Réplication de Données dans les Réseaux Peer-to-Peer (P2P)

Abdessamad Imine

Université de Lorraine & LORIA-INRIA Grand-Est

Abdessamad.Imine@loria.fr

Sommaire

- Présentation du Peer-to-Peer (P2P)
- Techniques de Réplication
- Réplication Optimiste
- Transformées Opérationnelles

Sommaire

- Présentation du Peer-to-Peer (P2P)
- Techniques de Réplication
- Réplication Optimiste
- Transformées Opérationnelles

Comment P2P est apparu ?

- Internet a été conçu comme un système P2P
  - Libre échange des données
    - Sans firewall, sans NAT, sans connexion asymétrique
  - Applications telnet, ftp ouvertes à tous
  - Coopération
    - sans spam et sans abus de bande passante
- Usenet News, DNS, protocole de routage (RIP, OSPF) : vieux modèles de P2P
- Emergence du P2P = renaissance du modèle originel d'Internet, au niveau applicatif

Connaissance

- Logiciels P2P de type file sharing
  - Napster
  - Gnutella
  - Edonkey
  - Kazaa
  - BitTorent
- Echange possible de musique ou de film
  - Souvent de manière illicite
- Mais d'autres protocoles et logiciels P2P
  - Manière forte et équitable de **collaborer en vue d'accroître le potentiel du réseau**
    - Seti@home
    - Groove
    - Jxta

Définitions

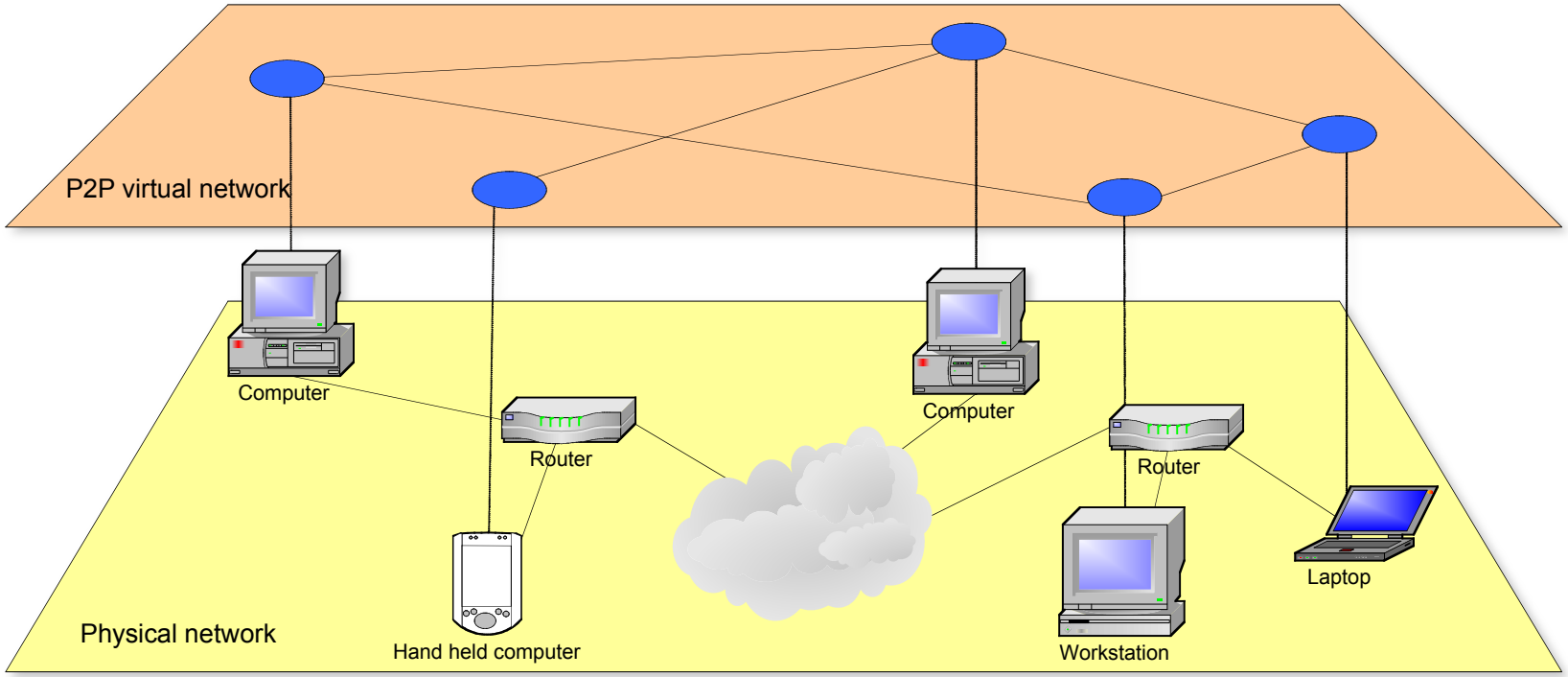
- « **Peer-to-peer** computing is the **sharing of computers resources and services** by direct exchange **between systems** »
- « **Peer-to-peer** refers to a class of systems and applications that employ **distributed resources** to perform a critical function in a **decentralized manner** »
- Échange direct des ressources et services entre ordinateurs, chaque **élément** étant **client** et **serveur**
  - Utilisation maximale de la puissance du réseau
  - Elimination des coûts d'infrastructure

➔ Retient l'attention de la recherche, des développeurs et des investisseurs

Définitions

- Eléments constituant un réseau P2P peuvent être de nature hétérogène
  - PC
  - PDA
  - Téléphone portable
- Réseau P2P présente une topologie virtuelle
- Nature « ad hoc »
  - Eléments constituants peuvent aller et venir
  - Topologie du réseau pas stable

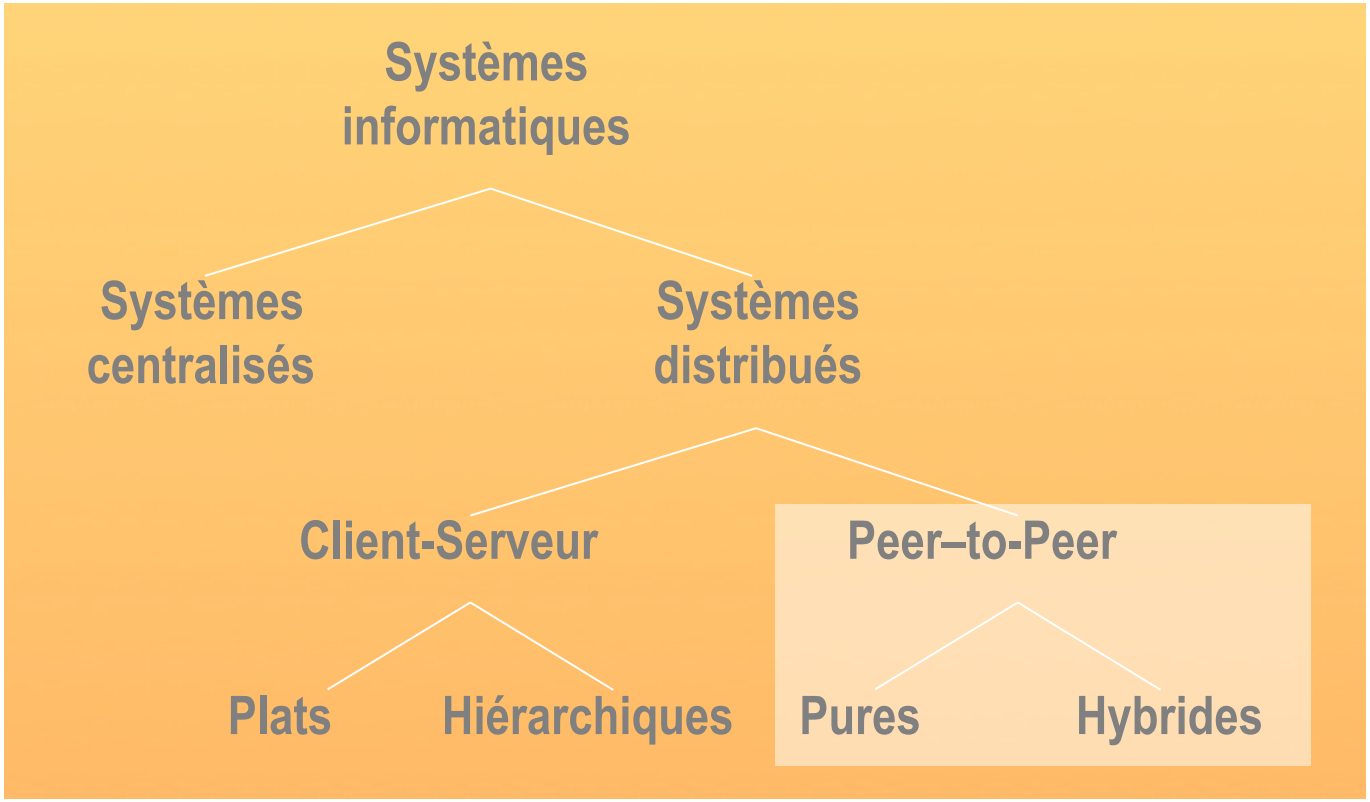
Topologie Virtuelle



Objectifs

- Modèle P2P est très général
  - ➔ Applications de nature différente, objectifs variés :
  - Partage et réduction des coûts entre les différents peers
  - Fiabilité (pas d'élément centralisé)
  - Passage à l'échelle (évite les goulots d'étranglement)
  - Agrégation des ressources (puissance de calcul, espace de stockage)
  - Accroissement de l'autonomie, chacun à la responsabilité de partager ses ressources
  - Anonymat pouvant être assuré par des algorithmes de routage ne permettant pas le pistage d'une requête
  - Communication ad-hoc et exhaustive

