

Systèmes et algorithmes répartis

Systèmes à grande échelle, pair à pair

Philippe Quéinnec

Département Informatique et Mathématiques Appliquées
ENSEEIH

27 août 2019



plan

- 1 Passage à grande échelle
- 2 Diffusion à grande échelle
 - Algorithmes structurés
 - Algorithmes probabilistes
- 3 Systèmes pair à pair
 - Principales difficultés
 - Classification
 - Systèmes non structurés
 - Systèmes structurés



Plan

- 1 Passage à grande échelle
- 2 Diffusion à grande échelle
 - Algorithmes structurés
 - Algorithmes probabilistes
- 3 Systèmes pair à pair
 - Principales difficultés
 - Classification
 - Systèmes non structurés
 - Systèmes structurés



Passage à grande échelle

Grande échelle

- Grand nombre de sites, d'objets. . .
- Grand nombre d'interactions
- Grande taille (géographique)

Grand = ? (ça dépend !)

Scalability

La capacité de croissance (*scalability*) est la propriété pour un système de conserver ses qualités (performance, robustesse. . .) lorsque sa taille change d'échelle

taille 4 → 128, 32 → 1024, 1000 → 100 000



Champs d'application

- Découverte, observation et accès à des ressources nombreuses
- Collecte de données (surveillance d'installations, capteurs)
- Détection de pannes
- Base de données à grande échelle (SIG – système d'information géographique)
- Absence d'infrastructure « officielle » : rôle symétrique des sites (tous client et serveur)



Fiasco pour le passage à grande échelle

- Algorithmes centralisés, point de contrôle unique
⇒ véritables algorithmes répartis
- Algorithmes linéaires ($O(n)$) en le nombre de sites, ou pires
⇒ $O(\log n)$
- Hypothèse sur la structure statique du système
⇒ ajout et retrait de sites, reconfiguration du réseau, partitionnement
- Considérer que la défaillance de site est un événement exceptionnel
⇒ il existe des sites défaillants en permanence
- S'adresser à l'ensemble des sites (diffusion générale)
⇒ propagation (par inondation, arborescente, probabiliste)



Outil : les réseaux de recouvrement

Réseau de recouvrement ou *overlay*

Réseau logique, virtuel, au-dessus d'un réseau physique existant

- Couche applicative :
 - Réimplantation du routage
 - Ajout de fonctionnalité : nommage, stockage
- Intérêt :
 - Indépendance par rapport au(x) réseau(x) physique(s) sous-jacent(s)
 - Souplesse et évolutivité (niveau applicatif)



Plan

- 1 Passage à grande échelle
- 2 Diffusion à grande échelle
 - Algorithmes structurés
 - Algorithmes probabilistes
- 3 Systèmes pair à pair
 - Principales difficultés
 - Classification
 - Systèmes non structurés
 - Systèmes structurés



Diffusion à grande échelle

Limites des approches classiques

- Ensemble bien identifié de sites (notion de groupe)
- Propriétés fortes (fiabilité, ordre, atomicité) néfastes au passage à l'échelle

Besoins

- Nombre de sites inconnu
- Grand nombre de sites
- Nombre et identité des sites variables



Diffusion par inondation

Cf chapitre IV « problèmes génériques »

Diffuser(m), sur p

```
-- p = émetteur, m = message
∀ s ∈ voisins(p) ∪ {p} faire
    envoyer(⟨p,m⟩) à s
fin pour
```

Réception(⟨p,m⟩), sur q

```
si q n'a pas déjà délivré m alors
    si p ≠ q alors                -- propagation
        ∀ s ∈ voisins(q) faire
            envoyer(⟨p,m⟩) à s
        fin pour
    fin si
    délivrer(m)
fin si
```

Groupes et diffusion

Notion de groupe

Cf chapitre V « Tolérance aux fautes »

Limites

- Vision synchrone des arrivées et départs
- Propriétés fortes (fiabilité, ordre), coûteuses et non indispensables
- Taille d'un groupe limitée



