Projet recherche: Etat de l'art

BECART Quentin, FOULON Vincent,
GRZELAK Gaël, GOURMELON Gwendal,

Octobre 2018

1 Introduction

Vitesse, confiance et portabilité sont les principaux enjeux du web aujourd'hui. Le public a besoin de tout, tout de suite et sur tous leurs objets connectés. Des solutions commencent à répondre à certains de ces besoins mais peu répondent à tous efficacement. De plus, ces solutions ne répondent pas toutes à d'autres problèmes qui commencent à émerger : la sécurité des données ainsi que la propriété des données personnelles. Il sera également intéressant d'explorer ces nouvelles problématiques afin de savoir les améliorations possibles du système.

2 Analyse de l'existant

Dans cette partie, nous allons d'abord décrire brièvement les différentes offres de stockage de données disponibles sur le marché. L'objectif est de pouvoir analyser le marché actuel, découvrir les technologies utilisées, leur popularité, et prendre un instantané des offres présentes.

2.1 Google Drive

Google Drive est un service de synchronisation et d'hébergement de fichiers proposé par Google. Il permet d'accéder à ses données de manière synchronisée sur n'importe quelle machine et n'importe quelle plateforme, simplement en se connectant sur son compte Google[1].

Il permet également de partager des fichiers (les fichiers texte, les tableurs et les diaporamas par exemple) avec d'autres utilisateurs, chaque utilisateur pouvant modifier le fichier en temps réel. Une gestion des droits avancée est aussi possible (lecture seule, lecture écriture ou fichier inaccessible). Des applications ajoutant des fonctionnalités existent également. "Backup and Sync", par exemple, est un plugin permettant une synchronisation entre un périphérique (un dossier présent sur une machine) et le drive en temps réel également.

Au niveau tarifaire, Google propose un hébergement gratuit jusqu'à 15 Go de stockage puis un tarif progressif (100Go pour 2€ par mois, 1To pour 10€ et 10To pour 100€)[2]. **Avantages:**

- Synchronisation et hébergement de fichiers en temps réel
- Très facile d'accès
- Possibilité d'éditer un fichier à plusieurs en même temps
- Gestion des droits de manière avancée
- Accessible depuis toutes les plateformes
- Gratuit jusqu'à 15 Go

Inconvénients:

- Payant pour un espace de stockage plus important
- Le débit est bridé volontairement par les serveurs de Google, que cela soit en download ou en upload
- Les données sont stockées sur les serveurs de Google
- La personnalisation est limitée
- Le logiciel n'est pas adapté pour de la synchronisation de code dans le cadre d'un projet de développement

Figure 1: Logo de Google Drive

2.2 Git

Git est un outil de versionning très populaire dans le monde du développement[3]. Il est principalement utilisé avec un serveur centralisé qui fait office de "remote". Le remote est la référence d'un fichier. Lorsqu'un utilisateur a besoin de faire des modifications sur ce fichier, il télécharge le fichier présent sur le serveur, fait la modification en local et git va pouvoir traquer les modifications réalisées en local mais aussi les modifications réalisées par le remote pendant que le développeur réalise des modifications. Après cela, git utilise des outils externes comme *vimdiff* afin de permettre à l'utilisateur de résoudre les potentiels conflits entre le remote et le local[4][5].

L'outil est par essence asynchrone car il est nécessaire d'exécuter des commandes afin de partager et recevoir des documents. Il existe également des

outils de partage comme *git daemon*, qui permet à d'autres utilisateurs de se connecter au dépôt. Cela permet d'éviter de passer par internet et des entreprises d'hébergement git externes (Github[6], Gitlab[7], ...).

Git peut utiliser le protocole SSH pour établir des connexions sécurisées tout en rendant l'outil plus facile d'utilisation.

Figure 2: Logo de Git

2.3 Subversion

Subversion est un outil de versionning similaire à Git. Son utilisation est plus simple mais en contrepartie offre moins de fonctionnalité. Historiquement, SVN était plus utilisé que Git grâce à la possibilité d'utiliser le protocole SSH[8]. SVN était plus sécurisé, surtout dans le domaine de l'entreprise. Mais après que Git ait ajouté ces fonctionnalités, SVN a perdu en popularité et dispose maintenant d'une plus petite communauté.

Le serveur Subversion est en revanche très simple à déployer[9]. Les clients graphiques permettent de voir l'état de la copie locale directement dans l'explorateur de fichier. De nombreux clients existent, tels que tortoiseSVN[10] ou rapidSVN[11].