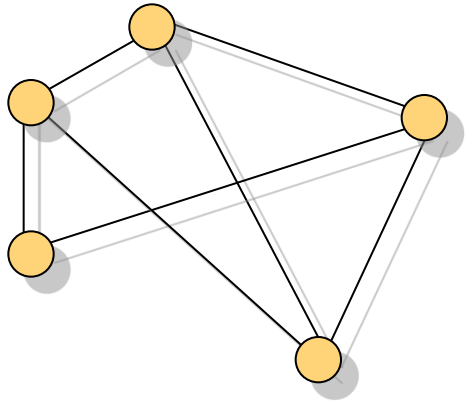
Taxonomie des architectures



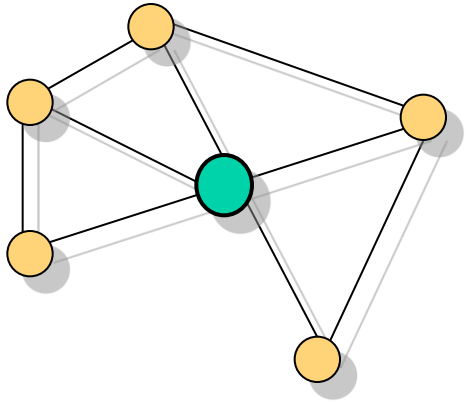
Classification des systèmes informatiques

Classification des systèmes P2P

- Modèle pur
  - Pas de serveur centralisé
    - Gnutella
    - Freenet



- Modèle hybride ou centralisé
  - Un serveur est contacté pour obtenir des méta-informations
    - identité du peer sur lequel sont stockées les informations
    - Vérifier les crédentials de sécurité
  - Echange réalisée en P2P
    - Napster
    - Groove



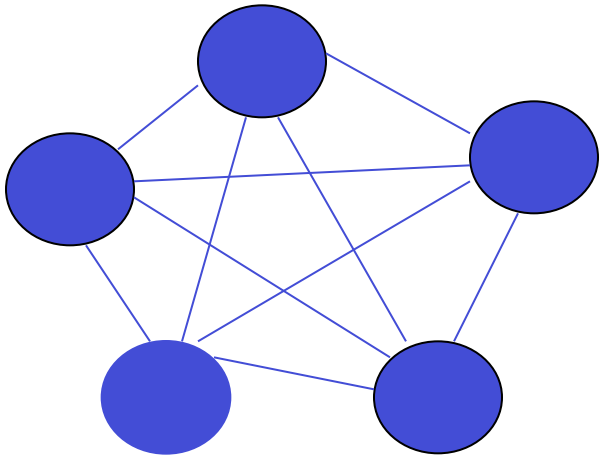
Classification des systèmes P2P

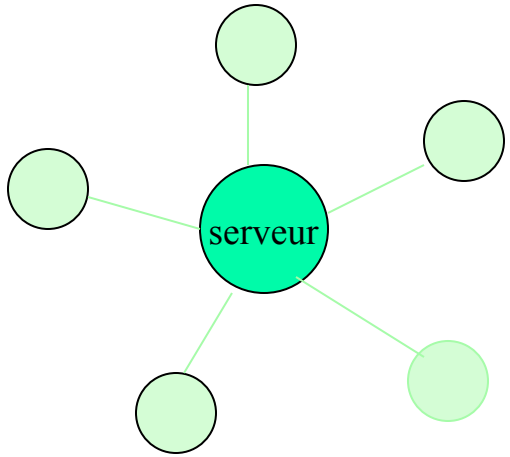
- Modèle super-peer, qui est un modèle hybride
  - Les super-peers contiennent des informations que les autres peers n'ont pas
  - Les autres peers ne consultent ces super-peers que s'ils ne peuvent trouver l'information autrement
  - Kazaa, FastTrack, …
- AUTRES CLASSIFICATIONS :
  - Classification applicative
  - Classification technologique (stockage et contrôle des données, usage des ressources, contrôle de l'état global, contraintes de qualité de service)
  - Classification architecturale (identité, découverte, authentification, autorisation)

Caractéristiques

- Localisation des fichiers dans un environnement distribué
- Meta-données ou index du réseau P2P
- Libre circulation des fichiers entre systèmes
- Capacité de connexion variable suivant les modèles
- Echanges d'informations non sécurisés
- Peers non sûrs
- Aucune vue globale du système

P2P vs. Client-Serveur





Auto-organisé
Evolution dynamique, ad-hoc
Découverte des peers
Flux distribué
Symétrie du réseau
Communication par messages
Adressage dynamique au niveau appli.
Entités autonomes
Attaques difficiles (mobilité, anonymat)

Gestion centralisée
Configuration statique
Consultation de tables
Flux centralisé
Asymétrie du réseau
Orienté RPC
Adressage statique @IP
Entités dépendantes
Attaques plus simples

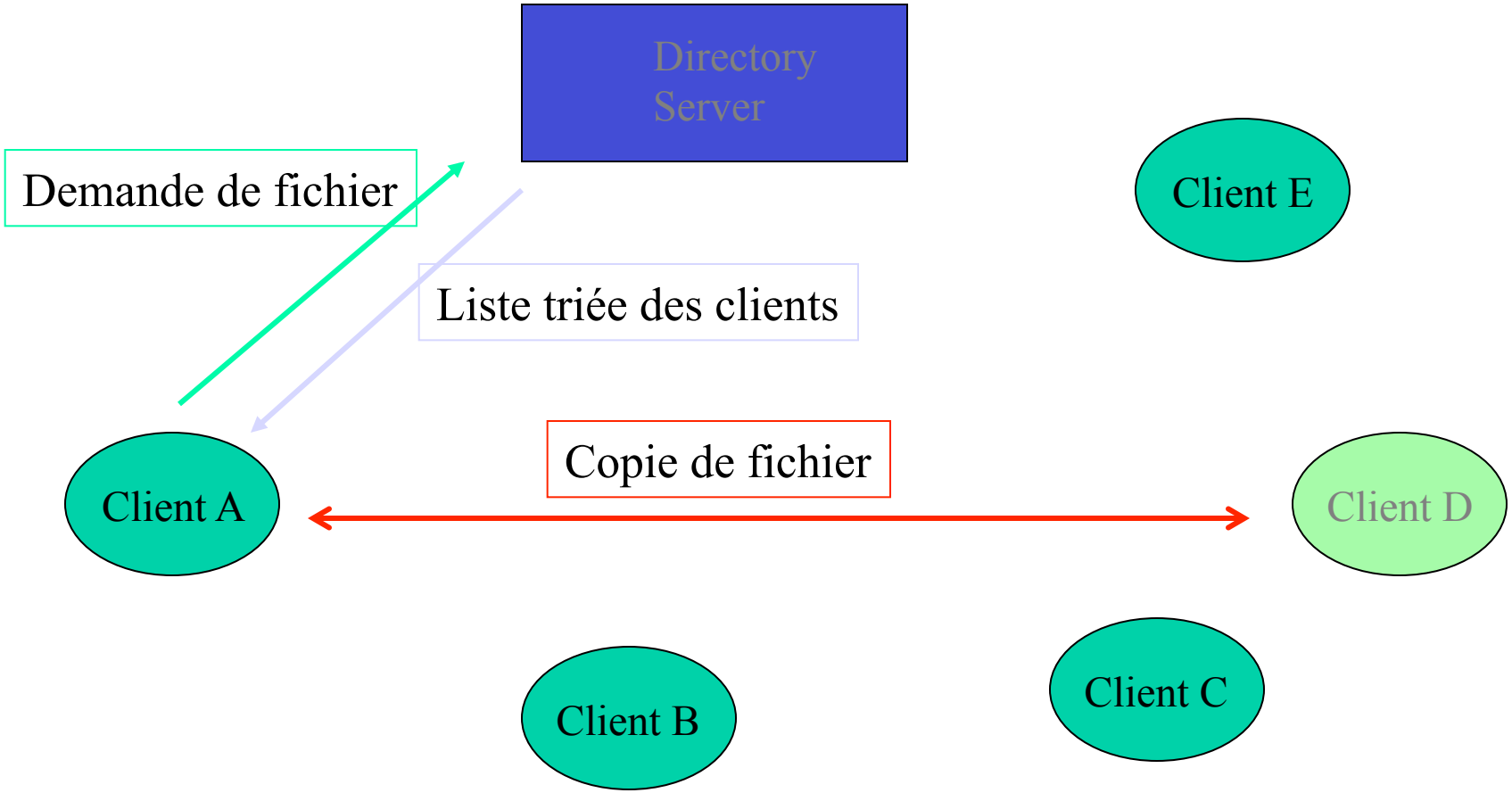
Services à fournir en P2P

- Découverte des peers
- Algorithme de l'organisation P2P
  - Au niveau réseau
  - Au niveau application (groupe d'intérêt)
- Capacité de publication
- Composants d'identification
- Facilité de communication entre peers

Napster

- Partage de fichiers mp3
- Fondé en mai 1999 par Shawn Fanning et Sean Parker
- Arrêté en septembre 2001, après poursuite judiciaire
- Application phare : 40 millions de téléchargements, 160 000 utilisateurs en moyenne
- Répertoire centralisé pour les données administratives et les index des fichiers mp3 téléchargeables par les peers

Echanges dans Napster



Bilan Napster

Les limites

- Pas d'anonymat partout
  - Vous êtes connus du serveur…
  - … Et des peers sur lesquels vous téléchargez
- Limites habituelles d'un serveur central
  - Disponibilité
  - Passage à échelle
    - Saturation de la bande passante
    - Saturation du nombre de processus
- Mauvaises informations du débit des peers pour ne pas être sollicités

Les avantages

- Avantages habituels d'un serveur central
  - Facile à administrer
  - Facile à contrôler, à fermer
- Evite les recherches coûteuses sur le réseau
  - Pas de routage
  - Planification de la gestion des utilisateurs
- Tolérance aux fautes
  - Par un sondage régulier des peers connectés, état cohérent
- Service novateur
  - Généralise le P2P
  - Généralise les procès contre le non-respect des droits d'auteur

