réponse reçue renvoyée.

Une API est disponible afin de pouvoir consulter différentes statistiques et pouvoir contrôler le comportement de notre implémentation via un navigateur. Ainsi, il est possible de demander au système de re-sélectionner des voisins, vider les caches ou la base de donnée contenant les profils, ou encore de consulter des informations tels que les profils utilisateurs, la liste des pairs du réseau, les index de cache locaux et distant, ou encore les différentes métriques utilisées pour notre analyse de performance.

Cette implémentation, écrite à l'aide de quelques 4500 lignes de code, répartie en une soixantaine de classes (voir figure 7), et re-utilisant près d'une trentaine de librairies (voir figure 6.3), nous permet de proposer un système portable de diffusion de contenu en pair-à-pair, utilisable aussi bien aussi bien au travers des navigateurs, qu'au travers de clients texte permettant la mise en place aisée d'expérimentations à large échelle.

6 Évaluation

6.1 Une première évaluation

Afin de valider notre modèle, nous avons tout d'abord chercher à mesurer le sur-coût introduit par celui ci, et les performances qu'il était possible d'attendre du système de cache distant. Pour cela nous avons cherché à mesurer le

FIGURE 8 – Dépendances de notre implémentation

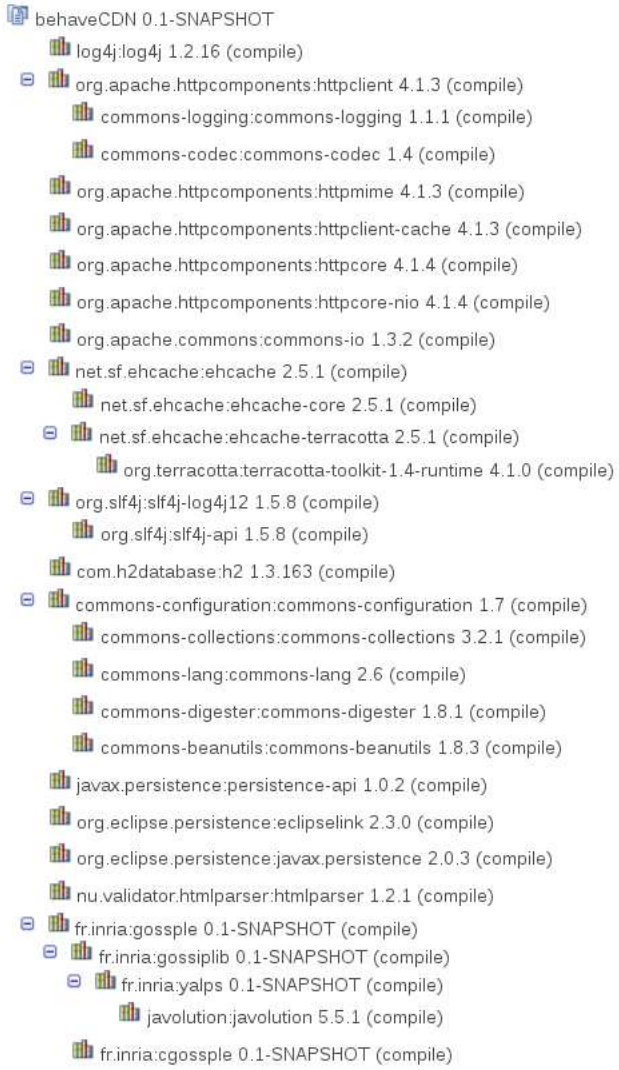


TABLE 1 – Temps d'accès moyen au site linuxfr.org

Méthode d'accès	Direct	Proxy	Proxy - cache local	Proxy - cache p2p
Temps d'accès (sec)	3.694	3.703	1.548	1.904

temps nécessaire afin de récupérer les pages d'accueil de plusieurs sites, ceci de plusieurs manières :

- Via un accès direct
- Via notre proxy
- Via le local local
- Via le cache pair-à-pair (via l'utilisation d'une machine virtuelle locale).

Le tableau 1 présente les résultats moyens obtenus pour récupérer la page d'accueil du site linuxfr.org, comprenant 84 éléments (page, images, scripts, feuilles de styles) pour un poids de 704Ko. Ces tests ont été effectué 25 fois, depuis une connexion domestique. Ceux-ci nous montre le faible sur-coût que cause notre système :

- Très peu de différence entre le temps d'accès direct, et via notre proxy
- Très peu de différence entre le temps d'accès via le cache local et le cache distant

6.2 Métriques

Afin d'évaluer notre modèle, nous nous intéressons à deux critères principaux :

- L'expérience utilisateur, au cœur même de notre modèle.
- La quantité de données échangées via notre système de cache pair-à-pair.

Pour cela nous utilisons différentes métriques, couramment utilisés dans l'analyse de performances de CDN :

- Le temps d'accès aux divers contenus visités, utilisé comme indicateur de l'expérience utilisateur.
- Le taux de hit du système de cache distribué, et la quantité de données échangées via celui-ci comme indicateur des performances "brutes"

de notre système.