Figure 3: Logo de Subversion

2.4 Resilio Sync

Resilio Sync est un outil payant qui permet un partage de fichier ou de contenu entre machines, en P2P. Depuis une application cross-platform, l'utilisateur peut choisir les dossiers dont il souhaite partager une information. Cet outil propose 3 solutions, Team, Enterprise et Home en fonction de l'usage de l'utilisateur. Resilio utilise un protocole BitTorrent modifié qui lui permet en effet d'améliorer le débit de transfert des informations, jusqu'à 40MBps pour chaque pair dans un réseau de 50 pairs.
Cependant, il existe des limitations à l'outil au niveau du partage de la bande passante. Tous les pairs n'ont pas accès à la même quantité de bande passante, ce qui rend certains pairs très lents à synchroniser.
[12] [13]

Figure 4: Logo de Resilio

2.5 Emule

Emule est un système de partage de fichier public et un des ancêtres du partage torrent moderne. Exploitant le protocole Gnutella, il possède en 2007 plus de 20 millions d'utilisateurs. Ce qui pour un système basé sur le pair-à-pair, représente un importante quantité de ressources et le rend donc excellent pour trouver des fichiers rares. eMule gère l'échange de sources entre les différents clients via le réseau Kad Network. Ce réseau est propre d'eMule, eMule se connectait jusqu'alors sur le réseau eDonkey. Kad Network implémente le protocole Kademlia et à l'avantage de pouvoir se passer de serveurs de connexion et ainsi réduire la charge sur les serveurs et leur importance. L'implémentation de Kademlia permet la mise en place d'une "Distributed Hash Table" qui rend les échanges de servers tiers indépendants et augmente aussi la vitesse du système. Cependant, le réseau n'est pas accessible seul. Il est nécessaire de connaître au moins une node du réseau afin d'avoir accès aux fichiers partagés. Typiquement, un serveur de découverte propose plusieurs "portes d'entrée" du réseau.

Parmi les avantages que eMule propose, nous pouvons parler du débit très important, lorsqu'il y a beaucoup de pairs, limité en général par notre propre connexion. Le nombre d'utilisateurs de la plateforme permet une répartition homogène du contenu partageable. eMule pose néanmoins des problèmes à l'utilisation : le logiciel est vieillissant, il y a donc de moins en moins de pairs. Un système de file est également en place. Il faut remonter la file avant de pouvoir télécharger un fichier. [14] [15] [16]

Figure 5: Logo d'Emule

2.6 SyncThing

SyncThing est un outil gratuit open-source permettant de faire de la synchronisation P2P de dossiers entre appareils connectés. Il est portable sur

Linux, Windows, Android, MacOS. Ses fonctionnalités répondent assez bien aux exigences que nous nous nous sommes posées en début de projet. Le projet implémente un protocole personnalisé permettant à l'application de complètement garder la main sur l'information. Cependant, ce protocole limite la taille maximum des fichiers à 12GB. Pour des raisons de sécurité, les utilisateurs doivent explicitement accepter une node dans le réseau P2P. Les données envoyées sont également encryptés. Ces deux fonctionnalités permettent la création d'un réseau isolé et anonyme. L'application possède trois serveurs permettant la découverte des pairs et ainsi la mise en relation des appareils. Cette fonctionnalité est améliorable dans le cadre de notre projet car il existe aujourd'hui d'autres manières de découvrir des pairs de manière sécurisée.

[17] [18]

Figure 6: Logo de SyncThing

2.7 Dropbox

Dropbox est le système de cloud storage le plus utilisé dans le monde et représente déjà une partie conséquente du trafic mondial. Dropbox permet de synchroniser des fichiers entre les différents périphériques et le cloud Dropbox. Une fois installée, l'application basée sur RSync crée une nouvelle couche d'abstraction au navigateur de fichier du système pour que l'utilisateur sache exactement si son fichier est synchronisé ou non. Il conserve un historique de fichiers pendant 30 jours. De €12 à €10/mois pour 3To De €18 à €15/mois pour un stockage illimité Développé largement en Python et grâce à des librairies comme librsync[35], il implémente efficacement le partage de fichier de petites tailles. Bien qu'efficace en terme d'allocation de bande passante , le débit de téléchargement est très impacté par la distance entre le client et le serveur[35].

2.8 Mega

Les informations relatives à Mega (articles) datent pour la plupart de 2013, soit peu après son lancement. Les informations des utilisateurs ne seraient pas à l'abri, celles-ci étant apparemment stockées et traitées de manière non chiffrée, contrairement à ce qui est indiqué sur la page d'accueil du site.

On rentre avec Mega dans certaines questions éthiques, notamment l'utilisation des données : leur CGU ayant fait polémique concernant les droits des utilisateurs et la politique vis à vis des données personnelles de ces derniers.

