Sélectionner le meilleur pair

Stéphane Bortzmeyer
AFNIC
bortzmeyer@nic.fr

29 novembre 2010

Le problème

Je veux récupérer un gros fichier (mettons une image d'installation d'un logiciel) en pair-à-pair.

Le système P2P que j'utilise (par exemple BitTorrent) trouve que 1 000 pairs potentiels (l'**essaim**) ont ce fichier. Lequel utiliser ? Certains sont chez le même FAI que moi, d'autres à l'autre bout du monde. Et j'ai juste une liste d'adresses IP.

Petit détour : comment on trouve la liste des pairs

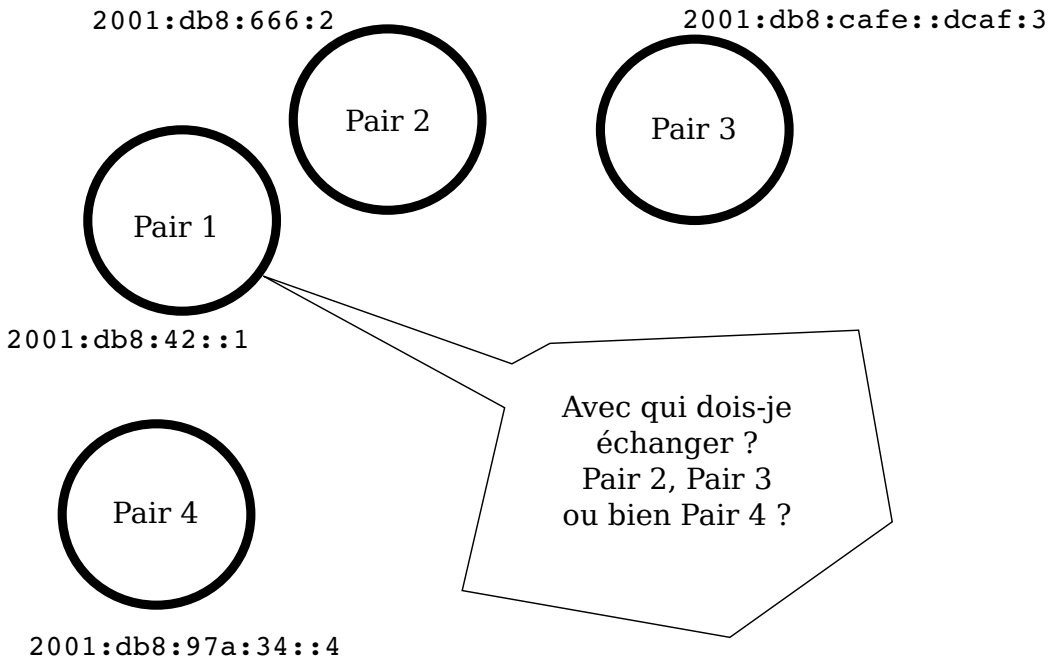
Elle peut être obtenue par un serveur central, le *tracker*.

Ou bien dans le DNS.

Ou encore en pair-à-pair, sur une DHT.

Dans le premier cas, le serveur aura peut-être déjà fait un tri.

Le problème, en un dessin



Avec seulement trois pairs, le choix est facile. Mais avec 10 000 ?

Les deux approches pour trier

1. L'application P2P peut mesurer elle-même la qualité des pairs et en déduire le meilleur.
2. Ou bien elle peut demander à un serveur spécialisé. Deux sous-cas :
   - 2.1 Le serveur donne une « carte » du réseau et l'application P2P en déduit le pair le plus « proche ».
   - 2.2 Le client donne la liste des pairs potentiels et le serveur lui indique les meilleurs.

