UNIVERSITE PIERRE & MARIE CURIE
LA SCIENCE A PARIS

Systèmes Pair-à-Pair

Olivier Marin
Laboratoire d'Informatique de Paris 6

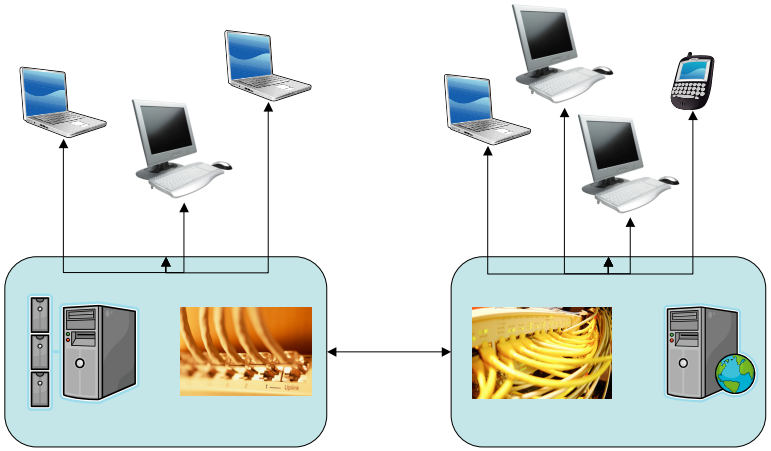
LIP6 Dpt : Réseaux et Systèmes Répartis

CNRS

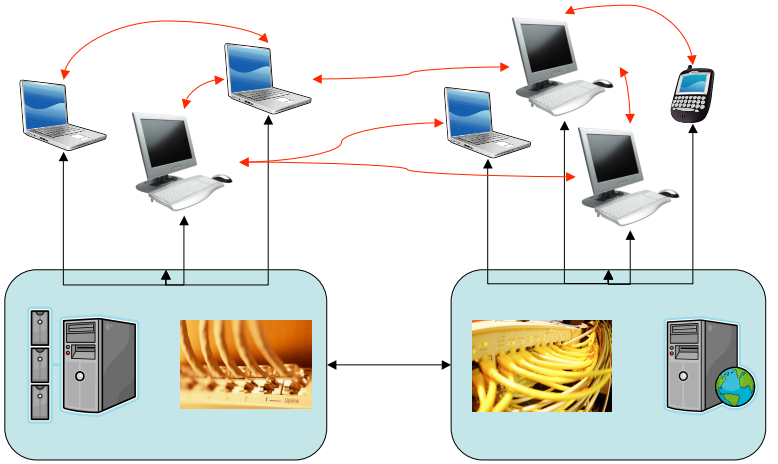
Systèmes Pair-à-Pair

- Concepts
- Exemples d'applications
- Infrastructure
  - Overlays structurés
  - Overlays non-structurés
- Problèmes classiques
- Perspectives (problèmes ouverts)

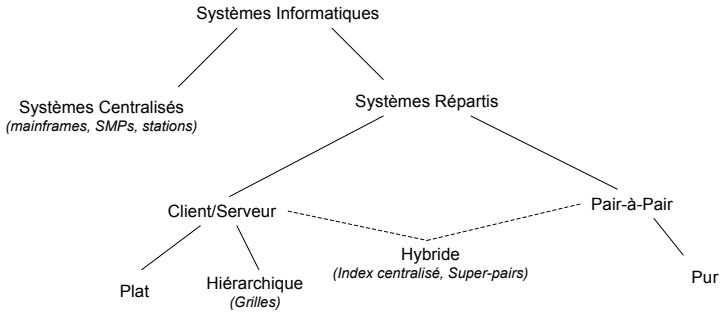
Réseaux Classiques



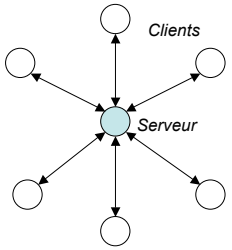
Réseaux P2P

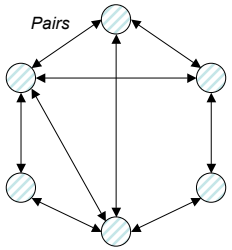


Taxonomie des Systèmes Informatiques



Client/Serveur vs. P2P





Client/Serveur vs. P2P

- Gérés
- Configurés
- Recherche de services
- Hiérarchique
- Ressources statiques
- Cycle de vie lié au serveur
- Centré IP
- Nommage basé sur le DNS
- Communications type RPC/RMI
- Synchrone
- Asymétrique
- Axé sur des modèles de liaison et d'intégration du langage de programmation (stub IDL/XDR, compilateurs, etc...)
- Sécurité de type Kerberos : acl, crypto