Mega a l'avantage de proposer un service permettant d'avoir une vitesse d'Upload illimitée, il n'y a également aucun temps d'attente avant un téléchargement et aucune publicité de viendrait perturber la navigation. Mega permet également aux utilisateurs d'effectuer plusieurs téléchargements simultanés et offre pas moins de 50Go d'espace de stockage.

2.9 RSync

Rsync est une commande de synchronisation sous licence GNU GPL disponible sur Unix puis par la suite disponible sur les autres systèmes (Mac et Windows). Elle a été développée par Andrew Tridgell dans les années 2000 pour donner suite à ses travaux sur la compression et le format zip. Caractéristiques

- Communique via le protocole SSH (sécurisé par échange de clé).
- Transfère uniquement la partie manquante des fichiers grâce à un checksum.
- Vérifie les fichiers qui seront modifiés après exécution (donc les différences entre 2 répertoires).
- Compresse les fichiers avant envoi pour amoindrir l'utilisation de la bande passante.

RSync est unidirectionnel, ce qui implique que les modifications apportées au répertoire A vont être imputées au répertoire B, mais les modifications effectuées dans B seront immédiatement ré-écrasées lors de la prochaine synchronisation. Il existe également Grsync et Duplicati qui sont en licence libre et probablement unidirectionnels. Unison en revanche, en licence GNU GPL, a l'avantage de pouvoir effectuer de la synchronisation bidirectionnelle. Les dates de modifications sont alors comparées pour savoir quelle version du fichier sera conservée. Chaque client est également serveur, tout le monde peut lancer la commande de synchronisation. La principale difficulté dans ce mode de fonctionnement réside dans le fait qu'un fichier peut être modifié en même temps sur plusieurs sites. En effet, dans ce cas, nous sommes rapidement confronté à de la perte de données si le fichier synchronisé était en cours de modification. [19] [20] [21]

3 Protocoles

3.1 IPFS

IPFS est un protocole de partage de fichier open source en P2P. L'objectif du projet est que le maximum d'appareil soient connecté au réseau IPFS. Ainsi, le réseau serait un espace d'échange mondial entre appareils se partageant des informations (source wikipédia). Quand on se connecte, on télécharge le contenu partagé et on ouvre un port pour que d'autres personnes puissent télécharger ce même contenu plus rapidement.

L'outil est utilisable et intéressant. Peu populaire auprès du grand public, les utilisateurs principaux sont des professionnels du domaine ou des personnes ayant des connaissances en informatique.

L'outil est aussi beaucoup utilisé dans les blockchains pour réduire les coûts de stockage de fichier. En effet, les transactions ne peuvent excéder une certaine taille. Si la transaction est décentralisée, on a plusieurs petites transactions. IPFS est complètement décentralisé grâce à l'utilisation de "Distributed

Hash Table" mais aussi parce qu'aucun pair n'est privilégié pendant les transferts. Divisé en sept modules, il regroupe un ensemble d'outils afin de pouvoir administrer la totalité du réseau. [22] [23]

Figure 7: Logo du protocole IPFS

3.2 FTP

Le "File Transfer Protocol" permet, comme son nom l'indique, de faire du transfert de fichier. Ce protocole utilise le réseau TCP/IP pour partager des fichiers[24].

Ce protocole permet de copier des fichiers depuis un ordinateur vers un autre ordinateur du réseau en utilisant le modèle client-serveur. [25]

3.3 TCP

Ce protocole permet d'envoyer des données tout en vérifiant que ces données ont bien été reçues grâce à un accusé de réception envoyé par le récepteur. Si les données n'ont pas bien été réceptionnées, l'émetteur effectue un nouvel envoi des trames qui ne sont pas passées.

3.4 Bittorrent

Protocole de transfert de données pair-à-pair. Il s'agit d'une méthode pour transférer une grande quantité de données en répartissant les charges inhérentes au transfert (bande passante, hébergement, puissance de la machine). Cela permet, contrairement à d'autres méthodes de transfert, de réduire la charge du distributeur originel qui n'est plus le seul à fournir les données : les clients eux-mêmes servent les données déjà reçues aux nouveaux utilisateurs.

Bittorent utilise également un système de "Tit-For-Tat", ou "Donnant-Donnant", qui rend le système plus paritaire. Ce système incite les pairs à partager leurs fichiers en récompensant les pairs offrant plus de bande passante montante et en pénalisant les pairs qui n'utilisent que la bande passante descendante.