6.3 Collection de traces

Afin d'évaluer notre système nous avons d'évaluer celui ci à l'aide de traces réêles obtenus à partir de données de navigation d'utilisateurs, plutôt que d'utiliser des traces générées à l'aide de modèles synthétiques.

Si il est aisé d'obtenir des traces de serveur web, il est plus difficile d'obtenir des traces de navigation (issues typiquement de serveurs proxy ou directement de navigateurs web). En effet, la mise à disposition de telles traces pose d'évidents problèmes de vie privée, comme évoqué en section 4.3.4. Bien que des méthodes existent pour circonvenir ces problèmes (à l'image de la RFC6235 [25]), il n'existe que peu d'organismes mettant à disposition de telles données, qui plus est assez récentes pour être pertinentes dans le cadre de l'évaluation d'un système de distribution de contenu en pair-à-pair [2].

Le projet IRCache [8] propose toutefois de telles traces. Celle-ci sont accessibles sur demande, après justification, et partiellement anonymisées (Les adresses IP sont remplacées par des adresses aléatoires, certaines parties des requêtes sont tronquées...). Une partie de ces traces sont accessibles publiquement, dans le cadre d'une initiative coordonnée visant à mettre à disposition des traces exploitables pour l'analyse et la caractérisation du trafic internet [29].

Le dataset proposé est ainsi constitué de quelques 13 millions de visites web de près de 2000 clients web, obtenus depuis une série de serveurs proxys mis en place dans divers universités aux états unis [8], et portant sur deux jours : les 9 et 10 janvier 2009.

D'autres traces sont également disponibles auprès du RIPE NCC et de l'université de Waikato [53, 66]. Celles-ci comportent aussi bien des données de navigation, que des exports brutes de paquets réseaux obtenus depuis divers routeurs. Leur utilisation pourra faire l'objet de travaux futurs.

6.4 Éléments d'analyse

Afin de valider notre modèle et de convenir des paramètres pouvant conduire à de meilleures performances, nous nous intéressons à l'analyse de plusieurs éléments :

Les paramètres du réseau pair-à-pair Gossple

2. Le projet Internet Trafic Archive [36] propose par exemple de telles traces, toutefois celle-ci datent des années 1990, et semblent donc peu à même de refléter les habitudes de navigation actuelles.

- Nombre de voisins dans le RPS et les *Gnets*
- Nombre de voisins échangés dans le RPS et les *Gnets*
- Intervalle de mise en à jour des voisins

Les paramètres de notre modèle de calcul de similarité

- Le choix de la méthode de Jaccard ou de la similarité cossine
- L'impact d'éventuelles pondérations entre les différents types d'éléments

Les paramètres de notre système de cache

- L'impact de la politique de cache via une heuristique
- Les meilleurs paramètres pour cette heuristique

6.5 Expérimentation

Afin de tirer pleinement profit des datasets obtenus, nous avons choisi de rejouer ceux-ci de manière expérimentale. Bien que de nombreux modèles existent pour parvenir à des simulations fiables [21,26,27], il nous semblait plus judicieux de réutiliser les traces obtenues et de rejouer celles-ci de manière réaliste.

En effet, la politique de cache actuellement utilisée par notre modèle et décrite en section 4.3.3 prend en compte les politiques de cache des sites web visités, et leur éventuelle mise à jour durant le processus de navigation. Aussi, il nous semblait délicat de parvenir à modéliser de manière réaliste ces différents comportements et les conséquences d'éventuelles actions extérieurs (mise à jour, suppression d'un contenu...) sur les performances de notre système.

Nous avons donc choisi de rejouer l'ensemble des traces obtenues de manière distribuée afin d'associer à chaque client web, une instance de notre système. Pour ce faire nous utilisons le réseau Planetlab [32] [3]La méthodologie appliquée est la suivante :

1. Isoler les visites de chaque client.

3. L'utilisation du réseau Grid5000 [24] avait été également envisagée. Toutefois, elle aurait obligé à inscrire l'ensemble des sites visités dans une liste blanche, ce qui ne semblait pas envisageable au vu de notre volume de donnée.

2. Pour chacun des clients web obtenus, et par ordre décroissant de nombre de visites [4], associer une instance de notre système sur le réseau Planetlab.

3. Rejouer en simultané, en respectant les délai entre les visites, et sur l'ensemble des instances de notre système, les traces obtenus à l'étape 1.

4. Analyser les résultats au cours de l'exécution et au terme de celle-ci.

A l'heure de l'écriture de ce rapport, et compte tenu des difficultés requises pour piloter de telles expériences de dont chaque itération prend 24h, monopolise plusieurs centaines de nœuds et effectue plusieurs milliers de requêtes Web, pour tester les différents paramètres de notre modèle, ce protocole expérimental n'a pas pu être effectué dans son intégralité. Les résultats de celui-ci sont attendus prochainement.