Gnutella

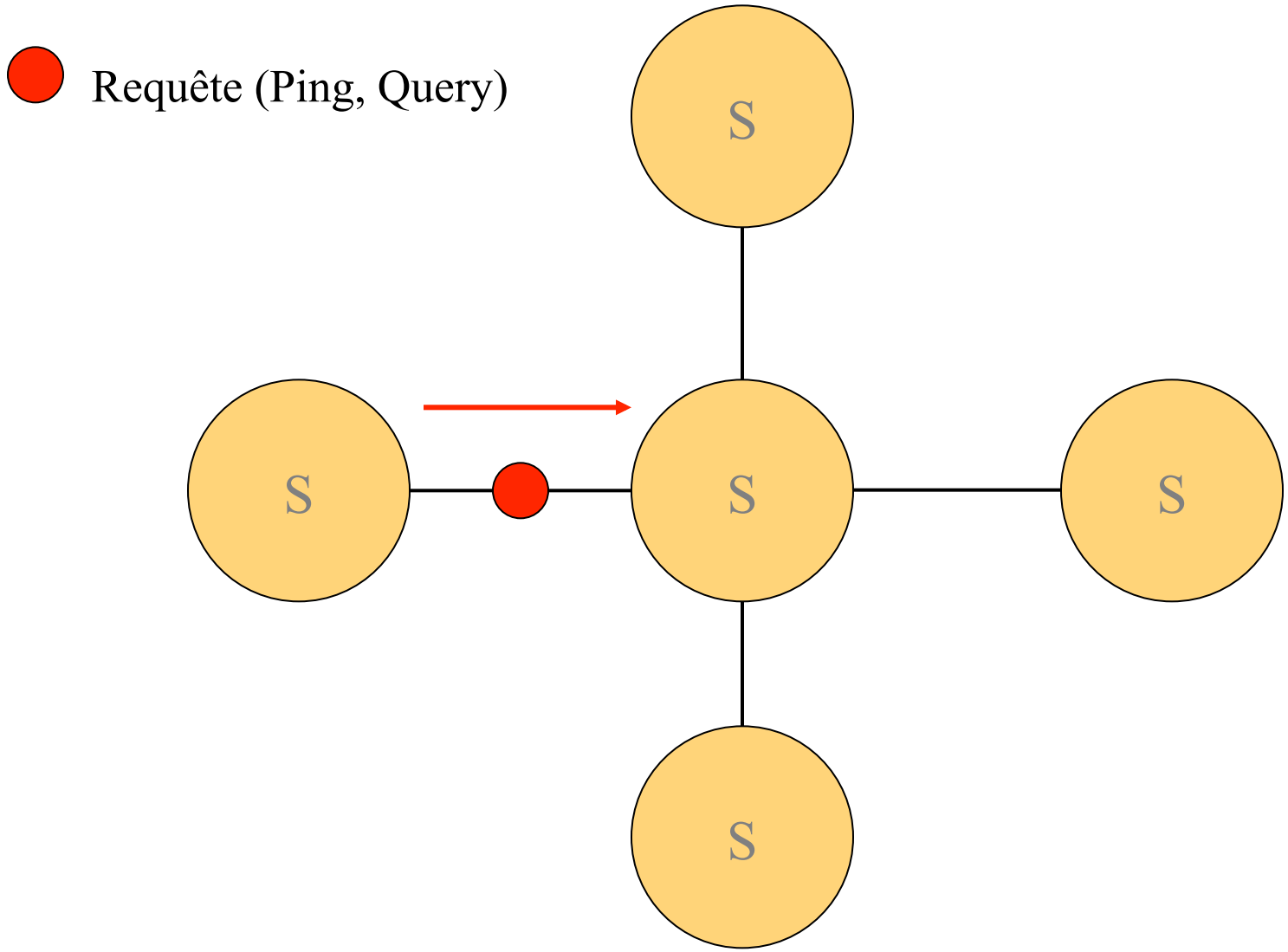
- Protocole de fichiers partagés
- Version 0.4 , en mars 2000, développé en une quinzaine de jours par Justin Frankel et Tom Pepper (Winamp)
- Plusieurs dizaines de milliers d'utilisateurs simultanés
- Réseau complètement décentralisé, modèle pur
- Servent = Serveur + Client
  - Chaque élément joue à la fois le rôle de client et serveur
- Messages : informations émises par les servents à travers le réseau Gnutella

Principaux messages de Gnutella

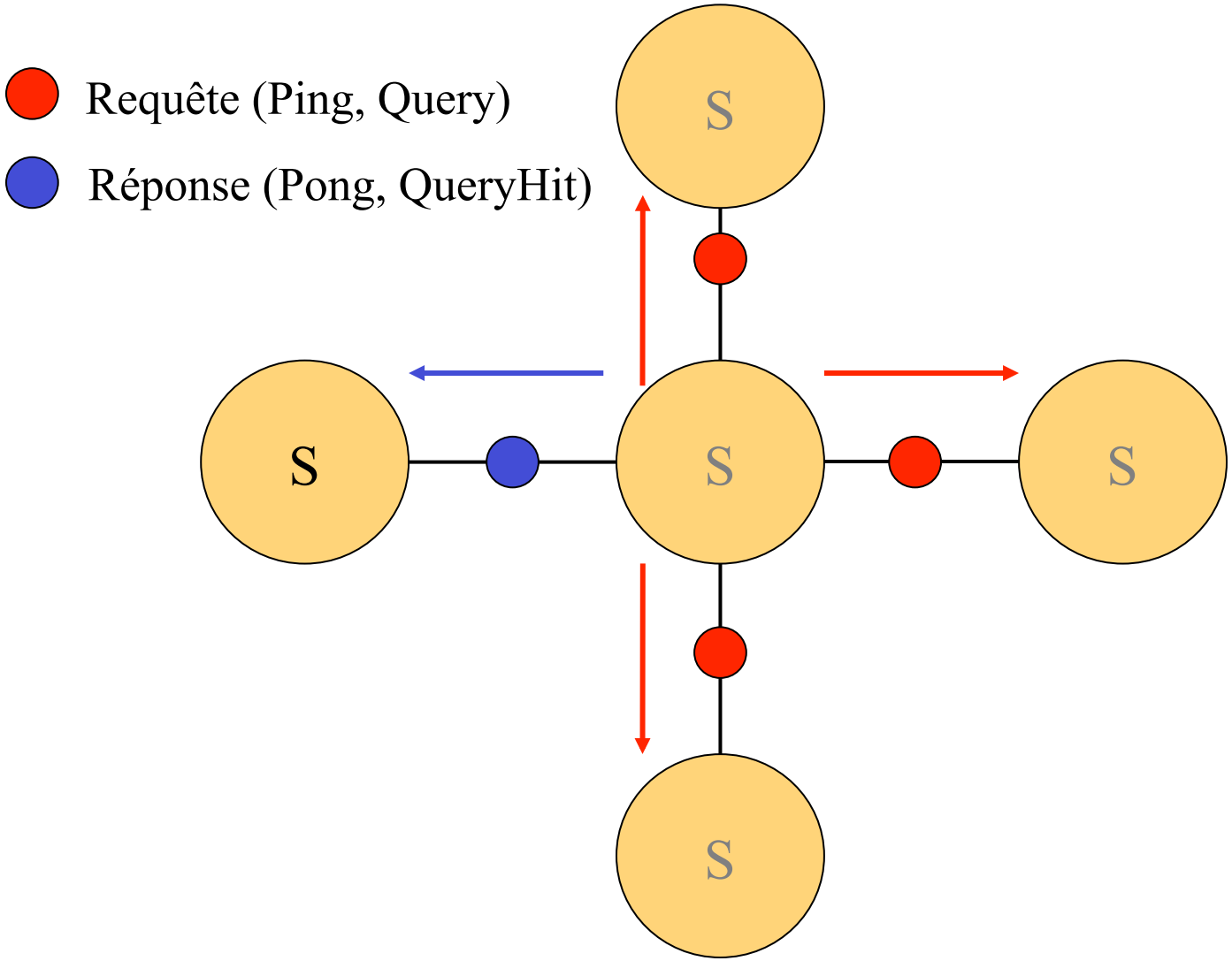
Message	Description
Ping (0x00)	Utilisé pour découvrir les autres servents sur le réseau Un servent qui reçoit un Ping doit répondre avec un (ou plusieurs) Pong
Pong (0x01)	Réponse à un Ping, contient : • Adresse IP + n° de port du servent • Le nombre et la quantité de données partagées
Query (0x80)	Utilisé pour la recherche d'un fichier
QueryHit (0x81)	Réponse à un Query
Push (0x40)	Permet de télécharger des données depuis un servent situé derrière un firewall

