RFC 5693 : Application-Layer Traffic Optimization (ALTO) Problem Statement

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 28 octobre 2009

Date de publication du RFC : Octobre 2009

https://www.bortzmeyer.org/5693.html

Le groupe de travail Alto <https://www.bortzmeyer.org/alto-wg.html> de l'IETF est chargé de développer un protocole permettant aux participants d'un réseau pair-à-pair de trouver le pair le plus « proche », pour améliorer la communication ultérieure et consommer moins de ressources réseau. Ce premier RFC officiel du groupe décrit le problème à résoudre.

Le fond du problème est que les pairs ne connaissent pas la topologie du réseau. Disposant de trop peu d'informations, ils risquent de choisir, parmi les pairs potentiels, un pair situé très loin d'eux, ce qui se traduira par une capacité réseau inférieure, et une utilisation exagérée des lignes à longue distance, alors que des réseaux à courte distance, moins chargés, étaient disponibles. Le but d'Alto est donc de donner aux pairs le moyen de faire des choix meilleurs que le simple hasard.

Le problème n'est d'ailleurs pas limité aux applications pair-à-pair. Comme le rappelle la section 1, les applications client/serveur traditionnelles ont également parfois à choisir entre plusieurs sources, pour une même ressource (« vais-je télécharger mon image ISO de NetBSD depuis ftp.nl.netbsd.org ou bien ftp.de.netbsd.org ? »). L'Internet est actuellement exploitée de manière très peu « développement durable ». Les applications gourmandes comme le chargement de vidéos ou l'échange de fichiers multimédia consomment beaucoup de capacité du réseau, sans faire d'effort pour minimiser cette consommation.

Comme dans l'exemple de la distribution de NetBSD ci-dessus, la ressource convoitée est aujourd'hui souvent disponible depuis plusieurs sources, chacune ayant une réplique de la ressource (section 1.1). Mais les applications ne disposent pas d'éléments suffisants pour choisir la réplique la plus proche. Des heuristiques comme le pays (dans l'exemple de NetBSD ci-dessus) sont très insuffisantes car la connectivité n'épouse pas les frontières nationales. Les mesures comme celle du RTT (mises en œuvre par un logiciel comme netselect <http://alumnit.ca/~apenwarr/netselect/>) ne donnent pas d'informations sur la capacité du réseau (cf. RFC 5136 [1]), ni sur le coût des différentes lignes (par exemple entre transit et "*peering*"). Voici un exemple avec netselect, où il affiche le serveur ayant le plus faible RTT (l'algorithme exact de netselect est plus riche, voir la documentation) :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc5136.txt

```
% netselect  ftp.be.netbsd.org ftp.nl.netbsd.org  ftp.de.netbsd.org ftp.es.netbsd.org
   25 ftp.be.netbsd.org
```

et, en plus détaillé :

```
% netselect -vv ftp.be.netbsd.org ftp.nl.netbsd.org  ftp.de.netbsd.org ftp.es.netbsd.org
Running netselect to choose 1 out of 5 addresses.
................................
ftp.be.netbsd.org                       11 ms  11 hops  100% ok (10/10) [   23]
192.87.102.43                         9999 ms  30 hops    0% ok
192.87.102.42                         9999 ms  30 hops    0% ok
ftp.de.netbsd.org                       24 ms  10 hops  100% ok (10/10) [   48]
ftp.es.netbsd.org                       17 ms  10 hops  100% ok (10/10) [   34]
   23 ftp.be.netbsd.org
```

Face à ce problème, plusieurs groupes ont expérimenté des solutions visant à donner davantage d'information aux applications (cf. par exemple le RFC 5632, et le RFC 6029 pour une étude détaillée des propositions techniques existantes). Si cela viole un peu le modèle en couches (pour lequel l'application devrait être totalement ignorante du réseau sous-jacent), cela permet en général une nette augmentation des performances **et** une diminution de la consommation de ressources.

L'état de l'art est résumé dans la section 1.2 du RFC. Outre le RFC 5632, on peut consulter les références données dans le RFC comme « "*P4P : Explicit Communications for Cooperative Control Between P2P and Network Providers*" <http://www.dcia.info/documents/P4P_Overview.pdf> ». Ces solutions ont en commun de fournir un mécanisme de découverte de la source d'information et, une fois qu'on a trouvé celle-ci, un mécanisme d'interrogation de la source, pour récupérer les informations qui permettront de prendre une bonne décision.

La section 2 définit les (nombreux) termes utilisés par Alto. Le schéma des protocoles est particulièrement recommandé, il illustre notamment le fait qu'Alto n'est concerné que par la **récupération** de l'information depuis les serveurs Alto, pas par la manière dont cette information a été récoltée par les serveurs.