Arbre de recouvrement

- Construire un arbre issu du site de diffusion et contenant tous les sites
- Approximation : graphe orienté acyclique avec détection de messages en doublon
- Difficulté : construire l'arbre...
- ... mais c'est simple quand on a une table de routage hiérarchique, cf transparent 38



Algorithmes épidémiques

Diffusion à grande échelle

- Grand nombre de sites (> 100), voire très grand (> 10000)
- Nombre inconnu et variable de sites
- Rôle symétrique de tous les sites
- Présence de sites en panne
- Topologie d'interconnexion inconnue (a priori non directement maillée)

Approche

- Algorithmes probabilistes
- Propagation aléatoire

1. *Epidemic Algorithms for Replicated Database Maintenance*, Alan Demers et al. 6th Symposium on Principles of Distributed Computing. Aug. 1987.

Algorithme épidémique : Rumeur

Rumeur (*Gossip*)

Contamination d'autres sites choisis aléatoirement avec une information supposée nouvelle.

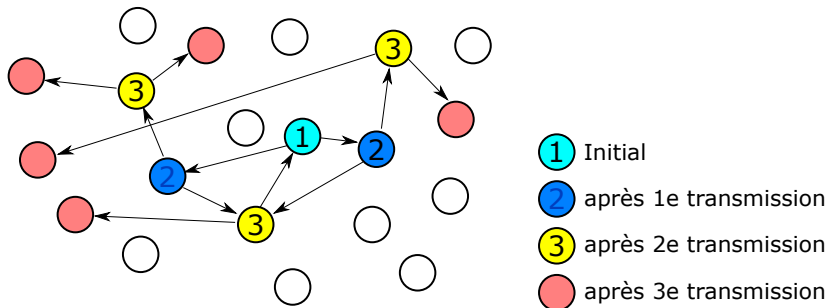
États d'un site :

- Infectable : ne possède pas l'information
- Contagieux : apte à contaminer des sites infectables
- Immunisé : a cessé d'être contagieux
 - Immunisation en aveugle ou avec rétroaction (échec d'une tentative de contamination)
 - Compteur (de tentatives ou d'échecs)
 - Probabilité d'abandon après chaque tentative / échec



Rumeur : exemple

20 sites, nombre de sites contactés à chaque tour = 2,
immunisation en aveugle à 1 tour

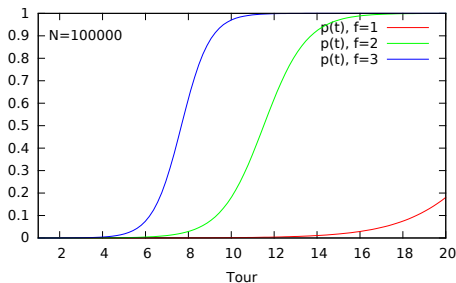


Rumeur : performance

Modèle simple : *infect forever* = pas d'immunisation

- N = nombre de sites
- f = nombre de sites contactés à chaque tour par chaque site
- $I(t)$ = nombre de sites infectés (contagieux ou immunisés) après le t -ième tour
- $p(t) = I(t)/N$ = proportion de sites infectés au t -ième tour

$$\text{Alors } p(t) = \frac{1}{1+(N-1)*e^{-f*t}}$$



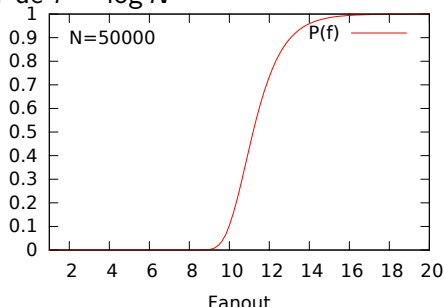
Rumeur : performance

Modèle simple : *infect and die* = immunisation après un seul tour

- N = nombre de sites
- f = nombre de sites contactés par chaque site
- P = probabilité que tous les sites finissent par être infectés (à l' ∞)