Pour une comparaison des deux approches, et une revue de la littérature, RFC 6029 `http://www.bortzmeyer.org/6029.html`

Mesurer soi-même

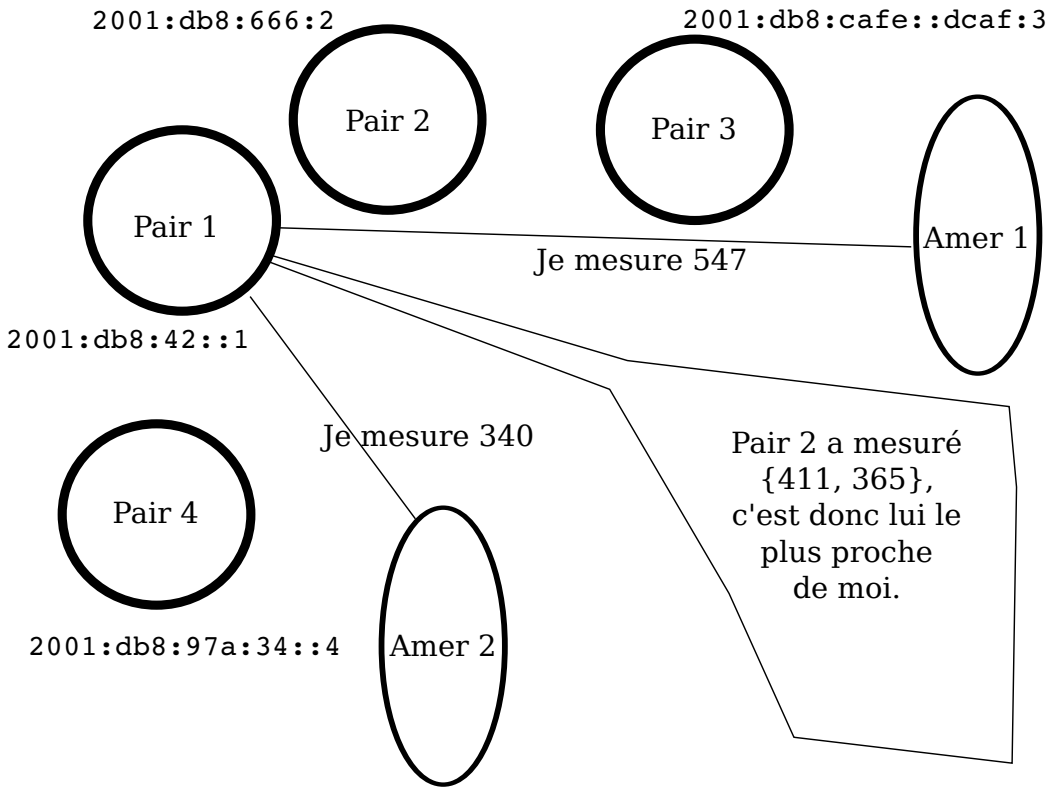
Quelques questions préalables :

1. Mesurer quoi : distance géographique, *AS path*, latence, taux de pertes ou débit ? Ce dernier ne peut se mesurer qu'a posteriori.
2. Mesurer comment : ICMP (`ping`) n'est pas forcément représentatif du trafic réel. Mesures actives (`ping`) ou passives (transférer et voir le résultat) ? Un GPS ou un client BGP dans chaque pair ?
3. Mesurer qui : directement les pairs ? Mauvaise *scalability* : on peut mesurer trois pairs potentiels mais pas 10 000. Il vaut mieux utiliser des **systèmes de coordonnées** ou bien des mesures de distance hiérarchiques.

Systèmes de coordonnées Internet

1. On définit N **amers**, des machines toujours (bien) connectées que tout le monde peut voir,
2. L'application mesure une « distance » (par exemple un RTT) à chaque amer,
3. Elle peut alors calculer sa position dans un espace à N dimensions,
4. Lorsqu'on récupère la liste des pairs, on récupère aussi leurs positions. Calculer la distance à chaque pair devient trivial.

Systèmes de coordonnées Internet, dessin



Avantages des systèmes de coordonnées

Le temps de mesure est proportionnel au nombre d'amers, pas au nombre de pairs. Mis en œuvre, par exemple, dans Vivaldi.
`http://www.bortzmeyer.org/internet-coordinates.html`

Mesures de distance hiérarchiques

Autre solution pour mesurer, avoir un système **hiérarchique**. Par exemple, Meridian repose sur un système de pairs déjà connus (et donc de distance connue) :

1. Les pairs connus sont regroupés en anneaux concentriques, par ordre de distance.
2. Pour un pair inconnu, on demande aux pairs connus, qui redirigent vers une machine plus proche.
3. Jusqu'à ce qu'on arrive suffisamment proche.