Enfin, le problème qu'Alto veut résoudre est défini rigoureusement dans la section 3. Dans un environnement où on veut une **ressource** (un fichier MP3, par exemple), pas une machine spécifique, il s'agit de trouver la meilleure machine pour la communication, parmi un ensemble de machines qui hébergent la même ressource. « Meilleure » peut signifier plusieurs choses, débit, coût ou même d'autres critères comme la volonté de ne pas concentrer tous les « clients » sur un seul « serveur ». En tout cas, « meilleur » doit être simplement « meilleur qu'en tirant au hasard », ce choix aléatoire étant la référence de base.

Dans quels cas un protocole de type Alto serait-il utile ? La section 4 donne des exemples de scénarios. Par exemple, le partage de fichiers (section 4.1) est une application courante et très gourmande, les fichiers en question étant souvent du gros multimédia. Si je veux télécharger un film, la quantité de données à faire passer par le réseau justifie certainement de passer quelques secondes à choisir la ou les meilleures machines d'où le charger.

Même chose lorsqu'il faut choisir, dans un protocole client/serveur traditionnel comme HTTP (section 4.2), le meilleur « miroir » d'un site donné (comme dans l'exemple de NetBSD plus haut).

Il existe encore plusieurs exemples comme l'utilisation d'Alto pour aider à construire une DHT (section 4.5).

Alto soulève plein de questions (comme l'avait montré la réunion animée à Dublin <https://www.bortzmeyer.org/alto-wg.html>). La section 5 tente d'y répondre. D'abord, quelle information doit être distribuée par Alto (section 5.1) ? Pour éviter de surcharger le protocole, il faut développer des critères permettant de dire ce qu'on distribue et ce qu'on ne distribue pas. On ne distribue certainement pas avec Alto de l'information très temporaire comme la congestion. Alto devrait servir à transporter de l'information relativement stable et, surtout, que l'application ne peut pas obtenir facilement elle-même. La mesure du RTT, par exemple, n'a pas besoin d'Alto. Mais séparer les liens Internet chers de ceux bon marché ne peut jamais être découvert par l'application et doit donc être fourni par Alto. Même chose pour toutes les notions « comptables » comme le quota restant, dans le cas d'une connexion Internet facturée à l'usage (notez que la plupart des offres Internet sur mobile sont dans ce cas, même si la publicité mensongère des opérateurs les nomme « Internet illimité ».

Et qui va rassembler l'information que les serveurs Alto distribueront (section 5.2) ? A priori, on pourrait imaginer que cela soit fait par les opérateurs (qui connaissent leur réseau), des tiers qui bénéficient de certaines informations fournies par les opérateurs (un bon exemple est Akamai) ou bien, en style davantage « 2.0 », des communautés d'utilisateurs travaillant de manière décentralisée.

Le service Alto pourra donc être présenté de manière différente (section 5.3). Il pourrait être un service centralisé ou bien être lui-même pair-à-pair. Il faudra donc prévoir un mécanisme de découverte, pour savoir où s'adresser.

Évidemment, les applications utilisée peuvent manipuler des données privées. Si j'utilise une application pair-à-pair pour récupérer des films pornos, je n'ai pas forcément envie que mon FAI soit au courant, même si c'est une activité parfaitement légale. La section 5.4 explore donc la question de la protection de la vie privée. Le serveur Alto souhaiterait avoir le plus d'informations possibles pour prendre la meilleure décision, mais l'utilisateur doit avoir un moyen de ne pas trop en dire. Un problème analogue est celui de la protection des données confidentielles de l'opérateur réseau. Celui-ci ne souhaite pas forcément que ses clients soient au courant de la topologie du réseau (section 5.5).

Enfin, le problème de la coexistence avec les caches fait l'objet de la section 5.6.

Alto pose aussi des questions liées à la sécurité (section 6). Si les problèmes de mise en œuvre du contrôle des contenus sont explicitement exclus, d'autres questions restent posées. Par exemple, comme le serveur Alto peut influencer le processus de choix d'un pair (c'est son but), Alto introduit une tierce partie dans le dialogue pair-à-pair et donc un composant de plus pour l'analyse de la securité.

Ainsi, si le serveur Alto est géré par le FAI, certains utilisateurs peuvent ne pas avoir envie de l'utiliser car ils craignent (à juste titre quand on voit le comportement de certains FAI <http://www.pcinpact.com/actu/news/45077-FCC-comcast-sanction-blocage-bittorrent.htm>) que le FAI ne se serve d'Alto pour imposer les quatre volontés de Warner ou de Disney, ou bien pour espionner l'activité de certains utilisateurs, ou encore pour appliquer des politiques de sélection qui sont dans l'intérêt du FAI mais pas dans celui des utilisateurs (faire passer par des routes lentes mais moins chères, par exemple).

Toutefois, il n'est pas envisagé de rendre Alto obligatoire. Si certains serveurs Alto n'agissent plus dans l'intérêt des utilisateurs, le plus simple sera de ne pas les utiliser.