RFC 6972 : Problem Statement and Requirements of Peer-to-Peer Streaming Protocol (PPSP)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 20 juillet 2013

Date de publication du RFC : Juillet 2013

https://www.bortzmeyer.org/6972.html

Il existe actuellement des tas de logiciels qui font du "*streaming*" (audio ou vidéo) en pair à pair mais toujours en utilisant des protocoles privés et fermés. Cela a des tas de conséquences négatives, notamment sur le plan du choix : l'utilisateur est enfermé dans l'offre d'une compagnie particulière. Il est donc urgent de développer un protocole libre et ouvert de "*streaming*" pair à pair et c'est la tâche du groupe de travail PPSP <http://tools.ietf.org/wg/ppsp> de l'IETF. Son premier RFC, ce RFC 6972 [1], est le cahier des charges du protocole (comme toujours à l'IETF, le travail a commencé bien avant la publication du cahier des charges et des mises en œuvre du protocole PPSP - normalisé dans le RFC 7574 - existent déjà).

Le "*streaming*" est un usage essentiel de l'Internet (le RFC cite une étude de Cisco <http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html> à ce sujet). Dans sa version classique, avec un gros serveur ou une batterie de gros serveurs qui distribuent à des millions de clients, le "*streaming*" ne passe pas à l'échelle. Si on en était resté à ce mode de distribution, seule une poignée de très grosses entreprises pourraient héberger du contenu vidéo. Heureusement, il existe une meilleure solution, la distribution en pair à pair où chaque client se transforme en distributeur pour une partie du contenu. Le "*streaming*" pair à pair permet à n'importe qui de distribuer de la vidéo sur l'Internet, avec seulement des serveurs ordinaires.

Mais, car il y a un mais, ce "*streaming*" pair à pair, aujourd'hui, se fait essentiellement avec des protocoles privés d'une entreprise capitaliste particulière. La section 3 décrit les problèmes que cela pose. D'abord, sur le plan technique, cela interdit de développer des mécanismes de cache communs, le cache

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc6972.txt

devant connaître le protocole utilisé (imaginez les caches Web si tout le monde ne parlait pas HTTP). Même chose pour les CDN qui ne peuvent pas être exploités facilement pour aider ce "*streaming*" puisqu'ils ne peuvent pas connaître tous les protocoles utilisés.

Enfin, les protocoles existants ne sont pas forcément bien adaptés aux mobiles qui sont pourtant une bonne partie des clients <http://www.bytemobile.com/news-events/2012/archive_230212.html>. Il y a eu des recherches à ce sujet (cf. J. Peltotalo et autres, « "*A real-time Peer-to-Peer streaming system for mobile networking environment*" <https://dl.acm.org/doi/10.5555/1719850.1719878> » en 2009) mais qui ne se retrouvent pas forcément dans les protocoles existants. PPSP va donc devoir gérer :

— Le fait que le mobile, contrairement au poste fixe typique, n'est pas allumé et connecté en permanence,
— Le fait que le mobile ait une batterie de capacité limitée (les protocoles actuels n'indiquent pas si le pair est presque à sec...),
— La capacité limitée du réseau avec le mobile : certains protocoles sont très bavards, s'envoyant plein d'informations même quand rien n'a changé,
— L'espace limité sur le mobile, qui rend l'installation de N applications différentes pour la même tâche (le "*streaming*") problématique.

À noter que la section 3 ignore les problèmes plus politiques comme l'excès de pouvoir que ces protocoles fermés donnent à l'entreprise qui les contrôle, et peut ainsi limiter la liberté des clients.

La section 4 décrit à un haut niveau les missions du protocole de "*streaming*" pair-à-pair. Pour modéliser ce protocole, elle reprend largement la terminologie de BitTorrent (la section 2 décrit tout le vocabulaire à connaître mais le RFC ne cite jamais BitTorrent), avec notamment la notion de "*tracker*", le mécanisme (pas forcément une machine, cela peut être une DHT) qui garde trace de tous les pairs servant un contenu donné (ce qu'on nomme l'**essaim**). Il y a en fait deux protocoles, un entre le pair et le "*tracker*" et un entre pairs. Lorsqu'il rejoint un essaim, le pair doit pouvoir trouver les autres pairs (avec le premier protocole) et les contacter (avec le second). Il doit pouvoir connaître les caractéristiques des pairs, pour les choisir astucieusement. Le protocole doit bien sûr être efficace (le "*streaming*" vidéo fait passer d'énormes quantités de données) et doit être robuste (par exemple en cas de panne du "*tracker*").

Le protocole pair¡-¿"*tracker*" va nécessiter un identifiant unique pour les pairs ("*peer ID*"). Il va falloir aussi classer les pairs, par exemple selon le fait qu'ils ont une adresse IP publique ou pas, qu'ils sont en IPv4 ou en IPv6, qu'ils ont des ressources importantes ou pas, etc. Ce sera plutôt un protocole requête/réponse (le pair se connecte, transmet ses infos, et reçoit une liste de pairs).