Figure 8: Logo du protocole Bittorrent

3.5 Gnutella

Protocole décentralisé de recherche et de transfert de fichiers en pair-à-pair, utilisé par eMule par exemple. Il s'agirait du premier réseau pair-à-pair décentralisé qui ait existé. Pour fonctionner, il envoie une requête pour rechercher un fichier aux pairs connus, la requête est alors propagée entre les machines de pair-à-pair jusqu'à ce que le fichier soit trouvé, puis la machine qui possède le fichier l'envoie en http à la machine qui lui a demandé.

Figure 9: Un logo proposé pour le protocole Gnutella

4 Comparatif des solutions cloud and synchronised storage

	Espace de stockage gratuit	Prix/mois	Taille de fichiers	Compatibilité	Historique des fichiers	Option de sauvegarde	Chiffrement stockage/transferts	Pays
Dropbox	2 Go (extensible à 20 Go par parrainages et astuces)	8,25 $/1 To 12,50 $/2 To 20 $/Illimité	Illimité (10 Go via interface Web)	Windows, macOS, Linux, Android, iOS, Kindle Fire, Windows Phone	30 jours	Non	Oui	USA
Box	10 Go	8 €/100 Go	250 Mo (gratuite) 5 Go (payante)	Windows, macOS, Android, iOS	Oui (compte payant uniquement)	Non	Oui	USA
Google Drive	15 Go	1,99 €/100 Go 9,99 €/1 To 99,99 $/10 To	5 To	Windows, macOS, Android, iOS	30 jours	Oui	Oui	USA
Microsoft OneDrive	5 Go	2 €/50 Go 7 €/1 To (+ Office 365) 10 €/5 To (+ Office 365 pour 5 utilisateurs)	15 Go	Windows, macOS, Android, iOS, Windows Phone	25 versions précédentes	Non	Transferts uniquement	USA
Amazon Cloud Drive	5 Go	1 $/100 Go 5$/1 To	Illimitée	Windows, macOS, Android, iOS	Non	Non	Transferts uniquement	USA
Apple ICloud	5 Go	0,99 €/50 Go 2,99 €/200 Go 9,99 €/2 To	Illimitée	Windows, macOS, iOS	Non	Terminaux iOS uniquement	Oui	USA
HubiC	25 Go	1 €/100 Go 5 €/10 To	Illimitée	Windows, macOS, Linux, Android, iOS, Windows Phone 8	Versions précédentes (illimité)	Oui	Oui	France

[26] [27]

5 Partage / Streaming

Utilisé principalement pour l'envoi de contenu sous forme de flux. Les données sont téléchargées en continu dans la mémoire vive et rapidement transférées vers un écran ou un lecteur multimédia (pour lecture) puis remplacées par de nouvelles données.

- On peut diffuser un périphérique d'entrée en direct ou diffuser un fichier.
- Dans le cas d'un périphérique d'entrée (caméra, micro...), on parle de streaming en live (direct). Le serveur va créer un ou plusieurs flux de qualité différente puis un grand nombre de clients va se connecter à l'un de ces flux en fonction de sa bande passante. Dans ce cas le serveur travaille à flux tendu, et les ressources de la machine doivent être adaptées puisque celle-ci va effectuer plusieurs compressions à la volée. La notion de latence devient à ce moment un réglage primordial pour trouver un bon équilibre entre qualité et fluidité.
- Dans le cas d'un fichier, on peut directement traiter le fichier en amont (à l'upload par exemple, comme YouTube). Ainsi, le client va se connecter au flux d'un fichier qui aura déjà été compressé. La machine sera donc beaucoup moins sollicitée.

[28] [29]

6 Modèles de réseau informatique

6.1 Pair-à-Pair

6.1.1 Architectures

Il existe deux types de structures pair-à-pair : les structures possédant un groupe serveur centralisé permettant la découverte de pairs et de fichiers et les structures complètement décentralisées possédant une "Distributed Hash Table" afin de découvrir les pairs. Dans les deux cas, les téléchargements se font directement entre les pairs découverts et l'objectif est de faciliter le partage de données entre individus d'un groupe.

Dans une architecture possédant des serveurs centralisés, comme Napster ou eDonkey, le réseau fonctionne en deux étapes. Tout d'abord, le nouveau pair s'authentifie au serveur, lui dit les fichiers qu'il est prêt à partager ainsi que quelques métadonnées d'utilisateurs. Ensuite, le pair fait une requête HTTPS-like au serveur avec en contenu, entre autres, le hash du fichier que le client souhaite télécharger. Le serveur retourne la liste des pairs possédant ce fichier et par la suite, les pairs se connectent entre eux et le partage commence.

Dans une architecture complètement décentralisée, comme Gnutella, la découverte de pairs se fait par inondation du réseau. Un nouveau pair construit une requête demandant à quiconque recevant celle-ci de lui envoyer un message.

