Programmation Réseau

Le pair à pair

Jean-Baptiste.Yunes@liafa.jussieu.fr

UFR Informatique

2011-2012

Le pair à pair

- l'architecture de système communicant est traditionnellement celle du client-serveur
  - dans ce mode, le serveur, entité passive, attend des requêtes en provenance de clients, entités actives, afin de les traiter
- le problème majeur de ce problème est la congestion
  - le serveur est un goulot d'étranglement

- il existe plusieurs façons d’alléger la congestion
  - en particulier la répartition de charge (réplicats de serveurs - pensons au service de google par exemple)
- dans le cas particulier de services type transfert de données
  - on peut imaginer aller chercher les données à la source
    - afin d’éviter de les déposer sur le serveur pour qu’un client aille les y chercher ensuite

- en ce cas, les différentes entités sont à la fois clientes et serveurs

  - clientes car elles peuvent aller chercher des données

  - serveurs car elles peuvent être sollicitées pour obtenir des données

- ce modèle est celui dit du pair-à-pair (peer-to-peer), d'égal-à-égal ou poste-à-poste (selon la recommandation de la Commission générale de terminologie et néologie)

- un peu de terminologie :
  - le **système** pair-à-pair fait référence à l'ensemble des technologies employées (essentiellement les protocoles employés)
  - un **nœud** est une entité logicielle intervenant dans le système
  - un **lien** est un canal de communication établi entre nœuds
  - un **réseau** est un instantané des nœuds et liens
  - dans un système pair-à-pair on s'échange des **objets**

université PARIS DIDEROT PARIS 7

- dasn un système pair-à-pair un problème majeur est à résoudre :
  - comment obtenir la localisation d'une donnée particulière ?
    - les différentes réponses apportées constituent l'histoire des systèmes pair-à-pair

1ère génération (aka Napster)

- pour partager un fichier, il suffit de se déclarer auprès d'un serveur qui possède l'annuaire général

- pour obtenir un fichier, il est nécessaire d'interroger le serveur qui en réponse donne l'adresse de tous ceux qui possèdent ce fichier

  - le transfert s'effectue alors directement entre nœud

- problème : centralisation de l'index...

2e génération (aka gnutella)

- plus d'annuaire central, chaque nœud indexe ses propres données et se déclare à ses voisins dans le réseau

  - problème : comment trouver des voisins ?

    - faire circuler des listes de nœuds (sur des pages web par exemple)

- les requêtes circulent de proche en proche jusqu'à trouver une source pour la donnée

- inconvénient : le réseau est globalement ralenti par les nœuds les plus lents

université PARIS DIDEROT PARIS 7

3ième génération (Kazaa, gnutella 0.6)

- distinction entre nœuds bien connectés et les autres : notion de super-pairs

  - l'index est possédé par les super-pairs

  - ceux-ci servent de paravent pour les requêtes de localisation (les nœuds ordinaires délèguent leur annuaire aux super-pairs)

  - les transferts continuent de se faire entre pairs quelconques

Un nouveau problème

- en ADSL, l'important c'est le A
  - **Asymmetric** Digital Subscriber Line
- le débit montant est bien plus faible que le débit descendant...
  - i.e. on sort plus lentement qu'on ne rentre...
- le téléchargement depuis une source en ADSL est donc pénalisé par le débit montant de cette source...

université PARIS DIDEROT PARIS 7

eDonkey, eMule

- solution : utiliser plusieurs sources en espérant que la somme des débits montants de cette source corresponde au débit descendant du client...

- pour se faire, les données sont découpées en tranches

  - des tranches différentes sont réclamées à différentes sources et ré-assemblées par le client

- on constate alors que ce découpage favorise le partage : on peut partager tout morceau déjà reçu sans attendre d'avoir tout reçu!
  - la coopération favorise le partage
    - ou télécharger c'est partager!!!
- d'autre part, une observation élémentaire permet de constater qu'on peut télécharger un fichier dans son intégralité alors même que personne ne le possède en entier!
  - la fragmentation favorisant la persistance
    - ou découper c'est mieux stocker

université PARIS DIDEROT PARIS 7

BitTorrent

- mais un nouveau problème survient
  - pendant la phase de téléchargement, la coopération est augmentée
    - mais les internautes ayant tendance à se déconnecter après le téléchargement
      - la coopération diminue fortement (puisqu'une source potentielle disparaît)
- solution : récompenser ceux qui partagent et pénaliser les goinfres radins...

Encore un problème

- le routage de proche en proche des requêtes est trop goinfre en termes de ressources (dans gnutella même volume pour les requêtes que pour les transferts!)
  - comment optimiser ce volume ?
- avant de répondre, faisons un détour par les table de hachage

Texte de la section

- avant de songer à bâtir un annuaire, il est nécessaire d'attribuer à chaque objet un identifiant

  - tout système d'identification imaginable est raisonnable

    - il existe une normalisation, la RFC 4122, « A Universally Unique Identifier (UUID) URN Namespace »

- dans les systèmes pair-à-pair on utilise habituellement un système de hachage

La table de hachage (hashtable)

- il existe de nombreuses structures permettant de stocker des données (tableaux, liste, arbres, etc.)

- la seule vraiment efficace est le tableau :

  - l'accès à la donnée s'effectue en temps constant

    - en supposant (évidemment) que l'on connaisse le numéro (l'identifiant) de la donnée

- dans le cas général, les données ne peuvent être simplement associées à un numéro

- la technique est de générer à partir de la donnée un nombre (valeur de hachage - hash value) qui servira d'index

  - l'idéal serait que pour N données distinctes quelconques (quelque soit leur volume) on sache générer une permutation correspondante de 1 à N

    - en pratique, ce n'est pas possible

      - mais la probabilité des collisions peut être extrêmement faible

- l'algorithme le plus employé est SHA-1 (il en existe une version améliorée SHA-2)
  - SHA-1 permet d'obtenir à partir de n'importe quelle suite finie d'octets une valeur de hachage de 160 bits
    - donc permet de distinguer 2^{160} données différentes
    - il est en théorie possible de retrouver deux données possédant la même valeur de hachage, mais c'est en pratique impossible à construire (2^{51} opérations sont nécessaires en moyenne)
- Note : SHA-1 est normalisé...

La table de hachage distribuée (DHT)

- le principe est le suivant :
  - l'idée est de découper en morceaux l'espace des valeurs de hachage
  - des nœuds différents seront responsables de parties différentes de l'espace
  - une correspondance simple est établie entre l'adresse sur le réseau pair-à-pair du nœud et la partie de l'annuaire qu'il prend en charge

- pour effectuer une recherche, il suffit de rechercher parmi les voisins ceux qui ont un identifiant le plus proche de l'empreinte recherchée
    - et de proche en proche on sélectionne les nœuds les plus adaptés à retransmettre la requête
        - on économise énormément (diffusion logarithmique)
- pour déclarer un fichier, il suffit de contacter le nœud en charge de la fraction de l'espace pour s'y déclarer en tant que source

université PARIS DIDEROT PARIS 7