- Auto-Gérés
- Ad-hoc
- Découverte de services
- Maillage
- Ressources volatiles
- Cycle de vie autonome
- Non restrictif à IP
- Nommage spécifique
- Communication par messages
- Asynchrone
- Symétrique
- Axé sur la localisation de services, localisation du contenu, routage applicatif
- Anonymat, haute disponibilité
- Plus difficile à maîtriser

Objectifs du P2P

- Partage / réduction des coûts
  - ✓ Aggrégation dynamique de ressources volatiles
  - ✓ Autonomie totale
    - Système disponible 7/7-24/24
    - Maintenance nulle
    - Indépendance vis-à-vis de l'infrastructure physique
- Passage à l'échelle
  - ✓ Disponibilité de ressources
  - ✓ Suppression des goulots d'étranglement
- Anonymat (hum...)

P2P : Besoins algorithmiques

- Découverte de services (nom, adresse, route, métrique, ...)
- Recherche de voisins
- Routage de niveau applicatif
- Rémanence, récupération sur faute de liaison ou d'exécution

Applications P2P

Catégories

- Calcul Massivement Parallèle
- Collaboration
- Partage / Répartition de données

≈ Plates-formes

Applications P2P

Calcul Massivement Parallèle

But : Exécuter des programmes inexploitables autrement
Partage des ressources de calcul disponibles/inactives
Décomposition de l'application en micro-tâches parallélisables

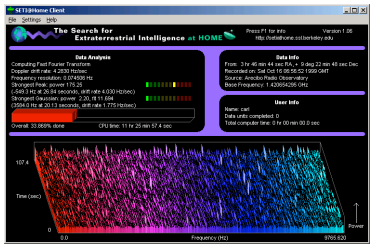
✓ Seti@home (astronomie)
✓ genome@home (ADN)
✓ folding@home (repliement des protéines)

- Calcul massivement parallèle
- Expérience en radioastronomie
  SETI : Search for Extra-Terrestrial Intelligence
  Analyse des données collectées par le radiotélescope d' Arecibo

- Exploite la puissance inutilisée des ordinateurs connectés via Internet
  Participants chargent et analysent les données durant la veille écran

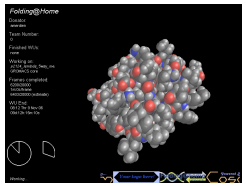
SETI@home
The Search for Extraterrestrial Intelligence

- 3.8M utilisateurs dans 226 pays
- 1200 années CPU / jour
- 38 TeraFlops soutenu (Le Earth Simulator Japonais obtient 40 TF)
- 1.7 Zetaflops (10^{21}) pour les 3 dernières années
- Très hétérogène :
  > 77 types de processeurs ≠



Folding@home distributed computing

- Calcul massivement parallèle
- Comprendre le repliement et l'agrégation des protéines
- Etude de maladies résultant d'un repliement anormal des protéines
  eg. Alzheimer, fibrose cystique, EBS (Vache Folle), nombreux cancers
- A dépassé le PetaFlops (10^{15}) ; supporte les PS3



Applications P2P

Collaboration

But : Mettre en relation des pairs par centre d'intérêt
Gestion d'annuaires, transport de données

- ✓Chat/Irc, NewsGroups
- ✓Instant Messaging (AIM, ICQ, Yahoo!Messenger, MSN)
- ✓Voice/IP (Skype)
- ✓MMORPGs (WoW, Ultima Online, Second Life)

Routage IP

- Collaboration
- Routeurs IP découvrent une topologie et la maintiennent
- Ne sont ni client ni serveurs
- Dialoguent continuellement entre eux
- Sont tolérants aux pannes
- Sont autonomes

- Collaboration
- Transfert de communications (VoIP)
- Pairs partagent leur bande passante
- Annuaire totalement décentralisé
- Routage au moyen de super-nœuds
- 246 millions d'utilisateurs
- Gros problèmes de sécurité

Applications P2P

Partage / Répartition de données

But : Partager des fichiers, des objets applicatifs, des services
Mise à disposition **volontaire** de données
Système de routage/localisation

- ✓ Napster, Publius, Freenet, MojoNation, FreeHaven, Groove, e-donkey
- ✓ Gnutella, Kazaa, BitTorrent
- ✓ Chord, Can, Pastry, Tapestry

Napster

- Partage de fichiers
- Système d'indexation centralisé ; fichiers restent sur les clients
  1. Connexion au serveur & chargement de la liste de fichiers (push)
  2. Envoi des critères de recherche sur la liste principale
  3. Sélection de la meilleure réponse (ping) puis téléchargement
- Considéré (à tort) comme le 1er réseau P2P
  SMTP
  UseNet News
  Archie : système d'indexation de serveurs FTP en accès libre
- Apparition des problèmes légaux (DRMs, ...)