6.6 Compléments

Au delà de l'indispensable validation expérimentale des résultats obtenus, il semble pertinent de réfléchir à de nouvelles pistes pour choisir les bons paramètres de notre modèle, et valider ceux-ci via une interaction directe avec des utilisateurs. Pour cela deux pistes d'études sont prévues :

- La mise en place d'étude sur des panels utilisateurs

- Une éventuelle diffusion open-source

6.6.1 Une étude sur un panel d'utilisateurs

Afin de compléter ce processus expérimental, il est aussi envisageable de procéder à des études, sur des panels utilisateurs restreints, afin de mesurer les performances du système de cache distribué, et le ressenti utilisateur, lors de l'utilisation de différents paramètres.

Le fait de disposer d'une implémentation, utilisable, et portable sur de nombreux systèmes d'exploitation et navigateurs, est de nature à faciliter grandement la mise en place de telles expérimentations.

L'un des grands intérêts de ce type d'expérimentation est la facilité de déployer différentes versions utilisant différents paramètres pour une durée donnée. À l'inverse, l'utilisation d'un panel restreint, d'utilisateurs avertis [5], est de nature à biaiser les résultats. Pour circonvenir à ce problème, une diffusion plus large semble plus appropriée.

4. Le réseau Planetlab n'offrant pas suffisamment de nœuds pour nous permettre de rejouer l'ensemble du dataset obtenu, nous nous intéressons uniquement aux clients les plus pertinents pour notre modèle : ceux ayant effectué le plus de visites.

5. au courant des mécanismes sous-jacents.

6.6.2 Une diffusion open-source

Ainsi, l'une des dernières pistes envisagée afin d'obtenir des résultats, serai une diffusion de l'implémentation réalisée sous la forme d'un logiciel open-source. Il convient toutefois pour cela de valider le fonctionnement de l'application de choisir au mieux les paramètres, mais aussi et surtout de revenir sur certaines problématiques écartées durant la conception de ce premier modèle, tel que la prise en compte des informations réseaux ou les problématiques de respect de la vie privée.

7 Futurs travaux

Notre modèle, permettant la diffusion de contenu web, en pair-à-pair et en réutilisant les réseaux épidémiques pour construire une infrastructure basée sur les modèles sociaux, laisse entrevoir de nouvelles possibilités qui pourraient faire l'objet de futur travaux.

- Dans un premier temps, il semble nécessaire d'adresser certaines problématiques écartées :
  - La prise en compte des problématiques liées aux mécanismes réseaux sous-jacents semble l'une des principales contraintes à adresser dans le but de maintenir voir d'augmenter l'expérience utilisateur amenée par notre système.
  - De la même manière, la prise en compte des problématiques liées à la vie privée des utilisateurs, semble être l'une des priorités qui devra guider de futurs travaux.
  - Enfin, la conception, ou la réutilisation d'un protocole spécifique, permettant le transfert de contenus multi-chemin et implémentant des mécanismes incitatifs, nous semble l'une des clefs nécessaires à notre modèle.
- A plus long terme, la conception d'un nouveau modèle de sécurité, permettant de reproduire les fonctions de chiffrement et d'authentification, nous semble être l'un des problèmes les plus ouverts, pour la diffusion de contenu Web en pair-à-pair.

7.1 Prise en compte des informations réseau

L'expérience utilisateur, au cœur même de notre modèle, nous amène à considérer la nécessité de prendre en compte les mécanismes réseaux sous jacents. La section 2.3.2 a montré que de nombreux travaux ont eu lieu dans ce sens, réduisant quelque peu la nécessité de concevoir un nouveau modèle. C'est ainsi que nous avons choisi de nous concentrer sur la réalisation d'un réseau basé sur les modèles sociaux. Toutefois, pour être pertinent celui-ci se doit de prendre en compte les contraintes réseaux sous-jacentes. Ainsi, la proximité réseau des

nœuds voisins, ou encore la bande passante disponible auprès de ceux-ci semblent autant d'éléments qu'il est nécessaire d'adresser dans le but de préserver l'expérience utilisateur, voir de l'améliorer.

7.1.1 localité réseau

L'utilisation des informations sur la localité réseau et la proximité des pairs nous apparaît comme l'un des éléments centraux de notre modèle. Si ceux-ci permettent d'augmenter la qualité de l'expérience utilisateur en permettant des débits plus haut et une meilleure réactivité, dus à une latence réduite, ils semblent également être un second levier permettant d'inciter au déploiement de notre système. En effet, et notre étude l'a monté, les fournisseurs d'accès ont tout a gagné à augmenter la localité du trafic réseau dans le but de diminuer leurs coûts.

La latence, le nombre de sauts ou encore le nombre d'opérateurs (AS) traversés sont autant d'informations qui permettent d'obtenir de bonnes heuristiques de la localité réseau entre deux pairs et des topologies sous-jacentes. L'utilisation corrélée de celle-ci telle qu'utilisée par TopBt [81] permet d'augmenter de manière significative la localité réseau sans pour autant introduire de dépendances vis à vis d'infrastructure externe[6].

