Parcours de réseau pair à pair

Sujet proposé par Laurent Viennot

Laurent.Viennot@inria.fr
http://gyroweb.inria.fr/~viennot/enseignement/projets/gnutella/

1 Introduction

Le but de ce projet est d'explorer le réseau pair à pair Gnutella. Un réseau pair à pair est constitué de clients interconnectés via Internet. Chaque participant se connecte avec un programme appelé *client* à d'autres clients. Ce type de réseau pair à pair est généralement utilisé pour échanger des fichiers de manière décentralisée. La principale fonction du réseau étant de trouver l'adresse IP d'un participant possédant un fichier répondant à certains critères de requête comme des mots du nom du fichier.

Un réseau pair à pair constitue donc un réseau logique : chaque client est connecté à un petit nombre d'autres clients dont il connaît l'adresse IP et avec lesquels il échange des messages. Deux clients directement connectés sont dit *voisins*. Le but de ce projet est d'explorer la topologie du réseau Gnutella.

2 Le réseau Gnutella

L'avantage du réseau Gnutella pour une telle étude est de reposer sur un protocole public [1]. Pour découvrir des clients Gnutella, il y a deux manières. La première consiste à trouver un client du réseau qui veut bien accepter une connexion de la part d'un nouveau client. La seconde consiste à échanger des messages une fois connecté selon le protocole Gnutella.

Pour connaître des clients, Gnutella repose sur un système de caches web. On pourra en trouver avec la requête `gwebcache` sur un moteur de recherche tel que `google`. On peut ainsi trouver facilement quelques dizaines de clients du réseau Gnutella pour essayer de s'y connecter. Chaque client est identifié par son adresse IP et le port sur lequel il écoute (par exemple `65.163.60.204:6346`).

2.1 Entrer dans le réseau pair à pair

Une fois que l'on connaît un client, on peut tenter de s'y connecter en ouvrant une connection TCP (classe `java.net.Socket` en java) à l'adresse et au port d'écoute de celui-ci. Une fois la connexion ouverte, il faut l'initier en envoyant :

```
GNUTELLA CONNECT/0.6\r\n
User-Agent: mon_programme\r\n
\r\n
```

Si le client accepte la connection, il répondra une séquence du type :

```
GNUTELLA/0.6 200 OK\r\n
User-Agent: BearShare\r\n
Query-Routing: 0.1\r\n
\r\n
```

Il faut enfin envoyer :

```
GNUTELLA/0.6 200 OK\r\n
\r\n
```

Différents champs du type `Champ: valeur\r\n` peuvent être échangés. `User-Agent` défini par exemple le nom du client.

Si le client auquel on veut se connecter refuse la connexion, il répondra une séquence du type :

```
GNUTELLA/0.6 503 Too many Gnet connections (5 max)
User-Agent: gtk-gnutella/0.91.1
Remote-IP: 128.93.43.62
X-Live-Since: Sun, 23 Feb 2003 09:07:50 -0600
X-Try: 172.177.201.249:7098, 24.107.38.175:6346, 12.153.201.41:6346,
    80.131.56.124:6346, 213.200.158.230:6346, 194.74.6.111:6346,
    68.80.180.41:6346, 68.15.146.124:6346, 68.82.193.226:6346,
    24.91.245.1:8687
```

Un client qui refuse une connexion est invité à proposer l'adresse d'autres clients (champ `X-Try`) auxquels on pourra alors essayer de se connecter.

2.2 Échange de messages avec un pair

Une fois la connexion établie, on peut échanger des messages avec le client pair. Tout message est constitué d'une en-tête et d'un contenu. Le format de l'en-tête est le suivant :

	Identificateur	Type	TTL	Hops	Longueur
Octets	0 15	16	17	18	19 22

Les champs sont les suivants :

- `Identificateur` : 16 octets aléatoires identifiant le message,
- `Type` : le type du message (`0x00` pour un Ping, `0x01` pour un Pong, `0x02` pour un Bye, `0x40` pour un Push, `0x80` pour un Query, `0x81` pour un QueryHit),
- `TTL` : « Time To Live » ou limite de vie (ce champ est décrémenté chaque fois que le message est retransmis, un TTL 0 implique que le message ne sera pas répété),
- `Hops` : nombre de sauts (nombre de fois que le message a été répété, `TTL + Hops` reste constant à chaque retransmission),
- `Longueur` : la longueur du contenu du message en octets (le codage est « little indian » : les octets 19 et 20 sont ceux de poids faible et les octets 21 et 22 ceux de poids fort).

Un Ping sert à trouver d'autres clients pairs dans le réseau : un noeud qui reçoit un Ping doit répondre normalement par un Pong qui contient entre autres son adresse IP et son port. Un Ping avec `TTL` 2 permet de connaître les voisins d'un client auquel on est connecté.

Un Ping n'a pas de contenu (il est constitué uniquement d'une en-tête). Un Pong a même identificateur que le Ping auquel il répond et son contenu a le format suivant :

	Port	Adresse IP	Nbfic	NbKo
Octets	0 1	2 5	6 9	10 13

Les champs sont les suivants :

- `Port` : le port d'écoute du client pair (codage « little endian »),
- `Adresse IP` : l'adresse IP du client pair (codage « big endian » : `1.2.3.4` est codé `0x01 0X02 0X03 0x04`),
- `Nbfic` : nombre de fichiers partagés,
- `NbKo` : nombre de kilo-octets partagés.

Les autres messages servent à propager des requêtes et à permettre d'ouvrir des connexions pour copier des fichiers. Pour plus d'information sur le protocole Gnutella et ses extensions, voir [2, 1].

3 Travail demandé

Pour explorer la topologie du réseau Gnutella, il est demandé de programmer un client Gnutella qui tentera de se connecter à un maximum de clients pairs et de récupérer la liste de leur voisins au moyen d'un Ping de TTL 2. On retiendra aussi, pour un client pair refusant la connexion, les clients qu'il pourrait lister dans un champ `X-Try` comme des clients qu'il connaît. Pour que l'exploration soit efficace, le programme devra émuler plusieurs clients simultanément au moyen de threads.

On tentera de donner des statistiques sur le nombre clients identifiés, ou ayant accepté la connexion. On tentera aussi de caractériser le graphe de voisinage observé, ainsi que le graphe de connaissance et l'union de ces deux graphes (on calculera par exemple le diamètre de ces graphes).

4 Extension

En extension, on pourra programmer un client Gnutella complet capable de se connecter à un très grand nombre de pairs. Comparer le taux de réussite des requêtes en étant connecté normalement au réseau ou en étant connecté à un très grand nombre de pairs.

References

[1] T. Klingberg and R. Manfredi. Gnutella 0.6, 2002. `http://rfc-gnutella.sourceforge.net/draft.txt`.

[2] Site sourceforge. Gnutella, 2002. `http://rfc-gnutella.sourceforge.net/`.