Si les deux servents sont derrière un firewall, le téléchargement est impossible

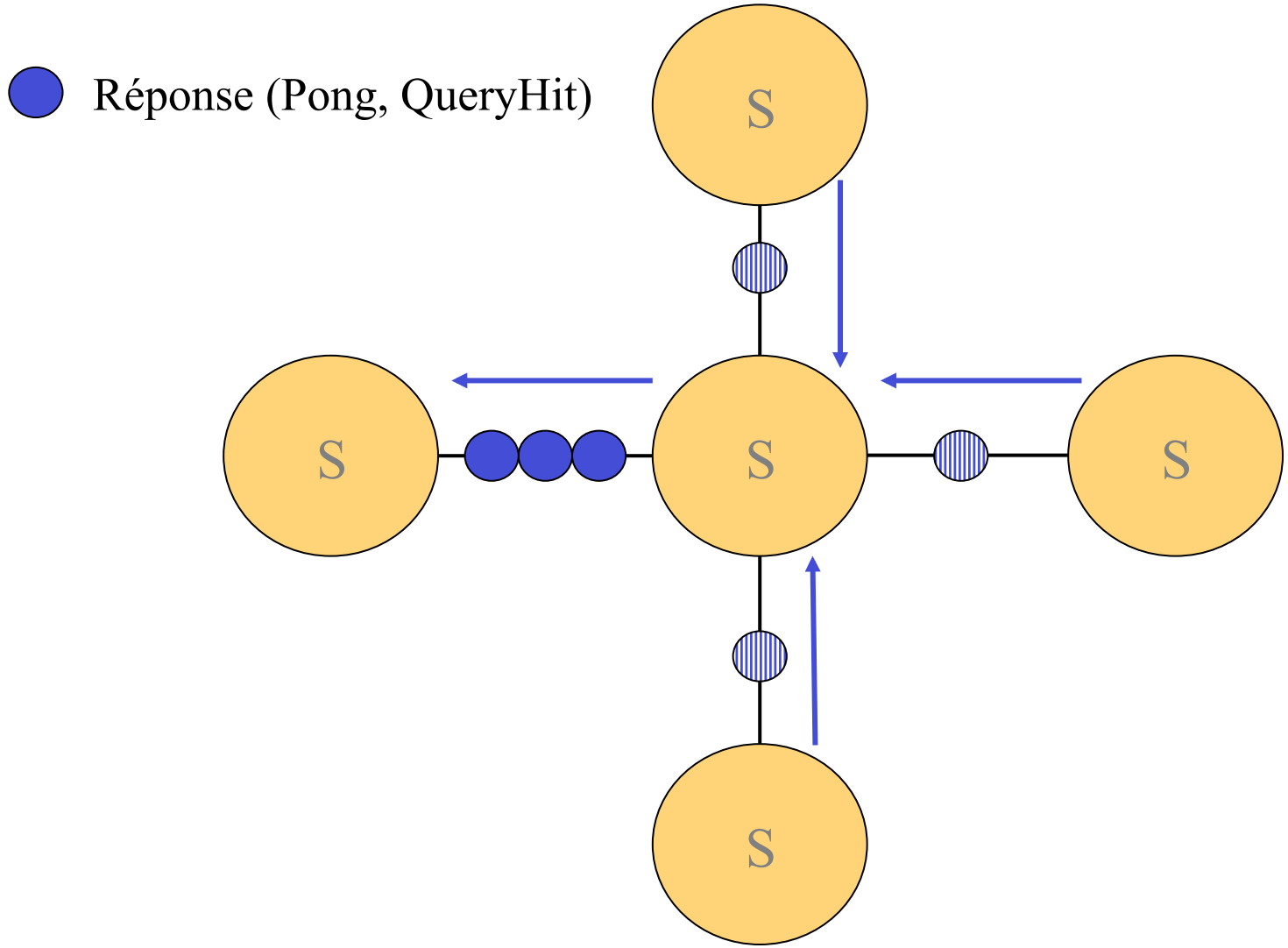
Routage de messages dans Gnutella



Routage de messages dans Gnutella



Routage de messages dans Gnutella



Bilan Gnutella

Limites

- Réseau rapidement inondé par des ping-pong
- Message push continuellement envoyé, si pas de réponse
- Manque de fiabilité dans ses requêtes

Avantages

- Administration simple, car mutualisée
- Topologie évolutive
- Disponibilité du réseau

Superpeers

- Caractéristiques
  - éléments du réseau choisis en fonction :
    - de leur puissance de calcul
    - de leur bande passante
  - réalisant des fonctions utiles au réseau
    - indexation des informations
    - le rôle d'intermédiaire dans les requêtes
- Ils rendent le réseau un peu moins robuste
  - les cibles à "attaquer" pour que le réseau devienne inopérant sont moins nombreuses que dans un réseau de type Gnutella
- Dans les réseaux FastTrack, comme Kazaa

Recherche : localisation, routage et partage de fichiers

- Limitation du modèle pur, type Gnutella, dû à l'utilisation du TTL
- Fournir des solutions P2P de stockage, de recouvrement de données fiables
- Séparation entre :
  - problème de localisation et de routage (dans un environnement distribué dynamique)
  - application de partage de fichiers

Table de Hachage

- Des protocoles optimisés, basés sur les tables de hachages distribuées (DHT Distributed Hash Table), permettant de réaliser des recherches en un nombre de messages croissant de façon logarithmique en fonction du nombre éléments du réseau :
  - CAN
  - Chord
  - Freenet
  - Tapestry
  - Pastry
  - Symphony

Sécurité d'un réseau P2P

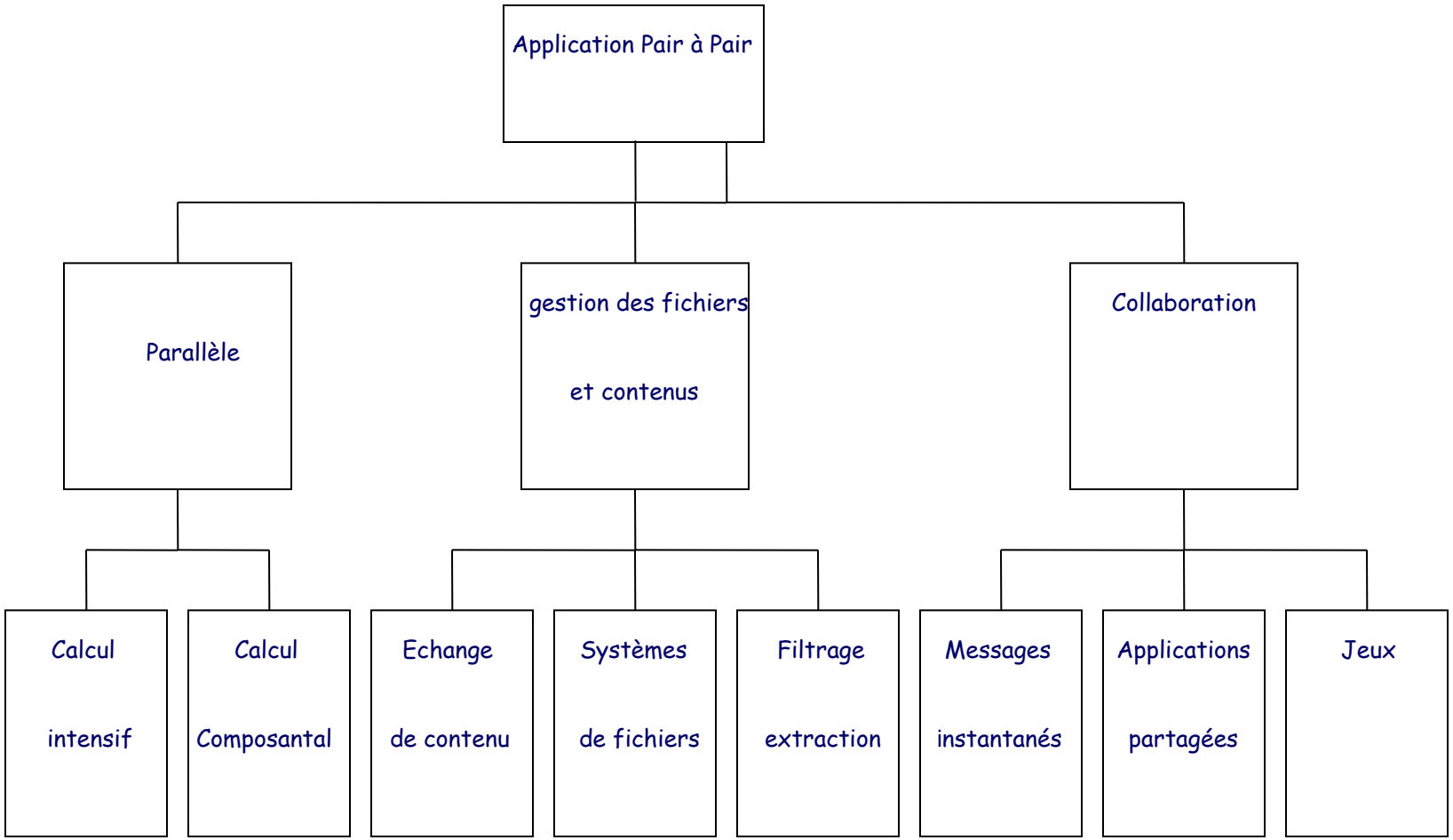
- Réseau P2P constitué de peers inconnus

Potentiellement dangereux

➔Mise en œuvre de mécanismes de sécurité

- Cryptage des données avec clés multiples
- Sandboxing (exécuter une application dans un environnement clos)
- Gestion des droits d'utilisation
- Facturation
- Protection des hôtes par des pare-feux
- IPV6 authentification et intégrité

Applications



Applications parallèles

- Utiliser les machines oisives d'un réseau pour effectuer les calculs
- Découper un gros calcul en petites unités sur un grand nombre de peers:
  - Calcul intensif : calcul d'une même opération munie de paramètres différents
    - SETI@Home
    - Genome@Home

  - Calcul composantal : découpe un même calcul en petites unités indépendantes que l'on peut assembler pour effectuer le calcul complet

Grilles de calcul

- Les grilles de calcul se rapprochent du P2P pour en utiliser l'architecture
  - Le Global Grid Forum (GGF) a rejoint en avril 2002 le P2P Working Group (Avaki, Lattice)
  - Les travaux portent sur le partage de mémoire et d'adressage global à très grande échelle

AVAKI

ILDG

Gestion des fichiers et contenus

- Stocker et retrouver des informations sur différents éléments du réseau
- Echange de contenu
  - Napster, Gnutella, Morpheus, Kazaa
  - Freenet (anonymat des sources et intégrité des documents)
  - BitTorrent (transfert parallèle de plusieurs sources, possibilité d'arrêt et de reprise de transfert)
- Etablir un système de fichiers distribué dans une communauté
  - PAST, OceanStore (plus de 6 millions d'utilisateurs)
- Bases de données et tables de hachage distribuées
  - Mariposa, Chord

Collaboration

- Collaborer en temps réel entre usagers sans utiliser de serveur central
  - Messages instantanés
    - Yahoo
    - AOL
    - Jabber
  - Travail coopératif ou applications partagées, permettant de travailler de manière commune sur un projet distribué
    - Groove
    - Magi
    - Powerpoint distribué
    - Projet de CAO
  - Commerce électronique
  - Jeux en réseau, dont l'architecture est exempte de toute autorité centrale
    - DOOM

Plates-formes

- Infrastructures génériques pour développer des applications P2P
- Assurent les composants primaires du P2P :
  - Gestion des peers
  - Nommage
  - Découverte de ressources
  - Communication entre peers
  - Sécurité
  - Agrégation des ressources
- Les plates-formes
  - JXTA
  - .net (Microsoft)
  - Anthill (Construite sur Jxta, soutenue par SUN)

Sommaire

- Présentation du Peer-to-Peer (P2P)
- Techniques de Réplication
- *Réplication Optimiste*
- Transformées Opérationnelles

Objectifs de la réplication

- Avantages
  - Accès simplifié, plus performant pour les lectures
  - Résistance aux pannes
  - Parallélisme accru
  - Evite des transferts
- Inconvénients
  - Overhead en mise à jour
  - Cohérence des donnés
- Terminologie
  - Réplica est une copie d'un objet (texte, XML, table BD, …)
  - Site est un nœud (ou peer) qui maintient un réplica

Objectifs de la réplication

Comment gérer la mise à jour des réplicas ?