La réutilisation de ces informations dans notre modèle pourrait être envisagée de plusieurs manières : Si la mise en place d'un nouveau niveau de clustering (situé typiquement entre le RPS et les Gnet) peut sembler l'approche idéale elle risque d'amener un trop grand niveau de clustering et de reproduire les problèmes de FlowerCDN (deux pairs ayant des intérêts extremement proches mais quelque-peu éloignés dans le réseau ne seraient pas voisins). À l'inverse, l'utilisation d'un clustering basé à la fois sur la proximité d'intérêt et sur la proximité réseau semble une approche plus élégante et efficace, mais elle requière de bien analyser la pondération de ces deux éléments (afin, typiquement de ne pas trop biaiser le système de clustering par intérêt au profit de pairs très proches dans le réseau mais n'ayant que peu d'intérêts communs).

L'analyse de ces différentes méthodes et des paramètres pouvant conduire à de meilleurs performances tout comme l'impact global au niveau de l'économie du réseau et de son empreinte énergétique, de l'augmentation de la localité du trafic web, pourront faire l'objet de futurs travaux.

7.1.2 Contraintes de bande passante

Notre étude l'a également montré, l'utilisation de mécanismes de prise en compte de la bande passante disponible auprès des nœuds semble être un levier important d'augmentation des performances du système. Pour estimer cette bande passante, le modèle décrit par Suselbeck et al. dans leur publication de

6. Mise à part pour l'import initiale des données de routage BGP.

2011, et décrit en section 2.3.2 semble être une solution idéale.

Afin de tirer pleinement ces informations, l'utilisation du modèle d'heterogeneous gossip [49], qui consiste à adapter le fanout d'un nœud en fonction de sa bande passante et ainsi augmenter ou diminuer le nombre de ses voisins semble être une solution idéale. L'intégration de celui-ci et le choix des paramètres optimaux pour notre modèle pourront faire l'objet de futurs travaux.

7.1.3 Un protocole de transport

Si il est une chose qui a fait le succès de BitTorrent, c'est bien l'implémentation d'un protocole spécifique, et incitatif. Nous en avons parlé en section 2.3.6. La conception d'un tel protocole, tel que Swift/PPSP [73], adapté à notre cas d'usage et à notre volonté de développer un système réactif, nous semble l'une des clefs du succès de notre modèle. En outre, de tels protocoles, peuvent permettre de remonter des informations sur la base passante disponible ou encore la congestion réseau, permettant ainsi d'adapter notre système aux contraintes de bandes passantes évoquées plus haut [73, 88].

L'exemple de OneSwarm [57] l'a montré, L'utilisation d'un tel protocole, dans le but de découper les données échangées en pièces et de sélectionner les meilleurs pairs pour obtenir celles-ci, et permettre un télé-chargement multi-chemin, permet d'améliorer grandement les performances (de l'ordre d'un facteur 20). En outre, un tel protocole peut être réutiliser afin d'inciter à la coopérations entre les pairs dans le but par exemple de créer un système de routage en oignon, afin de préserver la vie privée des utilisateurs.

7.2 Les problématiques de vie privée

De la même manière, les problématiques de respect de la vie privée, au cœur même de nos pré-requis ont été écartés durant la conception de notre modèle. Toutefois, la prise en compte de ceux ci semble être un élément indispensable à notre modèle. Pour cela plusieurs pistes sont envisagées : la mise en place d'un système de routage en oignon permettrai d'assurer l'anonymat des communications, tandis que des mécanismes originaux tels que les systèmes d'obfuscation de profils ou le scoring similarity [15] permettrait d'augmenter le niveau de confidentialité des informations échangées.

7.2.1 Anonymiser les communication

Le routage en oignon, popularisé par le logiciel Tor [38] repose sur un principe simple, celui d'introduire des nœuds intermédiaires lors des communications. Ceci, afin que deux nœuds communiquant ensemble ne puisse connaître l'identité de leur pair. Pour palier à certaines attaques, telles que les attaques par analyse des délais de transmission [70] , ou les attaques par collusion, Tor repose sur des mécanismes originaux tels que le rajout de délai aléatoires entre

les nœuds ou encore un mécanisme de diffusion probabiliste.

Ces mécanismes, réutilisés dans Gossple [22],sur lequel notre système repose, ou dans OneSwarm [57] permettent ainsi d'augmenter l'anonymat des communications, au prix toutefois, d'une augmentation des délais de transmission. L'impact de ces différents mécanismes sur notre système, et le choix des paramètres optimaux, dans le but de préserver la vie privée des utilisateurs, tout en maintenant une qualité d'expérience acceptable, demeure à évaluer. De plus, ces systèmes, si ils permettent d'anonymiser les communications, ne seraient être complets sans l'implémentation de mécanismes permettant d'anonymiser le contenu de ces communications.

7.2.2 Anonymiser les données échangées

En effet, et de nombreuses recherches l'ont montré, le fait de masquer l'identité des nœuds impliqués dans une communication, ne suffit pas à préserver leur anonymat. Bien souvent, le contenu des données échangées est susceptible de contenir des informations permettant de retrouver l'identité des utilisateurs impliqués [22, 71, 80]. Le fait de partager directement des profils de navigation, tout comme des index de cache n'est pas sans poser de problèmes. Si il existe des méthodes permettant d'effectuer des calculs de similarité sans dépendre de l'envoi du profil complet [15], le fait de partager des index de cache demeure une des limites de notre modèle qui pourrait faire l'objet de futurs travaux.