- Partage de fichiers
- Contournement des déboires légaux de Napster
  ⇒ Décentralisation du système d'indexation
  ⇒ Chaque nœud est à la fois serveur et client (*servent*)
- Routage des requêtes en *best effort*
  ⇒ Inondation de proche en proche
  ⇒ Durée de vie prédéfinie pour chaque requête (TTL)
  ⇒ Identification unique et non ambigüe des requêtes

Applications P2P

Plates-formes

But : Permettre le développement de systèmes à large échelle
Mise à disposition de composants paramétrables et réutilisables
Noyau : services prédéfinis (*eg. sécurité, gestion de groupes*)

- ✓ JXTA
- ✓ Globus
- ✓ .NET My Services

JXTA (Juxtapose)

- Plate-forme de développement
- Spécification open source de protocoles P2P
  - ⇒ Pipes (canaux de comm.)
  - ⇒ Peer Groups (gestion de groupes)
  - ⇒ Rendezvous network (routage avec super-pairs)
- Initialement conçu par SUN pour Java
  - ⇒ Adaptations pour C / C++ / C# (.NET)

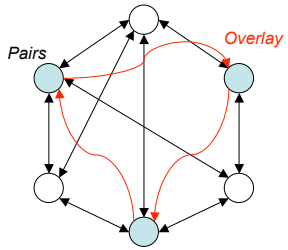
Infrastructures P2P

Overlays

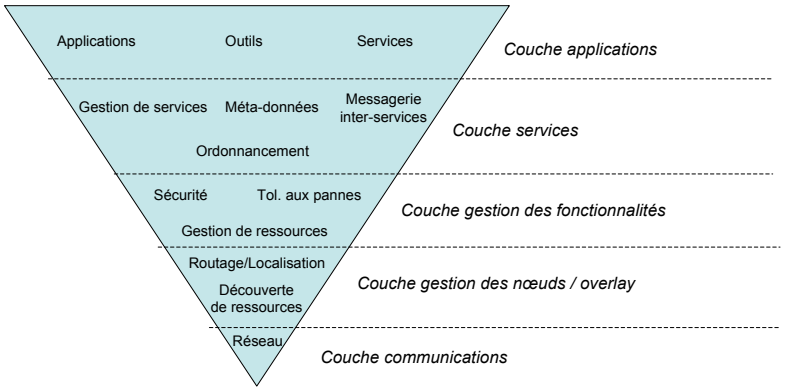
Infrastructures P2P

Définition d'un *overlay*

Réseau construit au-dessus du réseau physique
Ensemble des liens établis entre pairs qui se connaissent



Abstraction d'overlay



Overlays P2P

Non-structurés (Gnutella, Kazaa)

- Topologie du système déterminée par les utilisateurs
  Découverte du voisinage
- Placement ad-hoc des données dans le système
  Sans lien avec la topologie

Structurés (Chord, CAN, Pastry,Tapestry)

- Topologie particulière
  (eg. anneau, arbre, grille)
- Placement des données tient compte de la topologie
  Utilisation des fonctions de « hash »
  ⇒ Possibilité de déterminer l'***inexistence*** de réponse

P2P non-structuré : Gnutella

Protocole de recherche de données et services

Chaque nœud est à la fois client et serveur

Routage

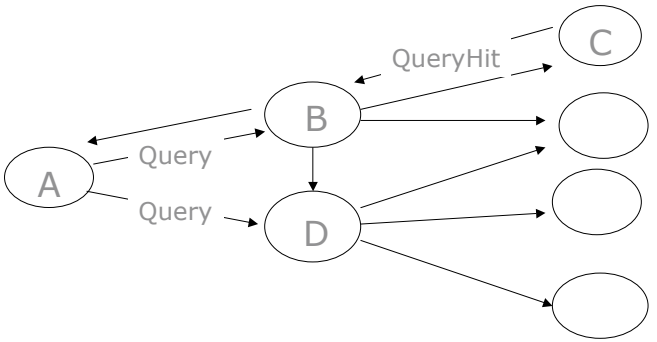
par inondation (flooding)

cheminement aléatoire (random walk)

Messages Gnutella (TimeToLive)

- Découverte de nœuds PING/PONG
- Découverte de données (fichiers) et services
  - Query
  - QueryHit

P2P non-structuré : Gnutella



Overlay semi-structuré

"Small-World Experiment" (Stanley Milgram, 1969)