Cette requête est ensuite retransmise de voisin à voisin par chaque pair atteint. Cette méthode réduit les coûts d'entretien du système mais impose un flux constant de requêtes sur le réseau.

Ce système engendre des asymétries dans le réseau. En effet, sur un réseau P2P, la demande de bande passante descendante est beaucoup plus élevée que l'offre de bande passante montante. [38]

6.1.2 Analyse

Pour résoudre ce problème, beaucoup de réseaux P2P publics ont instauré un système de "donnant-donnant" : plus un pair partage de fichiers, plus il est prioritaire quand il télécharge un fichier.[30] Ainsi, les architectures P2P proposent un service efficace de partage de données collaboratif. Les coûts de stockage le prix de la bande passante est réduit. L'aspect collaboratif est l'aspect le plus discutable du système car il est nécessaire d'avoir un bon degré de réplication sur le réseau afin d'offrir une qualité de service correcte. Nous dépendons également des pairs et de leurs données. Nous n'avons donc aucune assurance de la provenance des données, nous sommes donc vulnérables aux virus. Bien que ce problème soit également présent lors de stockage cloud classique, il est amplifié par la dualité client/serveur de chaque utilisateur.

Avantages:

- Réduction du coût de bande passante pour le distributeur d'origine
- Partage de fichier plus rapide grâce à un envoi de fichiers multiples depuis différents fournisseurs
- Si une machine tombe en panne, cela ne remet pas en cause l'intégrité du système

Inconvénients:

- Risque d'instabilité. Par exemple: une modification se propage alors qu'une autre, plus importante n'a pas encore été prise en compte. (pair déconnecté par exemple)
- Modification non souhaitée ou suppression accidentelle provoquant la disparition de fichiers.
- Risque de conflits plus complexes à gérer
- Hausse de la consommation de bande passante en "upload"
- Risque de saturation de la bande passante pour les mauvaises connexions en cas de mauvais paramétrage

6.2 Centralisé

Le stockage centralisé de données est couramment utilisé car il est plus simple de gérer l'état d'un fichier depuis une source unique. Ce type de fonctionnement comporte de grands avantages mais ces avantages ont un prix. En effet, le stockage centralisé est plus sûr d'utilisation car on ne dépend pas de pairs extérieurs, on gère directement la sécurité et les droits depuis le serveur. D'autre part, l'état global des fichiers est directement défini par le serveur, si celui-ci dispose d'un nouveau fichier, d'un fichier modifié ou supprimé, les clients n'ont qu'à se baser sur cette unique référence ce qui garantit un système toujours très stable. Le problème vient de la scalabilité. Lorsque l'on double le nombre d'utilisateurs connectés sur le serveur, on doit également doubler la quantité de bande passante pour garder un débit client similaire. En P2P, lorsqu'on double le nombre d'utilisateurs connectés, on diminue le débit nécessaire au partage de données (plus d'utilisateurs connectés = plus de pairs = plus de débit). Avoir un serveur centralisé est donc plus simple car on a besoin de ne connaître qu'une seule machine pour accéder à l'information. Alors qu'en P2P, on doit entretenir des listes (décentralisées ou non) qui contiennent les pairs possédant l'information que l'on cherche.

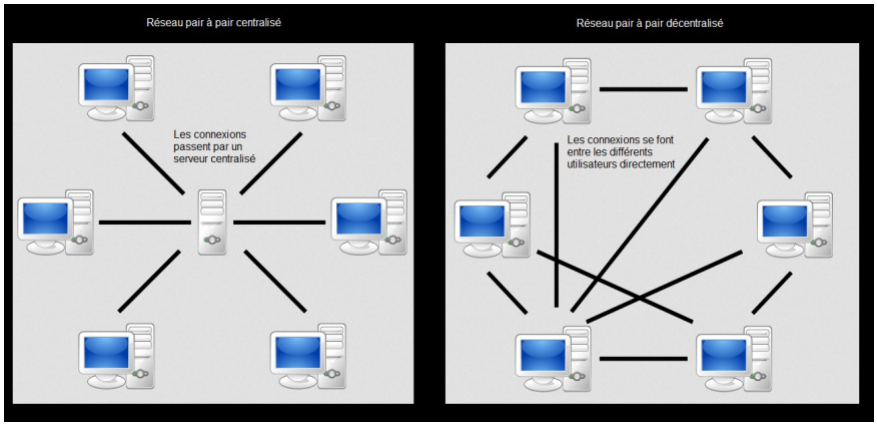

Figure 10: Modèle serveur-client vs. modèle pair-à-pair

7 Publications

7.1 Mahadev Satyanarayanan

Mahadev Satyanarayanan est un chercheur en informatique, pionnier en Edge Computing (méthode d'optimisation de Cloud Computing), informatique mobile, informatique ubiquitaire et IoT (internet des objets). C'est le créateur du systèmes de fichiers Andrew File System. Il a travaillé dans de nombreux domaines liés au partage de fichiers via l'internet. Il a passé un master en Inde

puis son doctorat aux Etats-Unis, à l'université de Carnegie Mellon[31]. Il s'agit d'un des acteurs majeurs dans le domaine des systèmes de fichiers[32].