- Le mécanismes de contrôle de réplication sont classés selon :
  - Où les mises à jour sont générés ?
    - Réplication Single Master
    - Réplication Multi-Master
  - Quand les mises à jour sont propagées aux autres réplicas ?
    - Propagation synchrone (eager)
    - Propagation asynchrone (lazy)

Objectifs de la réplication

Comment assurer la cohérence des réplicas ?

- Mettre à jour un objet répliqué et préserver l'égalité des états des réplicas est un **verrou scientifique**
- Deux réplicas sont **mutuellement consistants** s'ils contiennent le même état à un instant donné
- Deux réplicas sont **divergents** s'ils contiennent des états différents (exécution parallèle des mises à jour)

Types de réplica

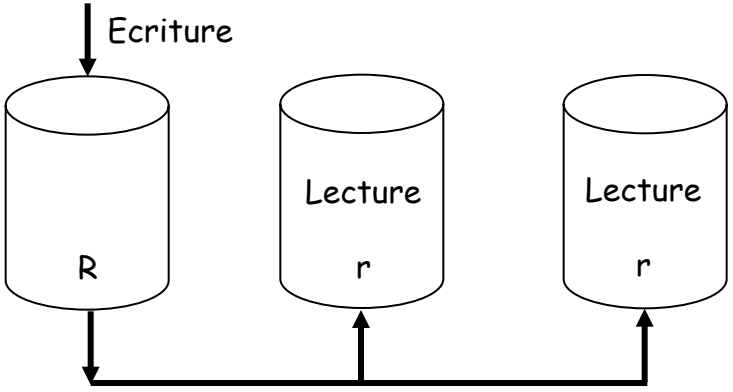
Il existe deux types de réplicas

- Réplica primaire (ou copie primaire)
  - Exécution d'opérations de Lecture et Ecriture
  - Hébergé par un site Master (ou Maître)

- Réplica secondaire (ou copie secondaire)
  - Limité à des opérations de Lecture
  - Hébergé par un site Slave (ou Esclave)

Single-Master

- Caractéristiques
  - Un seul réplica primaire
  - Plusieurs réplicas secondaires
  - La mise à jour est générée au niveau du réplica primaire puis elle est propagée vers les autres réplicas secondaires

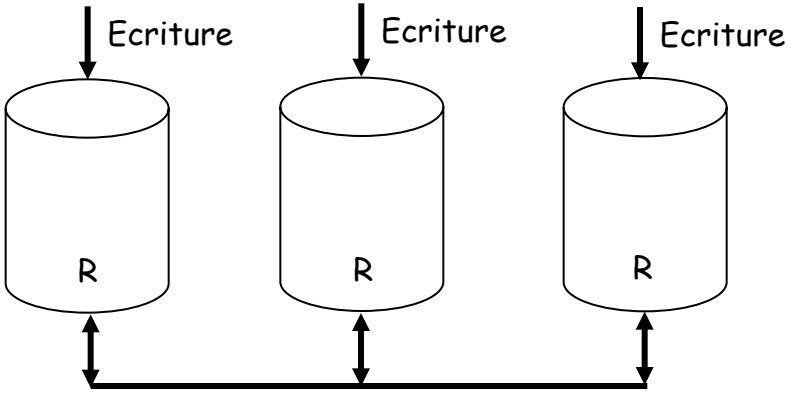


Single-Master

- Avantages
  - Un seul Ecrivain et Plusieurs Lecteurs
  - Pas de mises à jour concurrentes
  - La Cohérence est assurée comme le contrôle de concurrence est simple
- Incovénients
  - Une conception centralisée introduisant un goulot d'étranglement
  - Le site Maître est potentiellement un point de panne
  - La disponibilité des données peut être affectée

Multi-Master

- Caractéristiques
  - Plusieurs sites contiennent des réplicas primaires
  - Les mises à jour sont concurrentes
  - L'ordre d'exécution des mises à jour peut être différent d'un site à un autre



Multi-Master

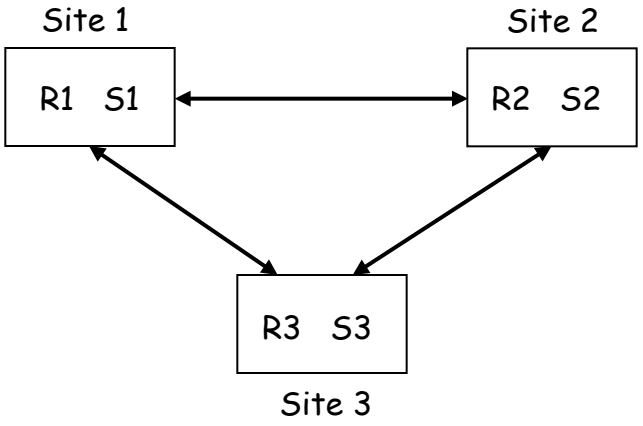
- Avantages
  - Pas de point de panne
  - Données de plus en plus disponibles
- Incovénients
  - La cohérence des données nécessite une coordination sûre
  - Utilisation d'algorithmes complexes de réconciliation
  - La communication entre les sites est coûteuse

Single-Master vs. Multi-Master

Elément de Comparaison	Single-Master	Multi-Master
Caractéristique	Une copie primaire	Plusieurs copies primaires
Contrôle de concurrence	Non appliqué	Coordination Réconciliation
Valeurs ajoutées	Copie primaire	Partout
Technique de mise à jour	Centralisée	Distribuée
Blocage	Site Master	Tous les sites Master
Panne	Oui	Non

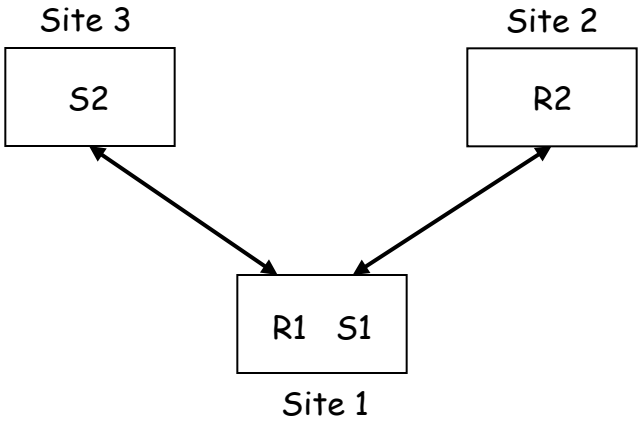
Réplication Complète (RC)

- Caractéristiques
  - Les objets partagés sont répliquées sur tous les sites
  - Tous les sites ont les mêmes capacités
  - Un site peut remplacer un autre site en cas de panne



Réplication Partielle (RP)

- Caractéristiques
  - Chaque site contient une copie d'un sous-ensemble d'objets
  - Les coûts du stockage et de communication sont réduits
  - Le protocole de propagation devient complexe

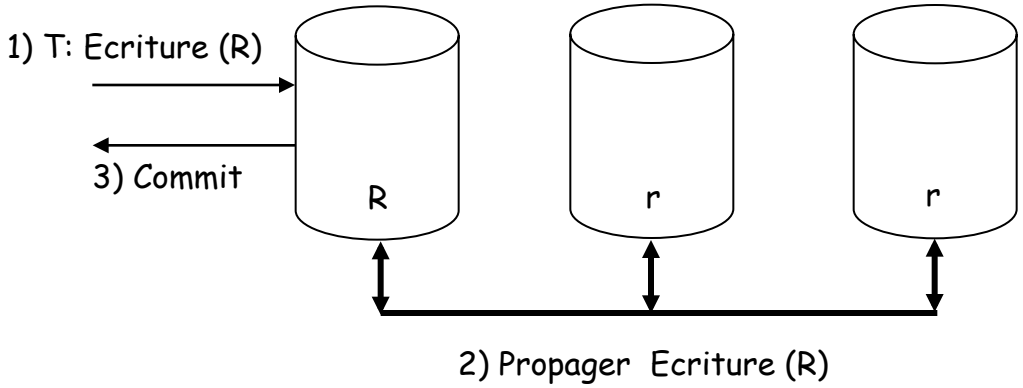


RC vs. RP

Elément de comparaison	Réplication Complète	Réplication Partielle
Caractéristique	Tous les sites contiennent toutes les copies des objets partagés	Chaque site contient une copie d'un sous-ensemble d'objets d'objets partagés
Chargement	Simple	Compliqué
Disponibilité	Maximale	Moins
Espace de stockage	Peut être coûteux	Réduit
Coûts de communication	Peut être coûteux	Réduit

Propagation Synchrone (eager)

- Caractéristiques
  - Propagation est une opération incluse dans la transaction
  - Après Commit tous les réplicas ont le même état