Lettre distribuée aléatoirement à 150 personnes
- Omaha (Nebraska) et Wichita (Kansas) → centre des USA
- Contient des infos sur un dest. à Cambridge (Massachusetts) → côte ouest des USA

But : faire parvenir la lettre au destinataire
- Transmission à des personnes
  - susceptibles de connaître le destinataire
  - connaissances proches uniquement (*first-name basis*)
- Indication des expéditeurs successifs pour neutraliser les boucles

Résultats
- Nb moyen de transmissions = **5** (entre 2 et 10 hops)
- Passe très bien à l'échelle (287 millions d'habitants)
- Basé sur des réseaux de connaissances : **pas de centralisation**
- Fiable : transmission malicieuse ne fait que redémarrer la recherche

P2P semi-structuré : FreeNet

Stockage persistant de données et services

Nœuds/Données identifiés par une clé binaire (fonction hash)
- Identifiant de nœud : NodeID(utilisateur) = hash(@IP)
- Identifiant (*clé*) de fichier : FileID(fichier) = hash(contenu)

Gestion de tables de routage
- Construction de voisinages
  - ⇒ Connaissance approximative du contenu des nœuds proches

Types de requêtes

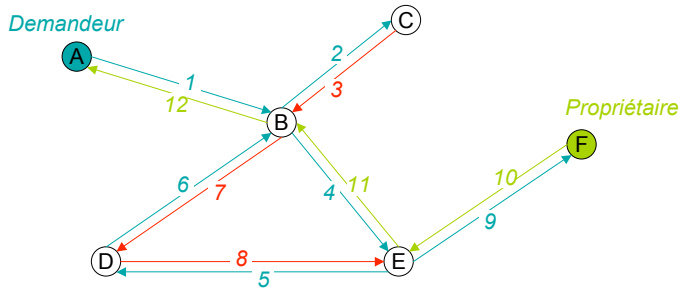
Join	Contact de nœuds connus : récup. d'un NodeID
Publish	Routage des données (FileID) vers le NodeID le + ressemblant
Search	Routage de requête à partir du FileID
Fetch	Récup. du fichier en cas de recherche fructueuse

P2P semi-structuré : FreeNet

Insertion de clé
- Diffusion d'un message routé vers le nœud tel que (key ~ nodeId)
  - msg d'insertion = clé + nb aléatoire de sauts (hop)
- Chaque pair contrôle si la clé est dans son système de stockage local
  - oui ⇒ génération d'une nouvelle clé
  - non ⇒ routée vers le nœud suivant (hop --) jusqu'à hop == 0
  - hop == 0 & pas de collision ⇒ clé insérée sur tout le chemin de routage

Données qui traversent un nœud sont copiées dans son cache
- Utilisation de la politique LRU pour la gestion du cache
- Information stockée pour chaque donnée (fichier)
  - code hash
  - dernier temps d'accès/modification

P2P semi-structuré : FreeNet



Séquence typique de routage de requête
- Dissémination de proche en proche
- Gestion de cul-de-sac (3) et de boucle (7)

Gnutella vs. FreeNet

- Routage basé sur la diffusion (flooding)
- Aucune mémoire du trafic véhiculé
- Read-only
- Système non sécurisé

- Routage dynamique basé sur la similarité des clés
- Tables de routage + Cache
- Read/Write
- Système sécurisé

P2P Hybride

Notion de super-pair

Pair "plus égal" que les autres

Auto-proclamé dynamiquement

⇒ Mécanisme d'acceptation/éviction au sein du voisinage

Création d'un niveau hiérarchique supérieur

⇒ Connaissance étendue du voisinage (données, clés, ...)

⇒ Serveur pour les pairs-pairs de ce voisinage

⇒ Prise en charge du routage avec les autres super-pairs

Adopté par Gnutella et KaZaA

P2P structuré : Motivations

- Faire mieux que les systèmes P2P ad hoc
- Guarantir le succès des localisations de noms
- Bornes démontrables sur les délais de recherche
- Preuve théorique du passage à l'échelle

P2P structuré : DHT

Distributed Hash Table

Table de hash : pierre angulaire de toute indexation

- Put (clé, valeur)
- Get (clé) → valeur
- Remove (clé)