En lien avec notre sujet, nous avons étudié une de ses publications traitant du Coda File System, le successeur du Andrew File System mais qui s'intéresse à l'internet mobile. Cette publication est la suivante :

> Coda: A Highly Available File System for a Distributed Workstation Environment [37]

Cette publication est directement liée à notre sujet puisqu'elle traite d'un protocole de stockage et de partage de fichier. De plus, Coda est spécialisé dans l'informatique mobile, et il s'agit de l'un des points clés de nos objectifs : la portabilité sur de nombreux supports, y compris le mobile.

Coda est donc un système de fichier pouvant être utilisé par un grand nombre de machines utilisant un système UNIX. Coda se veut disponible en toute circonstance. Pour cela, il utilise deux méthodes :

- Une méthode appelée *server replication* qui permet, comme son nom l'indique, de conserver des copies d'un fichier sur plusieurs serveur. Cette notion se rapproche de l'utilisation du pair-à-pair : chaque client conserve une version du fichier, ce qui permet de toujours garder une sauvegarde du fichier quelque part. Cependant, Coda utilise un système centralisé classique, à la seule différence qu'ici les serveurs et donc les sauvegardes sont multiples. Cette méthode permet donc de toujours avoir accès au fichier, et ce, même si un voire plusieurs serveurs sont victimes de problèmes techniques.

- Une méthode appelée *disconnected operation* permet de conserver temporairement en cache un site qui assume le rôle de réplique du site original. Cette notion est particulièrement importante pour les supports mobiles, n'ayant pas forcément toujours accès à une connexion internet.

Coda a été construit autour d'une idée principale : la possibilité d'accéder aux données en permanence. Cette idée est également un point central pour notre projet, et l'une des principales problématiques du pair-à-pair : comment accéder aux données lorsqu'aucun autre pair n'est connecté?

7.2 Arnaud Legout

Chercheur scientifique à l'INRIA (Institut national de recherche en informatique et en automatique). Diplômé Docteur des Sciences (PhD) à l'Université de Nice-Sophia Antipolis, il a effectué sa thèse de recherche à l'Institut Eurecom (école d'ingénieurs, Sophia Antipolis) sur les travaux de Ernst Biersack concernant le contrôle de la congestion multicast [33]. Il a préalablement obtenu sa Maîtrise de Mathématiques dans la même université. Il a effectué de nombreux

travaux, entre autres :

Actuels :

- *ElectroSmart*: Mise en lumière de notre exposition aux ondes électromagnétiques.
- *soTweet*: Propagation de l'information dans les réseaux sociaux.

Passés :

- *Streaming vidéo* : Quelles sont les caractéristiques du streaming vidéo ?
- *Bluebear*: Explorer les menaces sur la vie privée sur Internet.
- *P2P-CD*: Comprendre et améliorer la distribution de contenu P2P.

Relativement au sujet que nous traitons à travers ce document, Arnaud Legout est le co-auteur de nombreuses publications dont une, traitant de Bittorrent, s'intitulant :

Understanding Bittorrent : An Experimental Perspective [34]

Ce document nous explique notamment le principe des deux algorithmes les plus importants de Bittorrent, à savoir le *choke algorithm* ou encore le *rarest first piece selection algorithm*.

Premièrement, le *choke algorithm* a été introduit pour garantir une certaine réciprocité dans l'*upload* et le *download*. C'est-à-dire que cet algorithme va pénaliser les pairs qui ne transmettent pas assez (dont l'upload n'est pas suffisant par rapport au download), tout ceci dans le but d'éviter que certains pairs se comportent comme des sangsues : en prenant mais en ne donnant très peu ou rien.

En second, le *rarest first piece selection algorithm*, qui pourrait se traduire littéralement par *algorithme de sélection du premier morceau le plus rare* est très simple et pratiquement compréhensible de par son nom : Le pair local garde en mémoire le nombre de copies de chaque morceau de contenu. Cette information permet de déterminer quels morceaux ont le moins grand nombre de copies et donc identifie un ensemble de morceaux étant les plus rares. Cet ensemble évolue en fonction du nombre de copies ajoutées ou supprimées et redéfinit donc régulièrement l'ensemble des morceaux les plus rares.