Zooms successifs, pas besoin d'amers.
`http://www.bortzmeyer.org/meridian.html`

Mesurer soi-même, les limites

1. L'application P2P ne peut pas trouver seule qu'un lien est un transit payant ou un *peering* gratuit.
2. D'une manière générale, elle ne peut pas trouver seule quelles sont les préférences du FAI.

Demander à un serveur

Au lieu de mesurer soi-même, on demande à un serveur. Ce serveur aura pu avoir l'info :

1. Par les techniques de mesures présentées plus haut,
2. Par une configuration manuelle, typiquement par le FAI.

Il va falloir définir un protocole normalisé pour interroger ce serveur.

Faire confiance ?

Intérêt du FAI : utiliser les liens de *peerings* gratuits, même s'ils sont surchargés.

Intérêt de l'utilisateur : télécharger le plus vite possible.

Il peut donc y avoir conflit.

C'est un problème fréquent en allocation de ressources.

L'utilisation de ce système doit rester facultative (comme cela, les FAI seront motivés pour le rendre efficace pour les utilisateurs).

Cartes contre oracles

Deux approches pour un protocole d'information client/serveur (Attention : vocabulaire pas vraiment normalisé encore.)

1. La carte : le serveur envoie au client des informations sur la topologie du réseau, et le client sélectionne.
2. L'oracle (« *You're the Oracle ?* » « *Bingo. Not quite what you were expecting, right ?* ») : le client envoie au serveur la liste des pairs potentiels et le serveur les trie, puis indique les meilleurs

La carte, avantages et inconvénients

La carte donne au client une complète liberté de choix du pair.

Mais elle nécessite le transfert de données de grande taille.

Et elle oblige le FAI à déclarer sa topologie, ses relations d'affaire, etc.

L'oracle, avantages et inconvénients

L'oracle minimise les calculs faits sur le client.

L'oracle éviter au FAI de dévoiler sa carte.

Mais il oblige le client à indiquer ses pairs potentiels.

Et le client ne peut rien vérifier, il doit faire confiance.

≪ *Morpheus believes in you, Neo and no one, not you or even me can convince him otherwise.* ≫

Le protocole ALTO

ALTO (*Application-Layer Traffic Optimization*) sera le protocole standard de l'IETF pour interroger un serveur afin de trouver le meilleur pair.

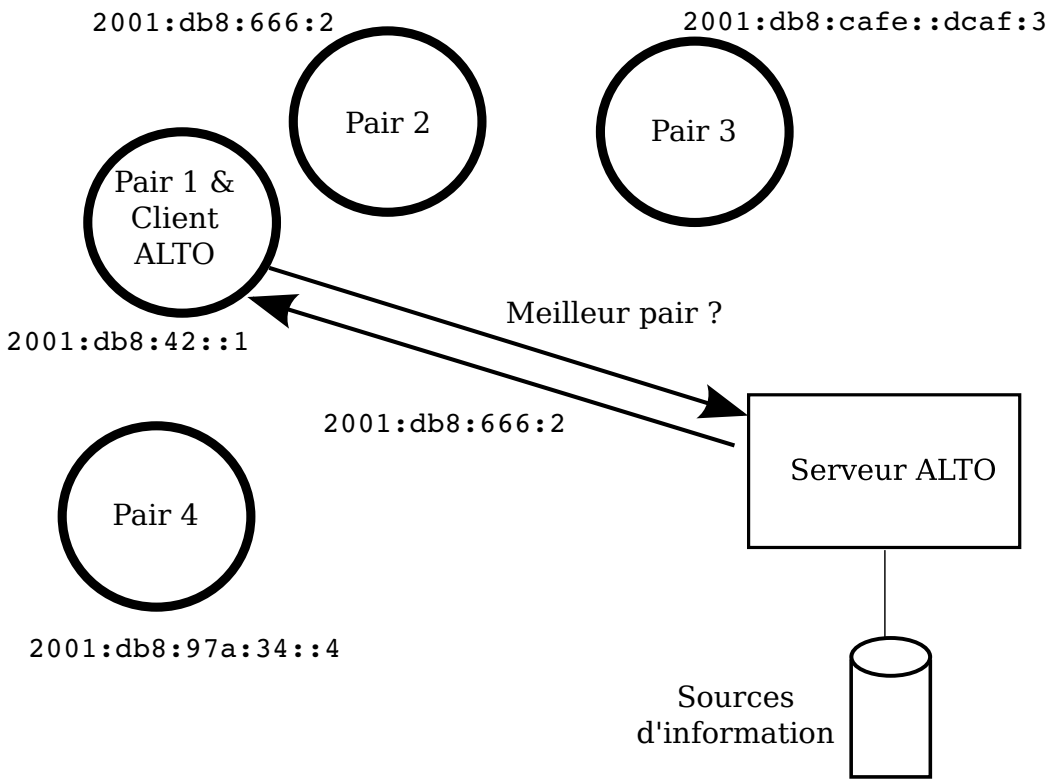
Le problème qu'essaie de résoudre ALTO est décrit dans le RFC 5693 `http://www.bortzmeyer.org/5693.html`