Alors $P \approx e^{-e^{\log N - f}}$

Inversion autour de $f = \log N$

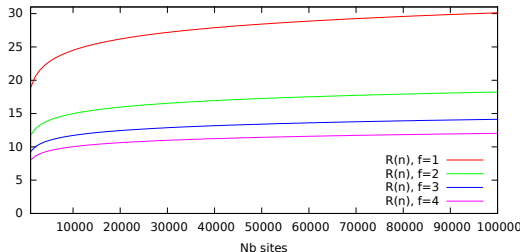


Rumeur : performance

Nombre de tours pour tout contaminer :

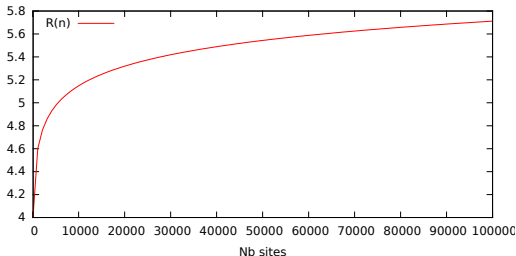
- Modèle sans immunisation (*infect forever*) :

$$R = \log_{f+1} n + \frac{1}{f} \log n + O(1)$$



- Modèle immunisation immédiate (*infect and die*) :
(fin de l'épidémie avec tous contaminés)

$$R = \frac{\log n}{\log \log n} + O(1)$$



Rumeur : coût

- Diffusion **probabiliste**
- Existence de sites non informés (poches d'ignorance si le choix des sites est plutôt local)
- Nombre de messages élevés (échecs de contamination)



Algorithme épidémique : Anti-entropie

Anti-entropie

Périodiquement, chaque site contacte aléatoirement un autre site. Les deux sites échangent alors des informations, et leurs objets sont mis en cohérence.

L'algorithme **converge** vers l'égalité des copies : cohérence à terme.
Nombre de tours pour tout contaminer = $O(\log N)$



Graphes petit monde

Les algorithmes précédents n'ont pas de notion de voisinage : choix arbitraire d'un site quelconque (graphe complet).

Graphe petit monde (*small-world graphs*)

Graphe connexe vérifiant :

- Grand nombre de nœuds ($N > 10000$)
- Faible connectivité des nœuds (de l'ordre de 5 à 10)
- Distance entre deux nœuds quelconques $\approx \log N$
- Remarquablement adapté aux algorithmes épidémiques, résistant aux pannes (de sites et de liens)
- Apparaît spontanément (ex : réseau routier : maillage local, rocade, autoroutes)
- ... ou pas (ex : réseau aérien avec quelques gros hubs à forte connectivité)



Plan

- 1 Passage à grande échelle
- 2 Diffusion à grande échelle
 - Algorithmes structurés
 - Algorithmes probabilistes
- 3 **Systèmes pair à pair**
 - Principales difficultés
 - Classification
 - Systèmes non structurés
 - Systèmes structurés



Problème à résoudre

- Stocker de l'information
- Trouver de l'information

⇒

- Utiliser l'ensemble des participants comme serveurs de stockage distribué
- Utiliser une partie des participants comme répertoire de nommage

Chaque nœud est à la fois client et serveur



Domaine d'application

Partage

- informations/fichiers
- ressources de calcul (grid computing)
- ressources de stockage (réplication)
- bande passante (CDN *Content delivery network*)
- interactions (jeux massivement multijoueur)



Réseaux de recouvrement

Réseau de recouvrement ou *overlay*

Réseau logique, virtuel, au-dessus d'un réseau physique existant

- Couche applicative
 - Réimplantation du routage
 - Ajout de fonctionnalité : nommage, stockage
- Pairs : réseau formé par les participants, tous (à peu près) égaux



Difficultés (1)

Maintenance du réseau de recouvrement

- Démarrage
 - Création du réseau ? (premier site : cas particulier)
 - Insertion/retrait d'un pair dans le réseau
- Maintenance continue
 - Faute, retrait involontaire, expulsion
- Terminaison : arrêt du réseau ?