Un identifiant global unique pour chaque nœud/fichier du système

Principe : répartir la table sur l'ensemble des nœuds

⁝ Pas de connaissance globale du système

Partitionnement de l'espace en propriétaires de clés

Redondance pour éviter la perte d'information

P2P structuré : CAN

Idée de conception

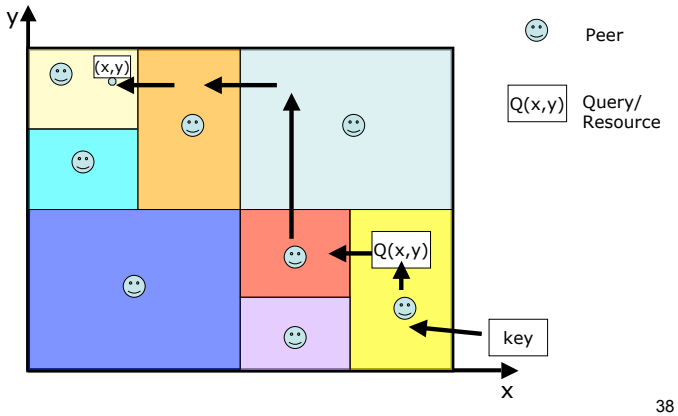
Espace cartésien virtuel bidimensionnel découpé en zones

Chaque nœud du système est propriétaire d'une zone
Connait les @ des propriétaires des zones adjacentes

Données stockées sous la forme (clé, val)
- hash(clé) → un point (x,y) dans l'espace virtuel
- (clé, val) stocké sur le nœud propriétaire de (x,y)