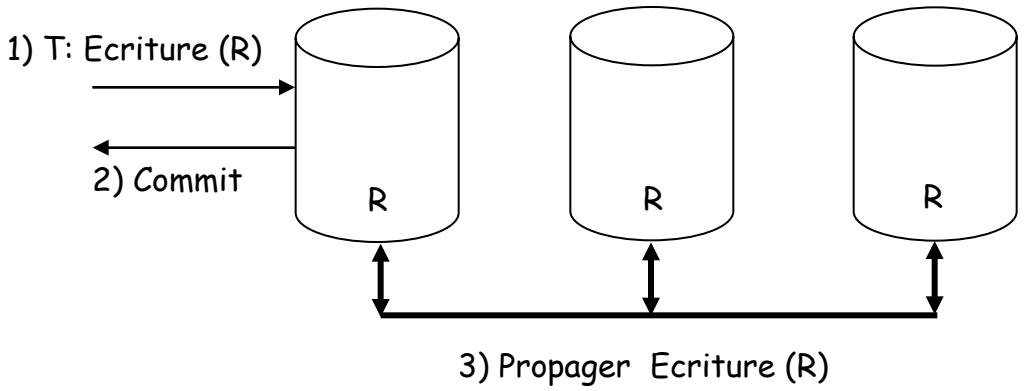
Propagation Synchrone (eager)

- Avantages
  - Eviter les divergences entre les réplicas
  - Les lectures sont donc correctes

- Inconvénients
  - Commit est conditionnée par la m-à-j de tous les réplicas
    - S'il y a une panne le processus est bloqué
  - Ce type propagation est inapproprié pour des réseaux dynamiques
  - Le passage à l'échelle est donc côuteux

Propagation Asynchrone (lazy)

- Caractéristiques
  - Propagation se fait après Commit
  - Deux classes sont considérées :
    - Optimiste (les conflits sont rares)
    - Non-optimiste (les conflits sont toujours possibles)



Propagation Asynchrone (lazy)

- Avantages
  - Tolérance aux pannes
    - les mises à jour ne sont pas bloquantes
  - Amélioration de la disponibilité des données
- Inconvénients
  - Les réplicas peuvent diverger (cas optimiste)
    - Ordre d'exécution diffère d'un site à un autre
  - Des algorithmes de réconciliation sont nécessaires

Sommaire

- Présentation du Peer-to-Peer (P2P)
- Techniques de Réplication
- *Réplication Optimiste*
- Transformées Opérationnelles

Réplication Optimiste

- Principe
  - Les réplicas peuvent diverger
  - Quand le système est au repos, les réplicas doivent être identiques
- Paramètres
  - Opération
  - Relations entre les opérations
  - Fréquence de propagation
  - Détection et résolution des conflits
  - Réconciliation

Opération

Prescription contenant les informations nécessaires pour la mise à jour d'un objet

- Opération persistante
  - Stockée dans un **fichier log** pour être ensuite propagée vers d'autres sites
  - Utilisée par des systèmes basés sur le **transfert d'opérations**
    - Exemple : Bayou, IceCube
- Opération transitoire
  - Eliminée après son exécution
  - Utilisée par des systèmes basés sur le **transfert d'état**
    - Exemple : DNS, Harmony

Relations entre opérations

- Associations implicites et explicites entre opérations
- Indication sur l'existence de conflits entre plusieurs mises à jour
- Résolution des conflits entre plusieurs mises à jour
  - Exemple : réarranger les opérations selon un ordre approprié
- Quatre types de relations :
  - Happens-before
  - Concurrence
  - Contrainte explicite
  - Contrainte implicite

Happens-before

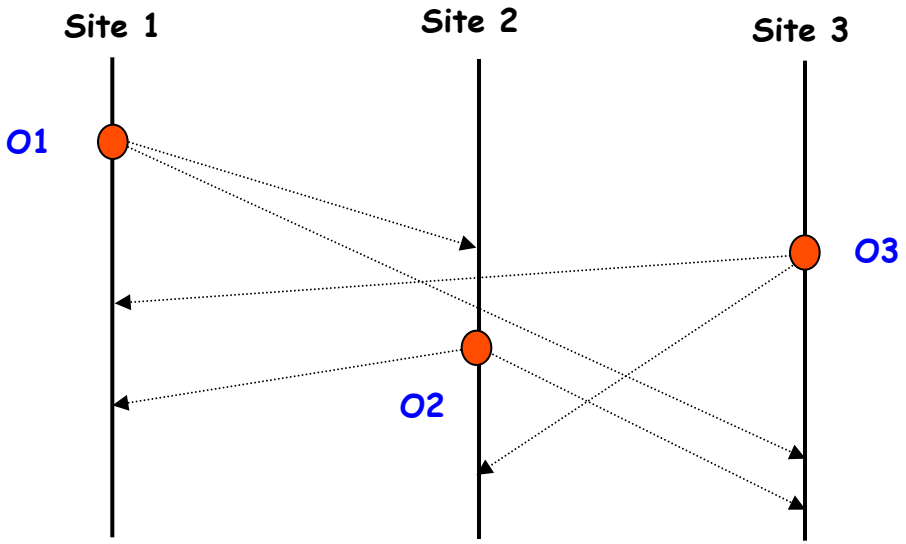
Elle capture un **ordre partiel** entre les opérations

Deux opérations o_1 et o_2 exécutées respectivement sur les sites i et j. Opération o_1 **happens-before** o_2 ssi l'une des trois conditions est satisfaite :

- i=j et o_1 était exécutée avant o_2
- i≠j et o_2 est exécutée après réception et exécution de o_1
- Il existe une opération o telle que o_1 happens-before o et o happens-before o_2

Concurrence

Deux opérations o_1 et o_2 sont dites **concurrentes** ssi o_1 n'a pas vu l'effet de o_2 l'autre et vice-versa



Contrainte explicite

Un **Invariant** introduit **dynamiquement** dans le système pour représenter la sémantique d'une application

Quelques exemples :

- De préférence, exécuter o_1 avant o_2 (crédit avant débit)
- si o_1 s'exécute alors o_2 aussi
- Exclusion mutuelle (soit o_1 ou o_2 mais pas les deux)

Contrainte implicite

Un **Invariant** introduit **statiquement** dans le système pour représenter la sémantique d'une application

Quelques exemples :

- Ordre causal (o_1 précède o_2)
- Tout ou rien (o1 et o2 ou aucune)

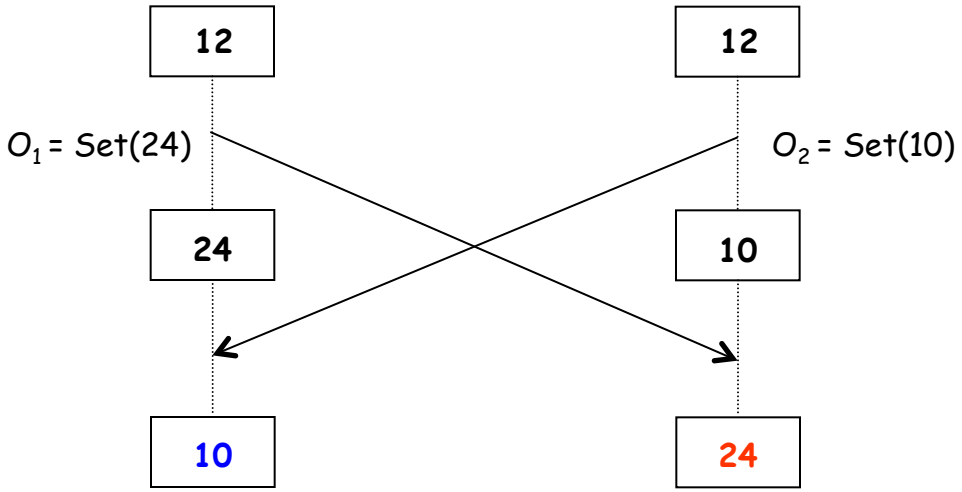
Fréquence de Propagation

Echange d'opérations ou de réplicas (états de l'objet de partage) entre les sites pour assurer la consistance

- Système Pulling
  - Chaque site obtient de nouvelles opérations (ou réplicas) en faisant la demande à d'autres sites
- Système Pushing
  - Chaque site envoie de nouvelles mises à jours de manière dynamique aux autres sites
- Système Hybride
  - Combiner les comportements Pulling et Pushing

Détection et Résolution des Conflits

- Plusieurs sites pourraient modifier les réplicas en même temps. Une telle situation génère des **conflits**
- Un **conflit** entre deux opérations o1 et o2 signifie que l'exécution des séquences $[o_1;o_2]$ et $[o_2;o_1]$ mène à deux réplicas différents

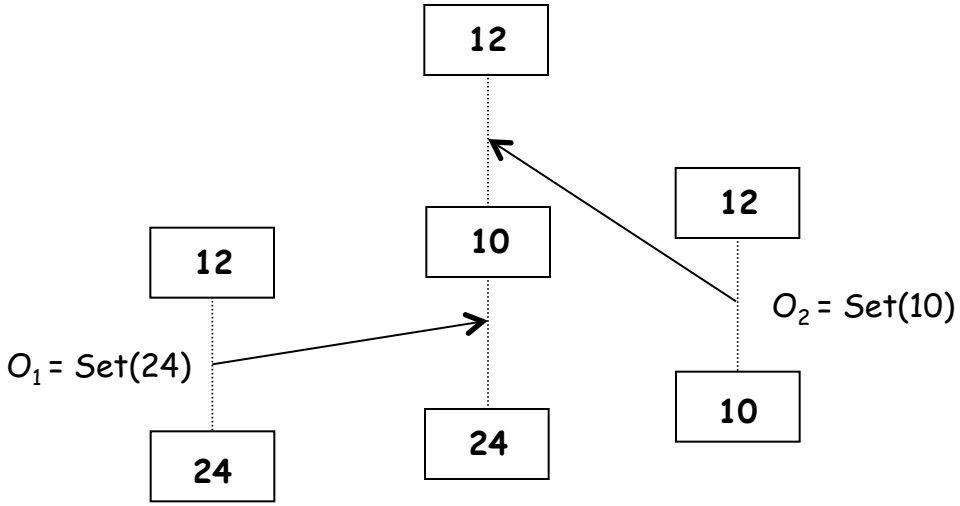


Détection et Résolution des Conflits

- **Détection de conflits** consiste à reconnaître les modifications conflictuelles
- Trois politiques sont utilisées :
  - Ignorer le conflit (dernier arrivé a toujours raison !)
  - Se baser la relation de concurrence
  - Se baser sur la sémantique de l'objet partagé