Passage à l'échelle

- Éviter un (ou quelques) serveurs centralisés
- Distribuer la charge sur les pairs
- Borner la charge sur chaque pair (CPU, bande passante, stockage)

Difficultés (2)

Équité

- Équilibrer la charge : égalitairement ? proportionnellement ?
- Utilisateurs égoïstes : contrôles et incitations à l'équité

Défaillances

- Maintenance de l'overlay à tout prix
- Défaillances de pairs, de liens de communication
- Partition temporaire du réseau, réinsertion ?



Difficultés (3)

Adaptabilité

- Ajout/retrait de sites par vagues (heures ouvrables)
- Ajout/retrait d'informations parfois massif (plusieurs milliers d'un coup)

Performance

- Efficacité : localisation, accès
- Latence du réseau (localité d'accès)
- Parallélisation



Classification : contrôle

- Contrôle centralisé : un serveur central met en correspondance les pairs
 - + simple
 - fragile
 - faible capacité de croissance
- Contrôle totalement décentralisé : tous les nœuds jouent un rôle symétrique (client et serveur)
 - + pas de point central, confidentialité
 - + disponibilité élevée
 - complexe, gestion hasardeuse
 - performance indéterminée
- Contrôle partiellement décentralisé : un ensemble dynamique de nœuds jouent un rôle privilégié



Classification : réseau virtuel

- Non structuré : le placement des données n'est pas lié à la topologie du réseau
 - + bonne adaptabilité avec un ensemble de nœuds très dynamique
 - recherche inefficace en absence de contrôle centralisé
- Structuré : les données sont placées en des points prédéterminés \Rightarrow recherche déterministe
 - + recherche rapide, en temps borné
 - ensemble de nœuds dynamique ?
- Faiblement structuré : partiellement déterministe

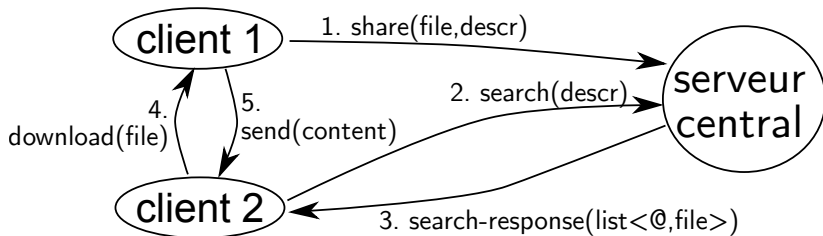


Classification

	Réseau virtuel		
	non structuré	faiblement structuré	structuré
contrôle centralisé	Napster		
partiellement décentralisé	eMule, FastTrack, Gnutella		
totalemment décentralisé	BitTorrent	Freenet	Chord, Pastry



Contrôle centralisé – Napster simplifié



- Un serveur central : l'annuaire
- Des clients pairs : stockage

Contrôle décentralisé – Gnutella

Chaque nœud est client, serveur et routeur

Messages

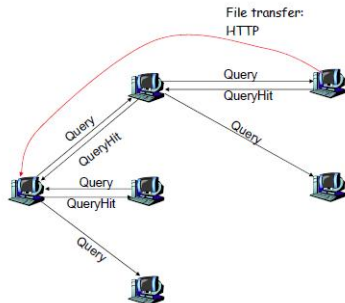
- ping : pour découvrir des correspondants
bootstrap : *gnutella caches* notoires puis par rebond
- pong : en réponse (+ informations : nb/taille des fichiers possédés)
- query : recherche (mots clefs)
- query-hit : en réponse (@ IP, id fichiers)

1. *On the Long-term Evolution of the Two-Tier Gnutella Overlay*, Amir Rasti, Daniel Stutzbach and Reza Rejaie. 25th IEEE Int'l Conf. on Computer Communications. April 2006.