Arnaud Legout a donc effectué des expériences concernant ces algorithmes, en mettant en place un client Bittorrent et en faisant varier diverses caractéristiques. Le résultat de ces expériences et donc la conclusion de ce papier sont :

- Le premier algorithme que nous avons vu permet une répartition équitable de la quantité de contenu entre les différents clients et permet également à un pair pris au hasard d'avoir les mêmes chances qu'un autre d'être servi par un pair donné.
- Le second algorithme permet quand à lui d'améliorer la diversité des informations et ainsi de diminuer la rareté d'un morceau de contenu.

7.3 Hamid Reza Faragardi

Chercheur à la KTH, Université Royal de Technologie, il a fait des recherches principalement dans le cloud computing et le IoT.
S'intéressant au cloud computing, aux systèmes en temps réel et à l'optimisation, il a écrit beaucoup de publications récemment sur l'optimisation des systèmes cloud.

Sa publication "Ethical Considerations in Cloud Computing Systems "[36] fait un tour d'horizon sur les problèmes éthiques du cloud et la provenance de ces problèmes. Les différentes technologies d'hébergement disponibles, *Software-as-a-Service*, *Infrastructure-as-a-service* et *Platforme-as-a-service* apportent chacun leurs lots de difficultés. Ces difficultés se déclinent en quatre sujets :

- La confidentialité et la sécurité : nous ne pouvons jamais savoir si nos données ont été piratées car nous dépendons d'un tiers pour sécuriser nos données. *A noter que l'Europe a depuis abordé ce problème grâce à la mise en place des RGPD*
- La conformité : la sécurité doit être assurée en respectant des standards communs
- Les performances : lorsque l'utilisateur utilise un service cloud, il signe un contrat de niveau de service. Cependant, si les performances du service diminuent de manière non mesurable par l'utilisateur, ce contrat ne s'applique pas. Si, de ces baisses mineurs de qualité de service, l'hébergeur gagne de l'argent sans être puni, nous pouvons considérer cela comme un problème éthique.
- L'environnement : les data-centers stockant les données de l'utilisateur consomment une part non négligeable de la production globale d'électricité. Aucune régulation sur la consommation d'électricité n'est actuellement en place.

Par la suite, l'auteur explique que les règles sont définies par les fournisseurs dans leurs *Terms&Conditions*. Ces règles sont floues et définies par le fournisseur lui même, elles ne sont donc pas dans l'intérêt commun. Il est donc nécessaire de définir un jeu de règles commun.

8 Limites de l'existant

8.1 Technologiques

La plupart des logiciels que nous avons évoqués utilisent un modèle de réseau centralisé. Ce modèle a plusieurs inconvénients. Par exemple, l'absence de stockage local rend impossible l'accès aux informations sans une connexion internet.

La bande passante est également limitée lors d'une utilisation simultanée par un grand nombre de clients.

D'autre part, une grande partie des logiciels que nous avons étudiée ne sont pas libres de droits, ce qui limite beaucoup la personnalisation de la part de l'utilisateur.

8.2 Ethiques

Le stockage centralisé entraine également des questions éthiques voire politiques. En effet, de nos jours, de plus en plus de personnes souhaitent avoir la main-mise sur leurs informations et n'apprécient pas que leurs données soient stockées par les géants du web (notion d'anonymat et d'informations personnelles voire confidentielles).

D'autre part, même s'il y a un chiffrement des données, les informations peuvent rester vulnérables si les datacenters sont victimes d'une attaque.

De plus, certains cloud analysent le contenu de leurs clients pour améliorer les résultats des moteurs de recherche, fournir de la publicité appropriée...

8.3 Financières

De manière générale, les logiciels étudiés sont payants à partir d'une certaine limite de stockage et/ou de bande passante. Ces limites et ces offres sont dûes aux besoins matériels pour le fournisseur des serveurs.

Des utilisateurs avec des besoins importants mais sans moyens financiers ou ne souhaitant pas payer pour ce type de service peuvent ne pas être satisfaits par ces types de logiciels.

9 Objectifs du projet

Ce projet a pour objectif de développer un système de partage de fichiers versionné, c'est-à-dire pour lequel on peut récupérer le fichier tel qu'il était à un moment donné, efficace en ressources. L'application fonctionnerait sur une topologie de type Pair-à-Pair (P2P), optimisant ainsi les flux de données et l'accessibilité. Le P2P permettra de fragmenter le point d'échec tout en réduisant les coûts globaux.