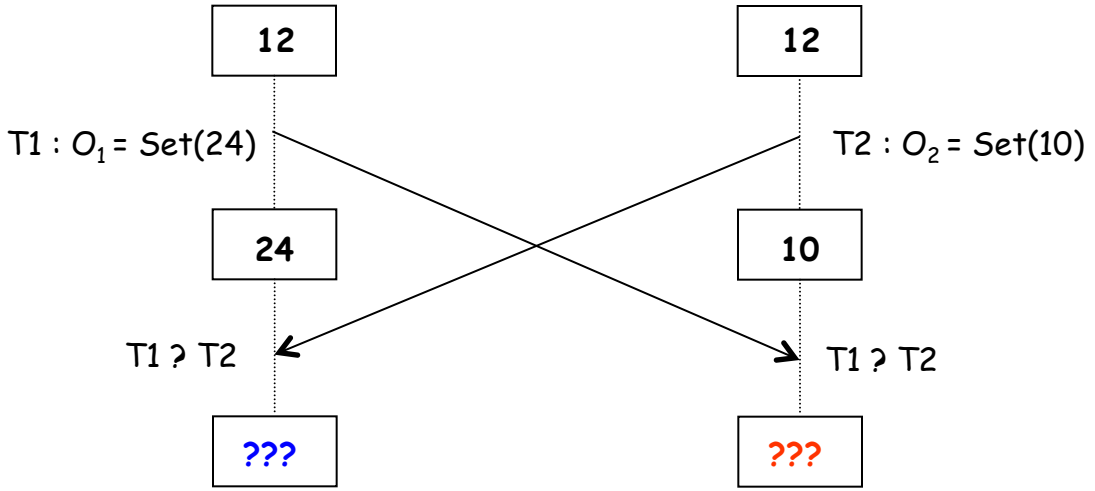
Détection et Résolution des Conflits

- Politique ignorant le conflit
  - Perte de modifications (e.g. DNS, Wikipédia)



Détection et Résolution des Conflits

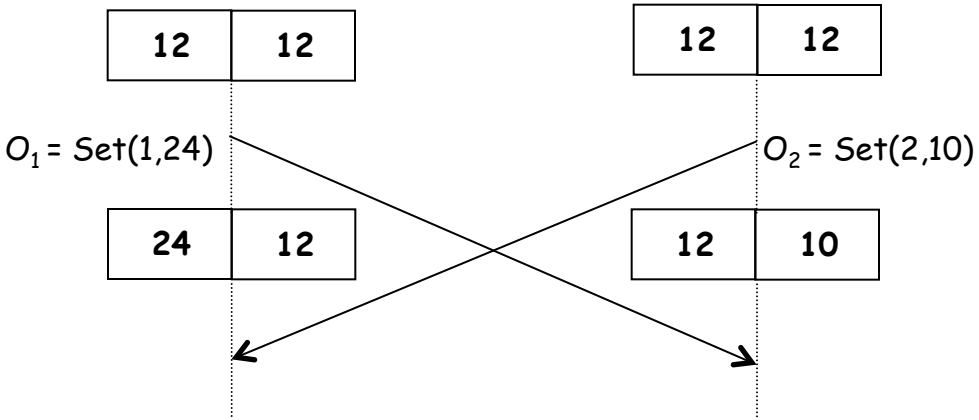
- Politique basée sur la concurrence
  - Traquer la relation de concurrence pour détecter les conflits



Comparer le timing des opérations ou des réplicas

Détection et Résolution des Conflits

- Politique basée sur la sémantique
  - Une connaissance de la sémantique peut aider à détecter les conflits



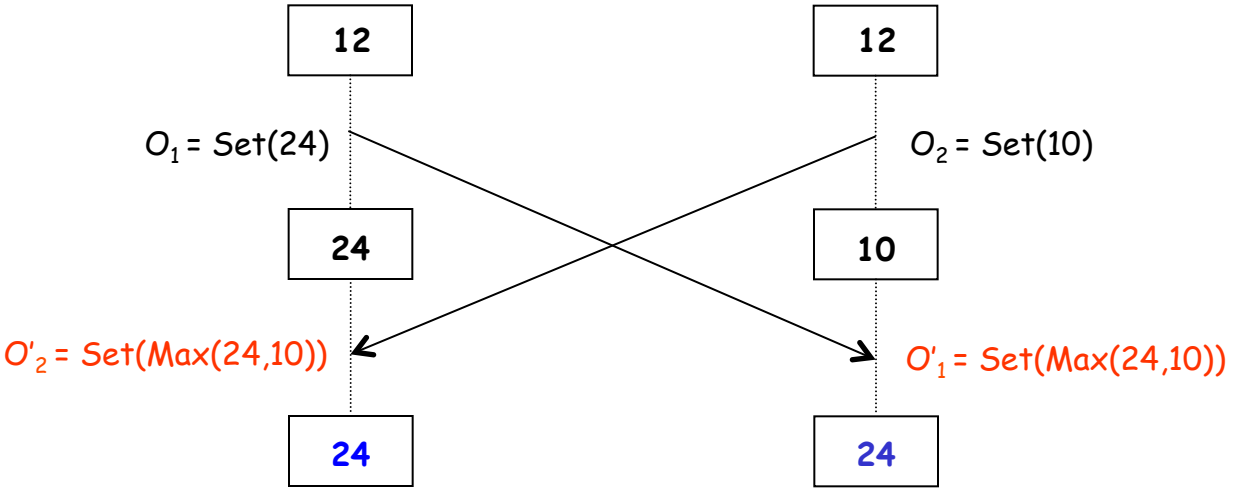
Les opérations sont appliquées sur deux cases différentes
Donc pas de conflit

Détection et Résolution des Conflits

- **Résolution de conflits** consiste à proposer des mécanismes pour lever les situations conflictuelles
- Deux types de résolution sont utilisées :
  - Résolution manuelle
    - La situation de conflit est soumise aux sites pour que ces derniers essayent de la résoudre (mode interactif)
  - Résolution automatique
    - Utilisation d'une procédure de fusion pour lever le conflit (mode automatique)

Détection et Résolution des Conflits

- Exemple de résolution automatique



Réconciliation

- Réconciliation est une activité qui **transforme** des **réplicas divergents** vers un **état consistant**

  Réconciliation = < Entrée , Ordre >

- Entrée
  - Réplicas : Réconcilation basée sur l'état
  - Opérations : Réconciliation basée sur les opérations
- Ordre
  - Informations ordinales (approche générique)
    - Préserver les modifications selon un ordre déterminé
  - Informations sémantiques (dépend de l'objet de partage)
    - Exploiter la sémantique de l'objet partagé pour réduire les conflits et converger vers un même

Exemple : IceCube

Système de réconciliation qui exploite la sémantique de l'objet partagé pour résoudre les conflits

Ses caractéristiques :

- Composé de n sites (n est fixe)
- L'objet partagé est répliqué dans chaque site
- Opérations = Actions
- Utilisation d'un log pour stocker les actions
- La sémantique est capturée à l'aide des constraintes entre les actions
- La réconciliation est un problème d'optimisation
  - Trouver le plus grand ensemble d'action qui ne violent pas les contraintes

Exemple : IceCube

Soit T = (K, A, B) une table relationnelle répliquée dans trois sites S1, S2 et S3

S1 a_1^1: update T1 set A=a1 where K=k1

S2 a_1^2: update T2 set A=a2 where K=k1

S3 a_1^3: update T3 set B=b1 where K=k1

a_2^3: update T3 set A=a3 where K=k2

Les actions sont : a_1^1 , a_1^2 , a_1^3 , a_2^3

Exemple : IceCube

Une contrainte est une représentation formelle d'un invariant d'une application

Une contrainte implicite est définie par l'utilisateur. C'est une contrainte statique.

- Parcel(a_1^3 , a_2^3)
  - Propriété d'atomicité (tout ou rien)
  - Soit a_1^3 et a_2^3 s'exécutent avec succès soit aucune ne s'exécute

Exemple : IceCube

Une contrainte explicite est déterminée par le système en se basant sur la sémantique de l'objet partagé. C'est une contrainte dynamique.