Gnutella : recherche

- Recherche par inondation : requête query transmise de voisin en voisin
- Id unique : éviter les retransmissions en boucle
- nombre de retransmissions (hops) limité



Améliorations :

- Envoi aléatoire, effectué en parallèle
- Distinction entre nœuds feuilles (connectés à 2 ou 3 ultranœuds) et ultranœuds (puissants, fortement interconnectés) \Rightarrow nombre réduit de hops

(source : Original uploader was ACNS at en.wikipedia – Commons CC BY-SA 3.0)



Table de hachage répartie (DHT)

Table de hachage répartie = *distributed hash table*

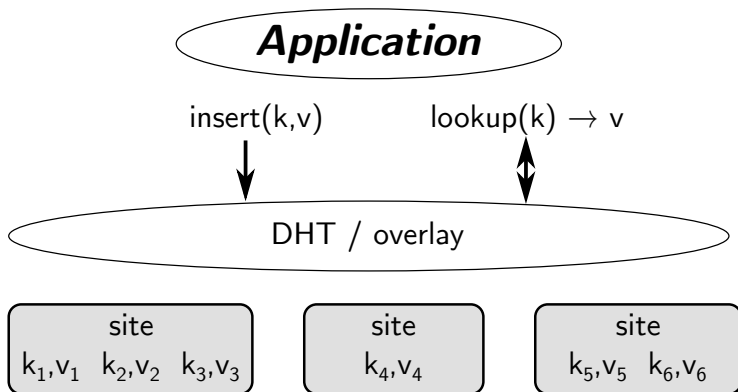


Table de hachage répartie (DHT)

L'infrastructure P2P établit le lien entre clef et site :

- Chaque site possède une ID : p.e. hachage de son adresse IP
- Chaque objet possède une clef et une valeur
- La clef d'un objet est p.e. le hachage de sa valeur, ou de son nom, ou de sa description. . .
- **Chaque site est responsable d'une partie de l'espace de hachage**, p.e. des clefs qui sont proches de son ID

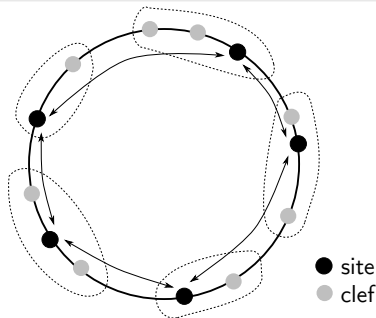
⇒

- Pas de connaissance globale centralisée
- Pas de point unique de défaillance
- Passage à l'échelle
- Répartition de la responsabilité



Exemple de DHT : Pastry

- Les sites sont organisés en anneau virtuel (rangé par ID)
- Chaque site connaît le suivant et le précédent
- L'espace des clefs est partagé : le site le plus proche est responsable d'une clef



Routage : trouver le nœud responsable d'une clef

`route(key, msg)` : acheminer le message (spécifique à l'application) au site en charge de la clef

1. *Pastry : Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems*, Antony Rowstron and Peter Druschel. Int'l Conf. on Distributed Systems Platforms. Nov. 2001.



Pastry : routage

Routage en suivant les liens suivants/précédents inefficace \Rightarrow table de routage

- N sites ($N = 16$ millions)
- Chaque site possède une ID : P digits sur B valeurs ($B = 16$, $P = \lceil \log_B N \rceil$). ex : 65a1fc
- Chaque site a une table de routage à P lignes, B colonnes
- La case (i, j) est l'adresse d'un site ayant les i premiers digits identiques au site, et j en $i + 1$ digit.