Le protocole pair¡-¿pair devra se colleter avec les problèmes de l'identification du contenu et de son intégrité (un problème important en pair-à-pair). Le contenu envisagé pour ce problème étant souvent de grande taille, il faudra le découper en morceaux ("*chunks*") et il faudra donc un mécanisme d'identification des morceaux et de leur disponibilité. Ce sera plutôt un protocole de bavardage (échange continu d'information).

Avant d'aborder le cahier des charges précis, le RFC pose une dernière question, quels sont les usages envisagés pour ce nouveau protocole de "*streaming*" (section 5) ? Il y a la distribution de vidéo en temps réel, par exemple lors d'un événement (cela ressemble fortement à l'usage décrit dans les RFC 6707 et RFC 6770). Il y a aussi la VoD. D'autres usages sont moins évidents à première vue comme la possibilité d'avoir des caches pour le "*streaming*". Même si on est dans une relation client/serveur, un protocole pair-à-pair de "*streaming*" permet d'envisager des systèmes de caches automatiques entre le client et le serveur, qui intercepteraient les requêtes et les mémoriseraient pour les futurs utilisateurs. Un exemple proche est l'extension P2Pyoutube <https://addons.opera.com/en/extensions/details/p2p-youtube/> du navigateur Opera.

Enfin, le cahier des charges proprement dit. Je ne vais pas reprendre ici toutes les exigences, classées en catégories (REQ : la base, OAM : gestion du protocole, TP : protocole pair¡-¿"*tracker*", PP : protocole

pair¡-¿pair) et numérotées. Certaines sont de l'ordre de l'évidence (PPSP.REQ.1 : chaque pair doit avoir un identificateur unique, PPSP.REQ.2 dit la même chose pour l'essaim et PPSP.REQ.4 pour le morceau). D'autres sont toujours bonnes à rappeler (PPSP.OAM.REQ-2 : il doit y avoir des paramètres de configuration, ayant des valeurs par défaut, PPSP.OAM.REQ-7 : on doit pouvoir les changer). Beaucoup sont communes à tout système de distribution de contenu en pair-à-pair et se retrouvent déjà dans BitTorrent. D'autres sont plus spécifiques au "*streaming*". Ainsi PPSP.OAM.REQ-4 parle des mécanismes pour atteindre une qualité suffisante pour le flux audio ou vidéo. Cela implique un délai raisonnable. Le RFC dit par exemple qu'une minute de délai pour la retransmission d'un évenement sportif est inacceptable (imaginez le supporter du PSG ayant encore son verre de bière à la main alors que dans les maisons environnantes, tous les autres hurlent déjà « Buuuuuuuuuuuuuuut ! »).

Souci classique à l'IETF, le protocole doit être gérable. Des exigences comme PPSP.OAM.REQ-5 demandent qu'on puisse accéder à l'information nécessaire pour déboguer et/ou optimiser le protocole et PPSP.OAM.REQ-6 ajoute qu'il faut pouvoir tester un pair, sa connectivité, son bon fonctionnement, etc.

De même, les exigences sur le protocole "*tracker*"¡-¿pair sont assez standards et évidentes (PPSP.TP.REQ-1 : le pair doit pouvoir obtenir du "*tracker*" une liste de pairs potentiels...). Notez quand même une exigence que l'authentification du pair soit possible, pour les essaims fermés (PPSP.TP.REQ-4). Même chose pour le protocole pair¡-¿pair. Notez toutefois PPSP.PP.REQ-3 (obtenir directement du pair, sans passer par le "*tracker*", d'autres pairs, de préférence vérifiés, ajoute PPSP.PP.REQ-5) et PPSP.PP.REQ-8, la possibilité d'obtenir plein d'informations sur les pairs.

Voilà, le cahier des charges est fini. Une faiblesse traditionnelle des systèmes pair-à-pair, la sécurité, fait l'objet d'une section à part, la 7. Elle analyse les risques dus à des pairs ou des "*trackers*" malveillants :

— Attaque par déni de service en envoyant plein de requêtes,
— Envoi de fausses informations, par exemple de disponibilité d'un morceau,
— Violation de la vie privée, en transmettant les informations à un tiers,
— Et d'autres encore.

Cela entraîne l'ajout de quelques exigences supplémentaires, étiquetées SEC pour "*SECurity*". PPSP.SEC.REQ-1 reprend l'exigence déjà mentionnée de pouvoir faire des essaims fermés, avec pairs authentifiés. PPSP.SEC.REQ-2 rappelle qu'il est indispensable d'avoir un mécanisme de contrôle de l'intégrité du contenu, pour empêcher un pair malveillant d'injecter des morceaux incorrects, et PPSP.SEC.REQ-3 appelle au réalisme en demandant que tout mécanisme de sécurité passe bien à l'échelle, on peut avoir des essaims immenses <http://torrentfreak.com/game-of-thrones-pirates-break-bittorrent-swarm-reco>.

Les auteurs annoncent qu'il y a déjà trois mises en œuvre du protocole mais il ne semble pas y avoir beaucoup de documentation à ce sujet. Le protocole correspondant à ce cahier des charges a été normalisé dans le RFC 7574.