7.3 Un nouveau modèle de sécurité

L'une des principales limites que rencontre notre modèle concerne les contenus chiffrés et authentifié. En effet, le modèle actuel de sécurisation utilisé pour une partie des échanges sur le Web repose sur un mécanisme de chiffrement et d'authentification de bout en bout entre les clients et les serveurs. Ceci, au travers du chiffrement et de l'authentification du canal de transmission via les protocoles SSL et HTTPS.

Aussi, il semble délicat de préserver ce modèle lors de l'utilisation de mécanismes pair-à-pair où les contenus échangés ne seront plus forcément reçus directement depuis le serveur d'origine mais peuvent passer par de multiples intermédiaires. En lieu et place, il convient de prendre en compte les mécanismes basés non plus sur l'authentification et le chiffrement des canaux de transmission mais sur les contenus eux mêmes, permettant ainsi de préserver ces fonctionnalités de chiffrement et d'authentification de bout en bout lors de l'utilisation de multiples intermédiaires.

L'utilisation de ces nouveaux mécanismes, déjà utilisés pour le DNS [18] en raison du grand nombre d'intermédiaires possible entre le client et le serveur d'origine, n'est pas sans poser de problèmes. En effet, ceux-ci cassent le principe de retro-compatibilité cher à notre modèle et posent d'évidents problèmes de

déploiement.

Ainsi, la prise en compte de ces problématiques, et la définition d'un nouveau modèle de sécurisation des échanges, compatible avec l'infrastructure existante et/ou facilement déployable, nous appairait comme l'un des principaux mais aussi plus délicats problèmes à prendre en compte dans de futurs travaux.

8 Conclusion

Partant du constat que les solutions actuelles de diffusion de contenus web utilisant les modèle client-serveur centralisé, tendent à montrer leurs limites en terme de passage à l'échelle et de robustesse, nous avons cherché à étudier diverses solutions. Dans une première partie, nous avons décrit les modèles actuellement disponibles, et les contraintes sous-jacentes. Dans une seconde partie, nous sommes revenus sur quelques pré-requis qu'ils nous semblaient nécessaire d'adresser pour concevoir un système efficace de distribution de contenu en pair-à-pair. À partir de l'analyse de ces éléments nous avons décrit un nouveau modèle, permettant la mise en place d'un nouveau modèle de distribution de contenu, pour le web, en pair-à-pair au travers de la mise en place d'un cache partagé, construit au dessus d'un réseau "social". Nous avons enfin décrit notre implémentation de ce modèle et le processus d'évaluation de celui-ci. Enfin, nous avons terminé cette étude, par l'analyse des problèmes encore ouverts à l'heure actuelle dans notre modèle.

Notre système, behaveCDN, permettant la mise à disposition de contenus en pair-à-pair, pour le web, est construit comme un cache distribué au dessus d'un réseau prenant en compte les centres d'intérêt des utilisateurs. Il est conçu afin de maintenir voir d'augmenter l'expérience utilisateur, tout en restant compatible avec l'infrastructure actuellement utilisée sur le web. Il permet de décharger cette infrastructure et d'augmenter ses facultés de montée en charge et sa tolérance aux pannes. Utilisant des algorithmes pair-à-pair non structurés, pour construire un réseau de partage de contenu, notre modèle est également très robuste. Il est naturellement résistant aux pannes et offre de bonnes facultés de montée en charge.

Afin d'évaluer notre modèle, et de réutiliser celui-ci une implémentation est disponible. Celle ci peut également être utilisée comme cadre d'expérimentation réel pour des travaux sur les algorithmes pair-à-pair, les problématiques de vie privée ou encore les systèmes de recommandation distribués. Celle-ci est portable, et peut être utilisée aussi bien pour des tests sur des panels réduits que pour des expérimentations à large échelle.

Des résultats pratiques, obtenus à partir d'expérimentations à large échelle seront disponibles prochainement.

Références

[1] Akamai. http://www.akamai.com.

[2] Amazon Cloudfront. http://aws.amazon.com/fr/cloudfront.

[3] Amazon CloudFront Pricing. http://aws.amazon.com/fr/cloudfront/pricing.

[4] Application-Layer Traffic Optimization Work Group . https://datatracker.ietf.org/wg/alto/charter/.

[5] Don't bubble us : Duckduckgo tutorial. http://dontbubble.us/.

[6] Don't track us : Duckduckgo tutorial. http://donttrack.us/.

[7] Freenet news. https://freenetproject.org/news.html.

[8] The ircache projet. http://www.ircache.net.

[9] Limelight. http://www.limelight.com.

[10] RIPE NCC's albatross ipv6 statistics. http://albatross.ripe.net/v6-clientresolver/.