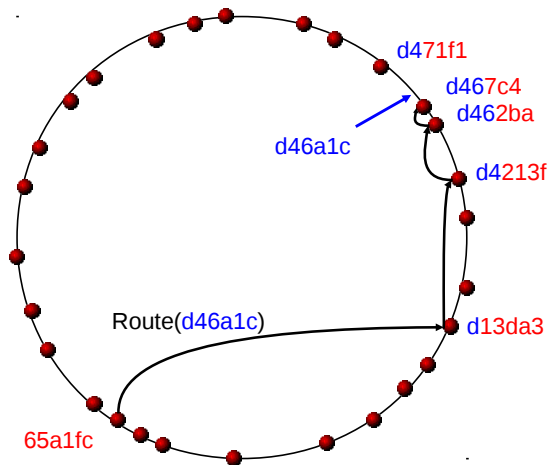
ex : sur le site 65a1fc, ligne 3, col 4 = site 65a4xx

L0	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
L1	60	61	62	63	64	65	66	67	68	69	6a	6b	6c	6d	6e	6f
L2	650	651	652	653	654	655	656	657	658	659	65a	65b	65c	65d	65e	65f
L3	65a0	65a1	65a2	65a3	65a4	65a5	65a6	65a7	65a8	65a9	65aa	65ab	65ac	65ad	65ae	65af
...																

- Trouver une clef = chercher le premier digit distinct et transmettre à ce site, qui poursuit le routage



Pastry : Routage



- max $\log_{16} N$ hops
- table de routage (par site) : $O(\log N)$

Dessin : Peter Druschel

Pastry : voisinage (*Leaf sets*)

Chaque site maintient les $L/2$ plus proches sites en deçà et en delà

- efficacité du routage
- résistance du routage
- détection de fautes (ping périodique)



Pastry : algorithme de routage

```
if (destination ∈ voisinage) then
    transmettre le message au membre concerné
else
    let  $l \stackrel{\Delta}{=} \text{longueur du préfix commun entre ce site}$ 
                                     et la destination
    let  $d \stackrel{\Delta}{=} \text{valeur du } l\text{-ième digit}$ 
    if routage[l,d] est défini et répond then
        transmettre le message à routage[l,d]
    else
        transmettre à un site qui
            - a au moins un préfixe commun de taille  $l$ 
            - est numériquement plus proche que ce site
```



Pastry : jonction

- 1 Le site d'id i veut s'insérer
- 2 Il envoie un message *join* à n'importe quel site
- 3 Ce message est routé comme précédemment au site j , actuellement responsable de i . Le site i sera inséré entre j et k (précédent ou suivant de j selon la valeur de i)
- 4 Le site j transmet à i son voisinage pour que i construise son voisinage initial
- 5 La table de routage de i est établie à partir des tables de j et k
- 6 Le site i interroge tous les sites de son voisinage initial :
 - i établit son voisinage définitif en gardant les $L/2$ plus proches dans chaque sens
 - Les sites du voisinage apprennent i et mettent à jour leurs propres table de routage et voisinage
- 7 Le site i devient actif

Seule difficulté : 2 insertions simultanées entre 2 même nœuds !



Pastry : départ (défaillance)

Les membres d'un voisinage s'échangent périodiquement des messages de vie.

Absence de réponse sur un message de vie ou un message de routage \Rightarrow considéré défaillant, enlevé du voisinage et de la table de routage si présent.

- Réparation du voisinage : augmenter son voisinage en demandant au site le plus loin dans son voisinage actuel
- Réparation du routage : obtenir la table des sites sur la même rangée que le site supprimé, puis en remontant



Pastry : bilan

Points positifs

- Passe à l'échelle, résistant à $L/2$ fautes simultanées (et en pratique bien plus), non centralisé (auto-organisation)
- Routage efficace $O(\log N)$ hops et messages en situation normale, $O(N)$ en cas pire (improbable)
- Information par site modeste $O(\log N)$

Améliorations

- Routage : prendre en compte la distance (en temps) pour établir la table de routage
- Réplication : un même objet clef/valeur est répliqué sur plusieurs nœuds voisins
- Sécurité : hachage non inversible avec peu de collision (SHA)

Conclusion

- Systèmes non structurés : performant si contrôle assez centralisé \Rightarrow capacité de croissance ? anonymat ?
- Systèmes structurés : dynamicité ?
- Contrôle décentralisé \Rightarrow qualité de service ?
 - sécurité, confiance ?
 - disponibilité non garantie (mais plutôt bonne)
 - site parasite ?
- Aucun standard, même architecturalement