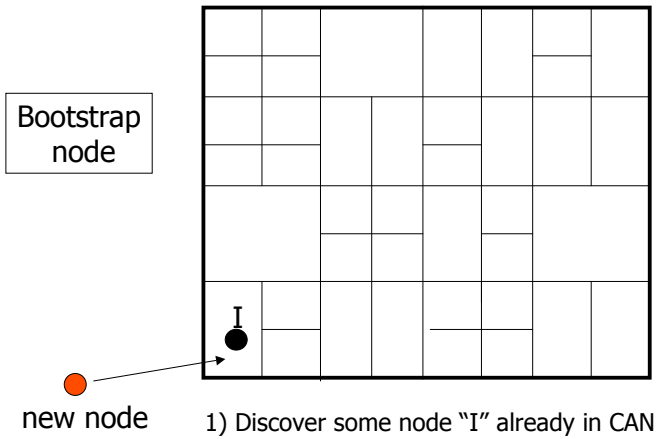
P2P structuré : CAN

Routage



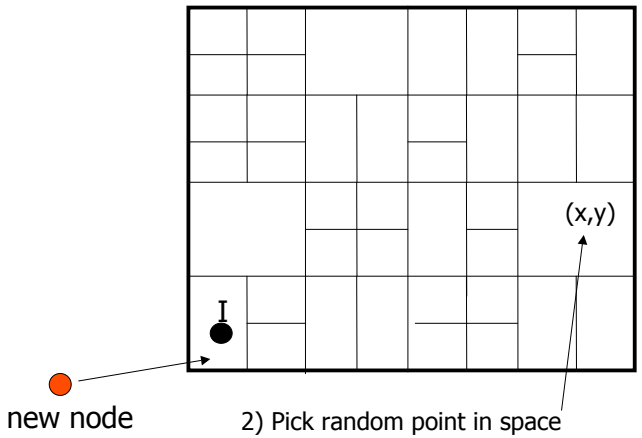
P2P structuré : CAN

Insertion de nœud



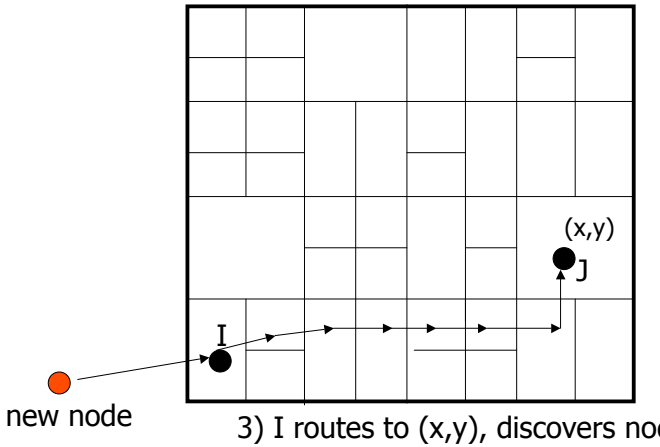
P2P structuré : CAN

Insertion de nœud



P2P structuré : CAN

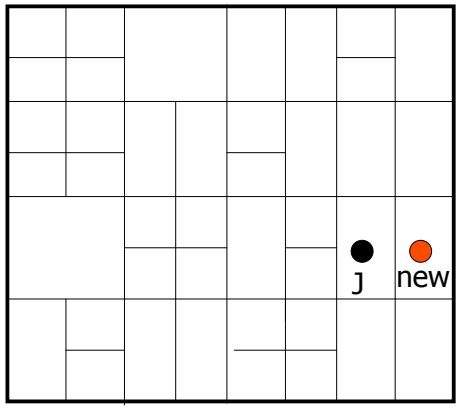
Insertion de nœud



3) I routes to (x,y), discovers node J

P2P structuré : CAN

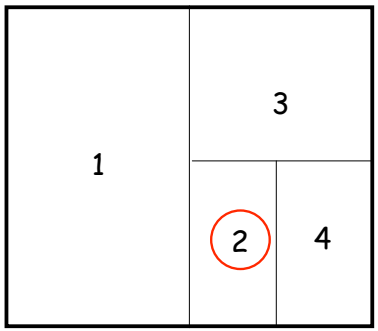
Insertion de nœud



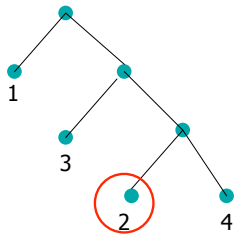
4) split J's zone in half… new owns one half

P2P structuré : CAN

Départ de nœud



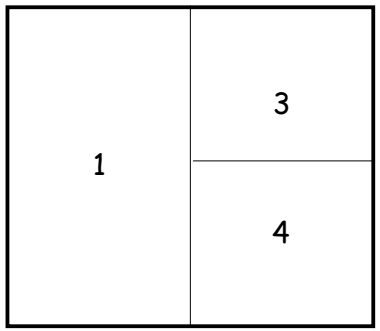
Espace virtuel



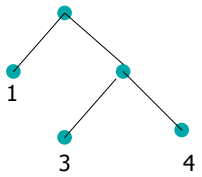
Arbre de partitionnement

P2P structuré : CAN

Départ de nœud



Espace virtuel



Arbre de partitionnement

P2P structuré : Chord

Une infrastructure de stockage et de routage

Identifiants sur m bits (2^m identifiants)

nodeID(nœud) = hash(@IP)

key(fichier) = hash(contenu)

L'espace des IDs est organisé en anneau

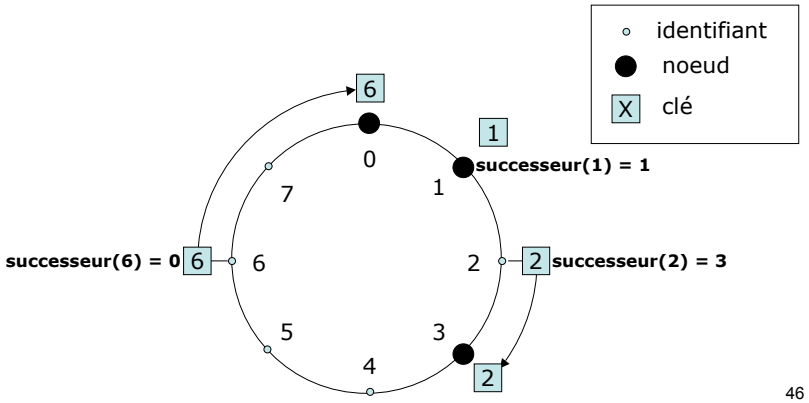
Un fichier de clé k (ou sa réf.) est stocké sur un nœud A tel que