[11] Tor metrics portal : Users. https://metrics.torproject.org/users.html.

[12] uPnP. http://upnp.org/specs/gw/igd2/.

[13] The world ipv6 launch. http://www.worldipv6launch.org/.

[14] V. Aggarwal, A. Feldmann, and C. Scheideler. Can isps and p2p users cooperate for improved performance? *ACM SIGCOMM Computer Communication Review*, 37(3) :29–40, 2007.

[15] M. Alaggan, S. Gambs, and A.M. Kermarrec. Private similarity computation in distributed systems : from cryptography to differential privacy. *Principles of Distributed Systems*, pages 357–377, 2011.

[16] S. Androutsellis-Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys (CSUR)*, 36(4) :335–371, 2004.

[17] C. Aperjis, M.J. Freedman, and R. Johari. Peer-assisted content distribution with prices. In *Proceedings of the 2008 ACM CoNEXT Conference*, page 17. ACM, 2008.

[18] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard), March 2005. Updated by RFC 6014.

[19] A. Bakker, R. Petrocco, M. Dale, J. Gerber, V. Grishchenko, D. Rabaioli, and J. Pouwelse. Online video using bittorrent and html5 applied to wikipedia. In *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*, pages 1–2. IEEE, 2010.

[20] A. Barbir, B. Cain, R. Nair, and O. Spatscheck. Known Content Network (CN) Request-Routing Mechanisms. RFC 3568 (Informational), July 2003.

[21] P. Barford and M. Crovella. Generating representative web workloads for network and server performance evaluation. In *ACM SIGMETRICS Performance Evaluation Review*, volume 26, pages 151–160. ACM, 1998.

[22] M. Bertier, D. Frey, R. Guerraoui, A.M. Kermarrec, and V. Leroy. The gossple anonymous social network. *Middleware 2010*, pages 191–211, 2010.

[23] B. Bollobás. *Random graphs*, volume 73. Cambridge Univ Pr, 2001.

[24] R. Bolze, F. Cappello, E. Caron, M. Daydé, F. Desprez, E. Jeannot, Y. Jégou, S. Lanteri, J. Leduc, N. Melab, et al. Grid'5000 : a large scale and highly reconfigurable experimental grid testbed. *International Journal of High Performance Computing Applications*, 20(4) :481–494, 2006.

[25] E. Boschi and B. Trammell. IP Flow Anonymization Support. RFC 6235 (Experimental), May 2011.

[26] A. Botta, A. Dainotti, and A. Pescapé. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*, 2012.

[27] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions : Evidence and implications. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 126–134. IEEE, 1999.

[28] J. Brutlag. Speed matters for google web search. *Google. June*, 2009.

[29] T. CAIDA and D.U. FAQ. The cooperative association for internet data analysis, 2009.

[30] Yiping Chen, Jimmy Leblet, and Gwendal Simon. On reducing the cross-domain traffic of box-powered CDN. In IEEE, editor, *ICCCN 2009 : 18th International Conference on Computer Communications and Networks*. INFO - Dépt. Informatique (Institut Mines-Télécom-Télécom Bretagne-UEB), Thomson R&D Rennes (Thomson), 2009.

[31] D.R. Choffnes and F.E. Bustamante. Taming the torrent : a practical approach to reducing cross-isp traffic in peer-to-peer systems. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 363–374. ACM, 2008.

[32] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. Planetlab : an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3) :3–12, 2003.

[33] B. Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems*, volume 6, pages 68–72, 2003.

[34] S. Crocker. Host Software. RFC 1, April 1969.

[35] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi : A decentralized network coordinate system. In *ACM SIGCOMM Computer Communication Review*, volume 34, pages 15–26. ACM, 2004.

[36] P. Danzig, J. Mogul, V. Paxson, and M. Schwartz. The internet traffic archive. *URL http ://ita. ee. lbl. gov*, 2000.

[37] A. Datta, M. Hauswirth, and K. Aberer. Beyond web of trust : Enabling p2p e-commerce. In *E-Commerce, 2003. CEC 2003. IEEE International Conference on*, pages 303–312. IEEE, 2003.

[38] R. Dingledine, N. Mathewson, and P. Syverson. Tor : The second-generation onion router. In *Proceedings of the 13th conference on USENIX Security Symposium-Volume 13*, pages 21–21. USENIX Association, 2004.

[39] P. Eckersley, F. von Lohmann, and S. Schoen. Packet forgery by isps : A report on the comcast affair. *Electronic Frontier Foundation*, 2007.

[40] M. El Dick, E. Pacitti, and B. Kemme. Flower-cdn : a hybrid p2p overlay for efficient query processing in cdn. In *Proceedings of the 12th International Conference on Extending Database Technology : Advances in Database Technology*, pages 427–438. ACM, 2009.

[41] Patrick Th. Eugster, Rachid Guerraoui, Anne-Marie Kermarrec, and Laurent Massoulié. Epidemic information dissemination in distributed systems. *IEEE Computer*, 37(5) :60–67, 2004.