La synchronisation se ferait de manière automatique et totalement transparente. L'utilisateur n'a pas besoin de saisir de commande pour se synchroniser avec les autres. L'outil intégrera un système de résolution de conflit, pour les fichiers textes dans un premier temps, puis avec d'autres types de fichiers par la suite. Des outils open-sources comme r-sync sont dans ces cas utiles et seront abordées.

Nous voudrions également proposer d'autres fonctionnalités comme du streaming multi-flux. Par exemple, un groupe de musiciens, auraient la possibilité de jouer chacun de leur côté un morceau en direct. Ce flux serait visualisable directement en streaming depuis l'application ou sur le web.

Un autre objectif serait de créer une application très simple d'utilisation, qui ne demande aucune compétence informatique particulière. Il suffirait de choisir le dossier que l'on souhaite synchroniser avec le dossier/l'adresse cible, sans autre réglage pour un fonctionnement basique, mais avec de nombreuses possibilités pour un fonctionnement plus spécifique.

Enfin, nous devons permettre à l'application d'être le plus portable possible. La portabilité est une nécessité pour séduire un public diverse mais également

permettra de rendre plus efficace l'application P2P : plus de plateformes supportées signifie plus de pairs disponibles.

Avec ces objectifs réalisés, nous souhaitons obtenir une solution satisfaisant les trois points exprimés dans l'introduction et qui réponde à la question suivante:

Comment répondre aux problématiques de densification des réseaux?

References

[1] https://fr.wikipedia.org/wiki/Google_Drive.

[2] https://www.google.com/intl/fr/drive/pricing/.

[3] https://fr.wikipedia.org/wiki/Git.

[4] https://doc.ubuntu-fr.org/git.

[5] https://git-scm.com/.

[6] https://github.com/.

[7] https://about.gitlab.com/.

[8] https://fr.wikipedia.org/wiki/Apache_Subversion.

[9] https://doc.ubuntu-fr.org/subversion.

[10] https://tortoisesvn.net/.

[11] https://en.wikipedia.org/wiki/RapidSVN.

[12] https://ieeexplore.ieee.org/abstract/document/7969177.

[13] https://sci-hub.tw/https://ieeexplore.ieee.org/abstract/document/7969177.

[14] https://fr.wikipedia.org/wiki/EMule.

[15] https://dl.acm.org/citation.cfm?id=1460907.

[16] https://en.wikipedia.org/wiki/Kad$_n etwork$.

[17] https://dl.acm.org/citation.cfm?id=1460907.

[18] https://forum.syncthing.net/t/limits-of-syncthing/134/9.

[19] https://www.andrew.cmu.edu/course/15-749/READINGS/required/cas/tridgell96.pdf.

[20] https://doc.ubuntu-fr.org/rsync.

[21] https://fr.wikipedia.org/wiki/Rsync.

[22] https://en.wikipedia.org/wiki/InterPlanetary$_{F}ile_{S}ystem$.

[23] https://arxiv.org/abs/1407.3561.

[24] https://fr.wikipedia.org/wiki/File$_{T}ransfer_{P}rotocol$.

[25] https://www.rfc-editor.org/rfc/rfc959.txt.

[26] https://www.lesnumeriques.com/informatique/quel-service-cloud-choisir-a3277.html.

[27] https://www.capital.fr/lifestyle/dropbox-google-drive-icloud-6-solutions-cloud-au-banc-dessai-1263408.

[28] https://fr.wikipedia.org/wiki/StreamingPrincipe_de_fonctionnement.

[29] https://doc.ubuntu-fr.org/streaming.

[30] http://www.bittorrent.org/bittorrentecon.pdf.

[31] https://en.wikipedia.org/wiki/Mahadev_Satyanarayanan.

[32] https://fr.wikipedia.org/wiki/Système_de_fichier.

[33] http://www.eurecom.fr/fr/publication/1041/download/ce-urvogu-021023.pdf.

[34] https://hal.inria.fr/inria-00000156v3/document.

[35] Inside dropbox:understanding personal cloud storage services. https://dl.acm.org/citation.cfm?id=2398827.

[36] Hamid Reza Faragardi. Ethical considerations in cloud computing systems. www.mdpi.com/2504-3900/1/3/166/pdf.

[37] Puneet KUMAR Maria E. OKASAKI Ellen H. SIEGEL David C. STEERE Mahadev SATYANARAYANAN, James J. KISTLER. Coda: A highly available file system for a distributed workstation environment. https://www.cs.cmu.edu/ satya/docdir/satya-ieeetc-coda-1990.pdf.

[38] Steven D. Gribble Stefan Saroiu, P. Krishna Gummadi. A measurement study of peer-to-peer file sharing systems. https://people.mpi-sws.org/ gummadi/papers/p2ptechreport.pdf.