Le protocole ALTO, suite

ALTO permet les **deux** fonctionnements : carte (*target-independent*) et oracle (*target-aware*).

- Le serveur envoie au client une *Network Map* et une *Cost Map* et le client calcule,
- Ou bien le client envoie une liste d'adresses IP au serveur et le serveur renvoie une *Cost Map* qui indique leurs coûts (distances). Le client peut indiquer le type de coût qui l'intéresse (coût de routage, par exemple).

ALTO en Oracle



Détails techniques sur ALTO

ALTO est actuellement décrit dans l'*Internet-Draft* `draft-ietf-alto-protocol-06`.

Il utilise HTTP, avec des requêtes codées en JSON.

Un PID (*provider-defined Network Location identifier*) identifie un groupe de machines proches, par exemple :

```
"Montpellier-POP" : [
                        "192.0.2.0/26",
                        "198.51.100.0/25"
                    ],
"Internet" : [ "0.0.0.0/0" ]
```

Un exemple de requête ALTO

Un exemple trivial où il n'y a que deux PID, ma ville et le reste de l'Internet :

```
POST /map/filter/pid/cost HTTP/1.1
Host: alto.example.net:6671

       {
           "srcs" : [ "Montpellier-POP" ],
           "dsts" : [ "Montpellier-POP", "Internet" ]
       }

       HTTP/1.1 200 OK
...
         "map" : {
             "Montpellier-POP":
                 { "Montpellier-POP": 0,  "Internet": 2 }
         }
```


Autres documents sur ALTO

- RFC 5693, décrivant le problème.
- `draft-ietf-alto-reqs`, le cahier des charges du protocole. L'exigence ARv06-25 impose les deux modes, carte et oracle.
- `draft-ietf-alto-protocol`, le protocole.
- Autres documents, par exemple sur les problèmes de déploiement (qui doit payer pour ce nouveau service ?), ou bien sur des extensions au protocole. . .

Mécanisme, pas politique

- Le protocole ALTO est « neutre ». Il n'impose pas de politique particulière,
- Tout le monde peut faire tourner un serveur ALTO, pas uniquement le FAI
- ALTO ne dit pas comment obtenir les informations mises dans les *maps*. Configuration statique ou bien mesure dynamique (et, dans ce cas, BGP, OSPF, ping, etc), c'est le serveur ALTO qui décide.
- ALTO ne dit pas comment le client peut utiliser cette information, ni même s'il doit l'utiliser.

État actuel d'ALTO

Le groupe de travail a fourni un effort important et les principes de base semblent acquis.

Le RFC important, celui sur le protocole, n'est pas encore fini (en retard sur les prévisions initiales). On peut espérer sa publication dans la première moitié de 2011.

Il y a déjà plusieurs mises en œuvre du protocole dans son état actuel. (Démo à IPTComm 2010 en août.)

Histoire d'ALTO

1. Atelier IAB en mai 2008. RFC 5594
   `http://www.bortzmeyer.org/5594.html`
2. Première réunion informelle IETF en juillet 2008 à Dublin
   `http://www.bortzmeyer.org/alto-wg.html`
3. Création du groupe de travail en novembre 2008
4. Tests P4P publiés en 2009, RFC 5632
   `http://www.bortzmeyer.org/5632.html`
5. Premier RFC du groupe ALTO en octobre 2009, RFC 5693 (*Problem statement*)