[42] J. Falkner, M. Piatek, J.P. John, A. Krishnamurthy, and T. Anderson. Profiling a million user dht. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 129–134. ACM, 2007.

[43] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Updated by RFCs 2817, 5785, 6266.

[44] B. Ford, P. Srisuresh, and D. Kegel. Peer-to-peer communication across network address translators. In *USENIX Annual Technical Conference*, volume 2005, 2005.

[45] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. Idmaps : A global internet host distance estimation service. *Networking, IEEE/ACM Transactions on*, 9(5) :525–540, 2001.

[46] M.J. Freedman. Experiences with coralcdn : A five-year operational view. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, pages 7–7. USENIX Association, 2010.

[47] M.J. Freedman, E. Freudenthal, and D. Mazieres. Democratizing content publication with coral. In *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation-Volume 1*, pages 18–18. USENIX Association, 2004.

[48] M.J. Freedman, K. Lakshminarayanan, and D. Mazières. Oasis : Anycast for any service. NSDI, 2006.

[49] D. Frey, R. Guerraoui, A.M. Kermarrec, B. Koldehofe, M. Mogensen, M. Monod, and V. Quéma. Heterogeneous gossip. In *Proceedings of the ACM/IFIP/USENIX 10th international conference on Middleware*, pages 42–61. Springer-Verlag, 2009.

[50] P. Garbacki, A. Iosup, D. Epema, and M. Van Steen. 2fast : Collaborative downloads in p2p networks. In *Peer-to-Peer Computing, 2006. P2P 2006. Sixth IEEE International Conference on*, pages 23–30. Ieee, 2006.

[51] C. Griffiths, J. Livingood, L. Popkin, R. Woundy, and Y. Yang. Comcast's ISP Experiences in a Proactive Network Provider Participation for P2P (P4P) Technical Trial. RFC 5632 (Informational), September 2009.

[52] C. GROSS. Le projet renater-cache. In *Journées réseaux*, pages 351–363, 1997.

[53] WAND Network Research Group et al. Wand wits : Auckland-iv trace data. *http ://wand.cs.waikato.ac.nz/wand/wits/auck/4*, 2001.

[54] J. Gwertzman and M. Seltzer. World-wide web cache consistency. In *Proceedings of the 1996 USENIX Technical Conference*, pages 141–151. San Diego, CA, 1996.

[55] S. Hyunjoong. Pcubecast : A novel peer-assisted live streaming system. In *Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on*, pages 212–215. IEEE, 2011.

[56] Google IPv6 statistics. `http://www.google.com/intl/en/ipv6/statistics/`.

[57] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson. Privacy-preserving p2p data sharing with oneswarm. *ACM SIGCOMM Computer Communication Review*, 40(4) :111–122, 2010.

[58] S. Iyer, A. Rowstron, and P. Druschel. Squirrel : A decentralized peer-to-peer web cache. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 213–222. ACM, 2002.

[59] M. Jelasity, S. Voulgaris, R. Guerraoui, A.M. Kermarrec, and M. Van Steen. Gossip-based peer sampling. *ACM Transactions on Computer Systems (TOCS)*, 25(3) :8, 2007.

[60] X. Jin and Y.K. Kwok. Network aware p2p multimedia streaming : Capacity or locality ? In *Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on*, pages 54–63. IEEE, 2011.

[61] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should internet service providers fear peer-assisted content distribution ? In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 6–6. USENIX Association, 2005.

[62] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, et al. Oceanstore : An architecture for global-scale persistent storage. *ACM Sigplan Notices*, 35(11) :190–201, 2000.

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

[64] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials*, 7(2) :72–93, 2005.

[65] Petar Maymounkov and David Mazières. Kademlia : A peer-to-peer information system based on the xor metric. In Peter Druschel, M. Frans Kaashoek, and Antony I. T. Rowstron, editors, *IPTPS*, volume 2429 of *Lecture Notes in Computer Science*, pages 53–65. Springer, 2002.

[66] T. McGregor, S. Alcock, and D. Karrenberg. The ripe ncc internet measurement data repository. In *Passive and Active Measurement*, pages 111–120. Springer, 2010.

[67] B. Metcalfe. Metcalfe's law : A network becomes more valuable as it reaches more users. *Infoworld*, 17(40) :53–54, 1995.

[68] Michael Burford. WebSeed - HTTP/FTP Seeding (GetRight style). `http://bittorrent.org/beps/bep_0019.html`.

[69] A. Muller, N. Evans, C. Grothoff, and S. Kamkar. Autonomous nat traversal. In *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*, pages 1–4. IEEE, 2010.

[70] S.J. Murdoch and G. Danezis. Low-cost traffic analysis of tor. In *Security and Privacy, 2005 IEEE Symposium on*, pages 183–195. IEEE, 2005.

[71] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 111–125. Ieee, 2008.

[72] T.S.E. Ng and H. Zhang. Predicting internet network distance with coordinates-based approaches. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 170–179. IEEE, 2002.