- nodeID(A) > k mod 2^m
- Il n'existe pas de nœud x pour lequel

  { nodeID(x) > k mod 2^m } & { nodeID(x) < nodeID(A) }

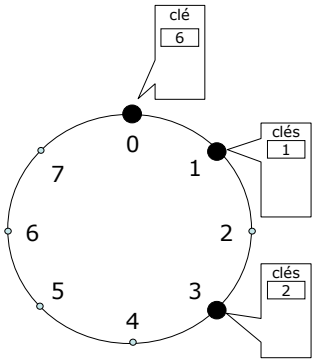
P2P structuré : Chord

Association clés - noeuds



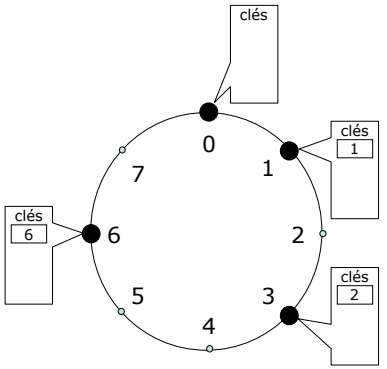
P2P structuré : Chord

Association clés - noeuds



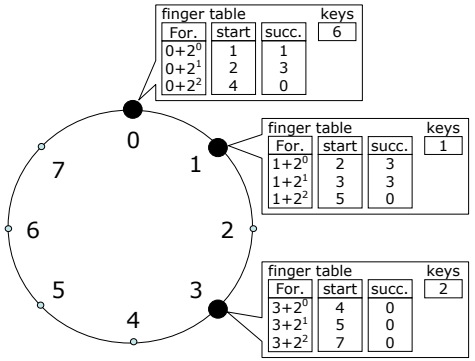
P2P structuré : Chord

Entrée du noeud 6



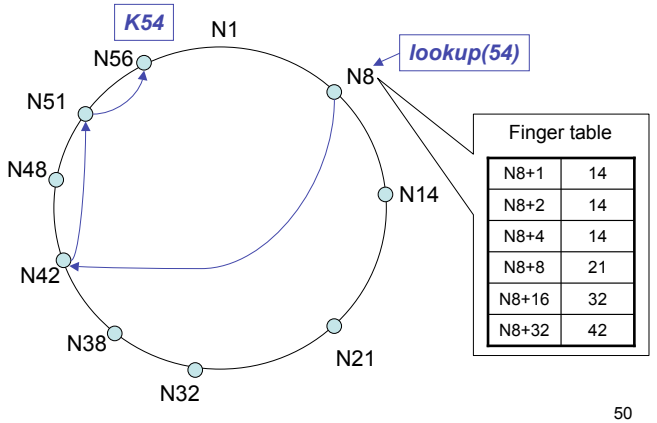
P2P structuré : Chord

Les raccourcis



P2P structuré : Chord

Recherche de la clé 54



P2P structuré : Chord

Caractéristiques significatives

- Mémoire utilisée par noeud en O(log(N))
- Temps de recherche d'une clé en O(log(N))
- Auto-reconfigurable
- Tolérant aux pannes

P2P structuré : Pastry

Une infrastructure de stockage et de routage

Mêmes principes de base que Chord

Identifiants sur m bits (2^m identifiants)

nodeID(nœud) = génération aléatoire

key(fichier) = génération aléatoire (*ou hash(contenu)*)

Topologie en anneau

Un fichier de clé k (ou sa réf.) est stocké sur son supérieur immédiat

Mais routage amélioré

Table de routage « incrémentale »

Voisinage logique (*leafset*) et physique (*neighbourhood set*)

P2P structuré : Pastry

Table de routage

Voisins logiques

Routage incrémental

Voisins physiques

NodeId 10233102

Leaf set

SMALLER		LARGER	
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

Routing table

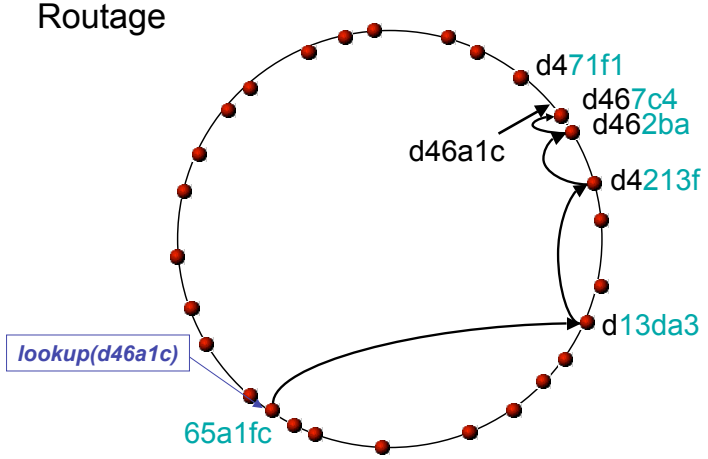
-0-2212102	1	-2-2301203	-3-1203203
0	1-1-301233	1-2-230203	1-3-021022
10-0-31203	10-1-32102	2	10-3-23302
102-0-0230	102-1-1302	102-2-2302	3
1023-0-322	1023-1-000	1023-2-121	3
10233-0-01	1	10233-2-32	
0		102331-2-0	
		2	

Neighborhood set

13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321

P2P structuré : Pastry

Routage



P2P structuré : Pastry

Caractéristiques significatives

- Mémoire utilisée par noeud en O(log(N))
- Auto-reconfigurable
- Routage optimisé (reste en *O(log(N))* et sécurisé
- Extrêmement tolérant aux pannes
  - Réplication des fichiers, détection de fautes dans le leafset
  - Table de routage prévoit les départs intempestifs

Applications P2P

Problèmes Classiques

Réplication de données

FreeNet

Données copiées par les nœuds qui participent à leur routage

MojoNation

Dissémination des copies les plus demandées par un serveur

CAN (multi-dimensionnel)

Une donnée peut avoir une clé par dimension

Past (Stockage de fichiers sur Pastry)

Données répliquées dans le leafset

Sécurité P2P

"On the Internet, nobody knows you're a dog."