[73] F. Osmani, V. Grishchenko, R. Jimenez, and B. Knutsson. Swift : the missing link between peer-to-peer and information-centric networks. In *Proceedings of the First Workshop on P2P and Dependability*, page 4. ACM, 2012.

[74] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do incentives build robustness in bittorrent. In *Proc. of NSDI*, volume 7, 2007.

[75] M. Piatek, T. Kohno, and A. Krishnamurthy. Challenges and directions for monitoring p2p file sharing networks-or : why my printer received a dmca takedown notice. In *Proceedings of the 3rd conference on Hot topics in security*, page 12. USENIX Association, 2008.

[76] M. Piatek, H.V. Madhyastha, J.P. John, A. Krishnamurthy, and T. Anderson. Pitfalls for isp-friendly p2p design. In *Proc. of HotNets*, 2009.

[77] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The bittorrent p2p file-sharing system : Measurements and analysis. *Peer-to-Peer Systems IV*, pages 205–216, 2005.

[78] J.A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D.H.J. Epema, M. Reinders, M.R. Van Steen, and H.J. Sips. Tribler : a social-based peer-to-peer system. *Concurrency and Computation : Practice and Experience*, 20(2) :127–138, 2008.

[79] D. Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *ACM SIGCOMM Computer Communication Review*, volume 34, pages 367–378. ACM, 2004.

[80] N. Ramakrishnan, B.J. Keller, B.J. Mirza, A.Y. Grama, and G. Karypis. Privacy risks in recommender systems. *IEEE Internet Computing*, 5(6) :54–62, 2001.

[81] S. Ren, E. Tan, T. Luo, S. Chen, L. Guo, and X. Zhang. Topbt : a topology-aware and infrastructure-independent bittorrent client. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[82] I. Rimac, V. Hilt, M. Tomsu, V. Gurbani, and E. Marocco. A Survey on Research on the Application-Layer Traffic Optimization (ALTO) Problem. RFC 6029 (Informational), October 2010.

[83] P. Rodriguez, S.M. Tan, and C. Gkantsidis. On the feasibility of commercial, legal p2p content distribution. *ACM SIGCOMM Computer Communication Review*, 36(1) :75–78, 2006.

[84] J. Rosenberg. Interactive Connectivity Establishment (ICE) : A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245 (Proposed Standard), April 2010. Updated by RFC 6336.

[85] K.W. Ross. Unraveling the bittorrent ecosystem.

[86] A. Rowstron and P. Druschel. Pastry : Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware 2001*, pages 329–350. Springer, 2001.

[87] H. Schulze and K. Mochalski. ipoque internet study 2008/2009. *ipoque. com*, 2009.

[88] S. Shalunov and G. Hazel. Low extra delay background transport (ledbat). *IETF Draft (March 2009)*, 2010.

[89] P. Srisuresh, B. Ford, and D. Kegel. State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs). RFC 5128 (Informational), March 2008.

[90] P. Srisuresh and M. Holdrege. IP Network Address Translator (NAT) Terminology and Considerations. RFC 2663 (Informational), August 1999.

[91] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord : A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 31(4) :149–160, 2001.

[92] R. Suselbeck, G. Schiele, P. Komarnicki, and C. Becker. Efficient bandwidth estimation for peer-to-peer systems. In *Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on*, pages 10–19. IEEE, 2011.

[93] J. Terrace, H. Laidlaw, H.E. Liu, S. Stern, and M.J. Freedman. Bringing p2p to the web : Security and privacy in the firecoral network. In *Proceedings of the 8th international conference on Peer-to-peer systems*, pages 7–7. USENIX Association, 2009.

[94] Johan Ugander, Brian Karrer, Lars Backstrom, Cameron Marlow, and Cameron Marlow. The anatomy of the facebook social graph. 2011.

[95] M. Varvello, M. Steiner, and K. Laevens. Understanding bittorrent : A reality check from the isp's perspective. *Computer Networks*, 2011.

[96] S. Wolchok and J.A. Halderman. Crawling bittorrent dhts for fun and profit. *Proc. of WOOT (Washington, DC, USA, 2010)*, 2010.

[97] D. Wu, P. Dhungel, X. Hei, C. Zhang, and K.W. Ross. Understanding peer exchange in bittorrent systems. In *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*, pages 1–8. IEEE, 2010.

[98] H. Xie, Y.R. Yang, A. Krishnamurthy, Y.G. Liu, and A. Silberschatz. P4p : provider portal for applications. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 351–362. ACM, 2008.

[99] D. Xu, S.S. Kulkarni, C. Rosenberg, and H.K. Chai. Analysis of a cdn–p2p hybrid architecture for cost-effective streaming media distribution. *Multimedia Systems*, 11(4) :383–399, 2006.

[100] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li. Design and deployment of a hybrid cdn-p2p system for live video streaming : experiences with livesky. In *Proceedings of the seventeen ACM international conference on Multimedia*, pages 25–34. ACM, 2009.

[101] X. Zhang, J. Liu, B. Li, and Y.S.P. Yum. Coolstreaming/donet : a data-driven overlay network for peer-to-peer live media streaming. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 2102–2111. IEEE, 2005.