Anonymat (majoritairement Freenet)

- ✓ Eviter les cx directes entre le demandeur d'information et le propriétaire
- ✓ Mensonge aléatoire au cours du routage
  → Mandataire prend la place du demandeur / propriétaire
- ✓ Utiliser de TTL choisis aléatoirement
- ✓ Dissocier le propriétaire d'un document de sa localisation (CAN, Chord)

Sécurité P2P

Intégrité des données

Vérification des données

Clés cryptographiques (CFS, Past)

Dissémination des données

Fichiers à stocker décomposés en n blocs

Mais m blocs (m<n) suffisent pour reconstituer le fichier

(Publius, Mnemosyne, FreeHaven)

Sécurité P2P

« Free-riding »

Un ou +sieurs utilisateurs profitent du système sans partager leur ressources

Problème : écroulement du système

Solutions :

- ✓ Utilisation de techniques d'incitation à la participation
  - Découverte des ressources du système proportionnelle à la participation
  - Paiement virtuel ou micro-paiement (MojoNation)
- ✓ Surveiller les pairs

Sécurité P2P

Collusion : « SYBIL attack » [Douceur 2002]

Un utilisateur peut entrer dans le réseau en utilisant plusieurs identités
Problèmes : attaques malicieuses
Monopolisation des copies d'un même fichier
Falsification du routage

Solutions
✓ utopique : identification unique des ressources d'un nœud
✓ à double tranchant : insertion coûteuse (paiement, computing challenge)

Stockage Persistant

Spécification

Persistance
Chaque objet est répliqué plus d'une fois dans le système

Atomicité
Chaque opération de lecture renvoie la dernière valeur écrite
L'ordre causal des événements est respecté

Stockage Persistant

Abstraction
Vivacité (Convergence)
Invocation pour un objet « id » renvoie, au bout d'un temps fini, un ensemble non vide de nœuds qui stockent les répliques de l'objet
Sûreté (Quorum)
2 invocations ≠ pour le même objet aboutissent à des résultats dont l'intersection est non-vide
Equilibrage de charge
2 invocations réalisées par le même utilisateur pour des objets ≠ renvoient avec une forte probabilité des ensembles de nœuds distincts

Implémentations (partielles)
Structurés : Malkhi et al (graphes de Bruijn), Naor et al. (diagrammes de Voronoy)
Non-structurés: Lynch et al. (RAMBO I, II, III), Geoquorums (version adhoc), SAM: Self-adjusting Atomic Memory (I-SPAN 2005)

Publication/Abonnement

Spécification

Légalité
Si un nœud p est notifié avec un événement « e »
alors p est abonné avec une condition « f » satisfaite par « e »

Validité
Si un noeud est notifié avec un événement « e »
alors il existe un nœud dans le réseaux qui a publié « e »

Vivacité - Evénement
Si un événement « e » survient
& si un nœud p est abonné avec une condition « f » satisfaite par « e »
alors p est notifié au bout d'une temps fini

Equité
Chaque nœud a le droit de publier infiniment souvent

Publication/Abonnement

Abstraction

Toute invocation avec un événement “e” renvoie l’ensemble connecté minimal de brokers pour les conditions “f” tel que “e” satisfait “f”

Implémentations

Systèmes P2P structurés : Pastry (ex. Scribe, SPLIT-STREAM), CAN (Meghdoot)

Systèmes P2P non-structurés : Gossip, Structures de filtrage

Problèmes ouverts

- Connectivité et tolérance aux fautes des structures de filtrage
- Réduction du nombre de nœuds qui reçoivent des événements non-désirés

Résolution de Conflits

Spécification

Sûreté

Accès en exclusion mutuelle aux objets

Vivacité

Chaque invocation du service comporte une réponse au bout d’un temps fini

Résolution de Conflits

Abstraction

Registre Test&Set/Reset ou exclusion mutuelle (Sûrete)

Service de location de verrous (Vivacité + Equité)

Implémentations

Systèmes P2P structurés : problème ouvert

Systèmes non-structurés

- A fault-tolerant token based mutual exclusion using a dynamic tree (EuroPar 2005)
- Stabilizing Mobile philosophers (IPL 2005)

Problèmes ouverts P2P

- Mise en place de travail coopératif
  (eg. Wikipedia, CVS, éditeurs de texte)
- Allocation de ressources
- Observation / Evaluation de performances
- Construction de systèmes de confiance
- Traitement des problèmes de sécurité