- MutuallyExclusive(a_1^1 , a_1^2)
  - Propriété d'alternation
  - Soit a_1^1 ou a_1^2 peut s'exécuter mais pas les deux
  - Les deux opérations constituent un conflit

Exemple : IceCube

La réconciliation permet d'obtenir le plus large ensemble d'actions à exécuter qui respectent toutes les contraintes

- Les actions à ordonner sont divisés en **clusters**
  - Un cluster est un sous-ensemble d'actions qui sont reliées par des contraintes
  - Les actions d'un cluster sont ordonnées indépendamment des autres clusters
- Le résultat s'obtient par concaténation de toutes les actions ordonnées des clusters

Exemple : IceCube

- Input
  - Toutes les actions
  - Les contraintes implicites et explicites
- Output
  - Un log (appelé **schedule**)
    - Une suite ordonné d'actions à exécuter

Pour ordonner un cluster :

- Choisir une action selon son degré de mérite
- Supprimer l'action choisie du cluster
- Supprimer du cluster les actions qui sont en conflit avec l'action choisie

Exemple : IceCube

S1 a_1^1: update T1 set A=a1 where K=k1

S2 a_1^2: update T2 set A=a2 where K=k1

S3 a_1^3: update T3 set B=b1 where K=k1

a_2^3: update T3 set A=a3 where K=k2

Deux clusters sont définis:

$C1 = \{a_1^1 , a_1^2\}$ et $C2 = \{a_1^3 , a_2^3\}$

C1 est défini par rapport à MutuallyExclusive(a_1^1 , a_1^2)

C2 est défini par rapport à Parcel(a_1^3 , a_2^3)

Exemple : IceCube

S1 a_1^1: update T1 set A=a1 where K=k1

S2 a_1^2: update T2 set A=a2 where K=k1

S3 a_1^3: update T3 set B=b1 where K=k1

a_2^3: update T3 set A=a3 where K=k2

Le résultat est constitué de deux logs possibles :

$L1 = [a_1^1 , a_1^3 , a_2^3]$

ou

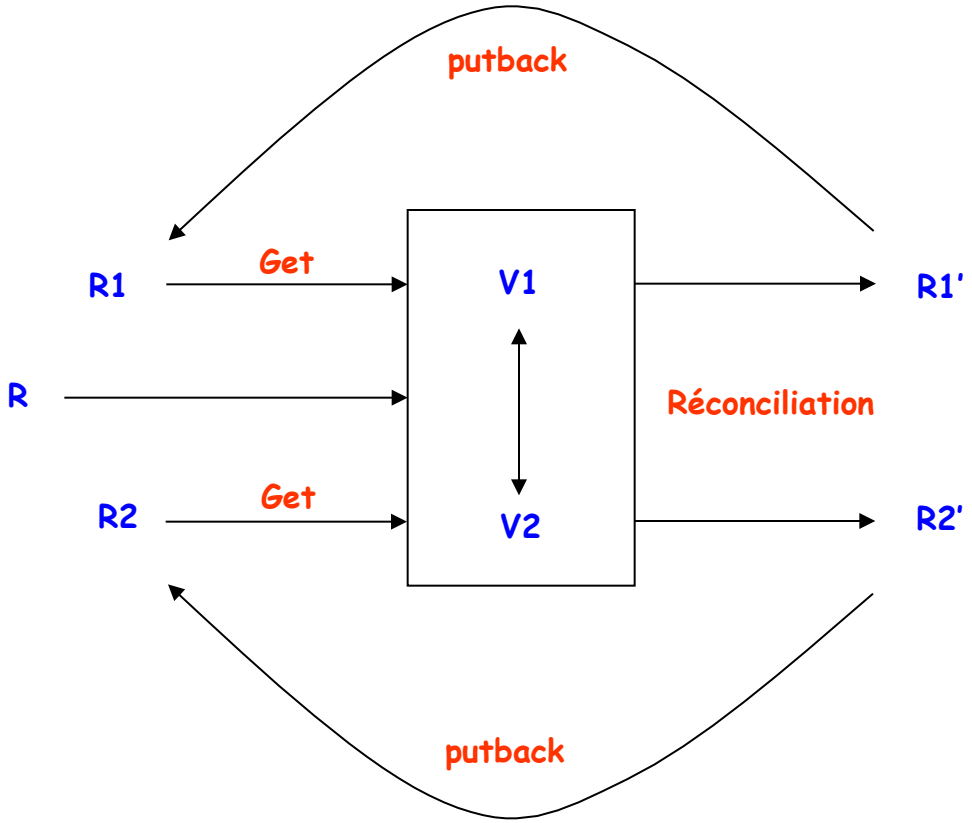
$L2 = [a_1^2 , a_1^3 , a_2^3]$

Exemple : Harmony

Harmony = environnement de réconciliation pour des objets hétérogènes et dont la structure est arborescente

- Il est utilisé, par exemple, pour réconcilier des marque-pages (bookmarks) issus de plusieurs formats d'explorateur internet (web browsers)
  - Mozilla
  - Safari
  - Internet Explorer
  - Camino

Exemple : Harmony



Architecture de Harmony

Exemple : Harmony

Harmony utilise des contraintes implicites sur un arbre modifié par les opérations suivantes:

Create and Delete

Contraintes implicites	**Conflits**
Un nœud d'un arbre ne peut pas être supprimé dans un réplica et modifié dans un autre (ajout d'un nouveau fils)	Delete/Create
A sous-arbre ne peut pas être entièrement supprimé dans un réplica et partiellement supprimé dans un autre	Delete/Delete
Différents sous-arbres ne peuvent être crées à la même place	Create/Create

Exemple : Harmony

L'algorithme de réconciliation procède comme suit :

- Soient 2 paires de nœuds nR1 et nR2 issus de deux réplicas R1 et R2
- Si nR1 et nR2 sont égaux alors ils sont déjà réconciliés
- Si nR1 et nR2 sont différents et ils sont conflictuels alors nR1 et nR2 restent inchangés
- Si nR1 et nR2 sont différents et ils ne sont pas conflictuels alors des modifications sont appliquées à l'un et/ou l'autre pour les faire converger au même noeud

IceCube vs. Harmony

	IceCube	**Harmony**
Objet	Applications spécifiques (documents XML, table, etc.)	Arbre (document XML, système de fichiers, bookmarks, etc.)
Stockage d'opérations	Opérations persistantes (utilisation du log)	Opération transitoires
Relations entre opérations	Contraintes implicites et explicites	Contraintes implicites
Propagation	Pull (transfert périodique)	Transfert d'état à la demande
Conflit, Détection et Résolution	Détection basée sur la sémantique Résolution automatique (optimisation)	Basé sur la sémantique de l'arbre Résolution manuelle (pas de résolution pour les nœuds en conflit
Réconciliation	Considère les actions et les contraintes de plusieurs logs pour produire un log global	Considère les états des réplicas Réconcilier les nœuds non conflictuels
Consistance	Consistance éventuelle	Pas de garanties

Sommaire

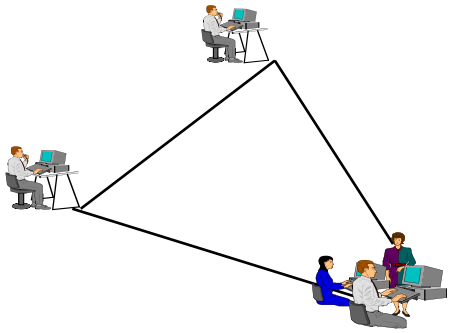
- Présentation du Peer-to-Peer (P2P)
- Techniques de Réplication
- Réplication Optimiste
- Transformées Opérationnelles

Transformées Opérationnelles (OT)

Conçue à la base pour **l'Edition Collaborative**

Caractéristiques

- Groupe de personnes éloignés géographiquement
- Edition concurrente d'un même document (texte, XML, image)
- Document répliqué sur les sites
- Document partagé = Objet Collaboratif
- Exemple : CVS, CoWord, Wiki ...



Transformées Opérationnelles (OT)

Le défi de **l'Edition Collaborative** = Assurer la consistance (ou la convergence) des réplicas malgré

- Degré important de concurrence
  - Modifier localement et propager la modification aux utilisateurs
- Latence de la communication
  - Communiquer via Internet
- Groupe dynamique
  - Collaborer dans un réseau P2P
  - Joindre et quitter le groupe à tout moment

Transformées Opérationnelles (OT)

Exemple d'Edition Collaborative

Objet Collaboratif = Texte

- Opérations
  - Ins(p,c)
    Insérer le caractère c à la position p
  - Del(p,c)
    Supprimer le caractère c à la position p

QuickTime™ et un décompresseur TIFF (non compressé) sont requis pour visionner cette image.

Situation de Divergence

Transformées Opérationnelles (OT)

Eviter la divergence :
Techniques pessimistes

- **Principe**
  - Accès exclusif aux réplicas
    Utilisation des verrous
- **Inconvénients**
  - Problèmes d'attente et d'interblocage
  - Contrôle central

QuickTime™ et un
décompresseur TIFF (non compressé)
sont requis pour visionner cette image.

Transformées Opérationnelles (OT)

Eviter la divergence :
Techniques optimistes
Par Sérialisation

- **Principe**
  - Accès concurrent aux réplicas
  - Tolérance à la divergence
  - Même ordre d'exécution (Convergence)
- **Inconvénients**
  - Annulation et réexécution
  - Perte de modifications

QuickTime™ et un décompresseur TIFF (non compressé) sont requis pour visionner cette image.

Transformées Opérationnelles (OT)

Eviter la divergence :
Techniques optimistes
Par Transformation

QuickTime™ et un
décompresseur TIFF (non compressé)
sont requis pour visionner cette image.

- **Principe**
  - Ordres d'exécution différents
  - Transformation des opérations (Convergence)
  - Prise en compte de toutes les opérations

Transformées Opérationnelles (OT)

Exemple d'algorithme de Transformation proposé par Ellis and Gibb [Sigmod'89]

IT(Ins(p_1,c_1,pr_1), Ins(p_2,c_2,pr_2)) =
if *(p_1 < p_2)* **then return** *Ins(p_1,c_1,pr_1)*
else if *(p_1 > p_2)* **then return** *Ins(p_1+1, c_1,pr_1)*
else if *(c_1 == c_2)* **then return** *Nop()*
else if *(pr_1 > pr_2)* **then return** *Ins(p_1+1,c_1,pr_1)*
else return *Ins(p_1,c_1,pr_1)*
endif *;*

IT(Del(p_1, pr_1), Ins(p_2, c_2, pr_2)) =
if *(p_1 < p_2)* **then return** *Del(p_1, pr_1)*
else return *Del(p_1+1, pr_1)*
endif *;*

Transformées Opérationnelles (OT)

Exemple d'algorithme de Transformation proposé par Ellis and Gibb [Sigmod'89]

IT(Ins(p_1, c_1, pr_1), Del(p_2,pr_2)) =
if *($p_1 < p_2$)* **then return** *Ins(p_1, c_1, pr_1)*
else return *Ins(p_1-1, c_1, pr_1)*
endif *;*

IT(Del(p_1,pr_1),Del(p_2,pr_2)) =
if *($p_1 < p_2$)* **then return** *Del(p_1,pr_1)*
else if *($p_1 > p_2$)* **then return** *Del(p_1-1,pr_1)*
else return *Nop()*
endif *;*

Transformées Opérationnelles (OT)

Composants de OT

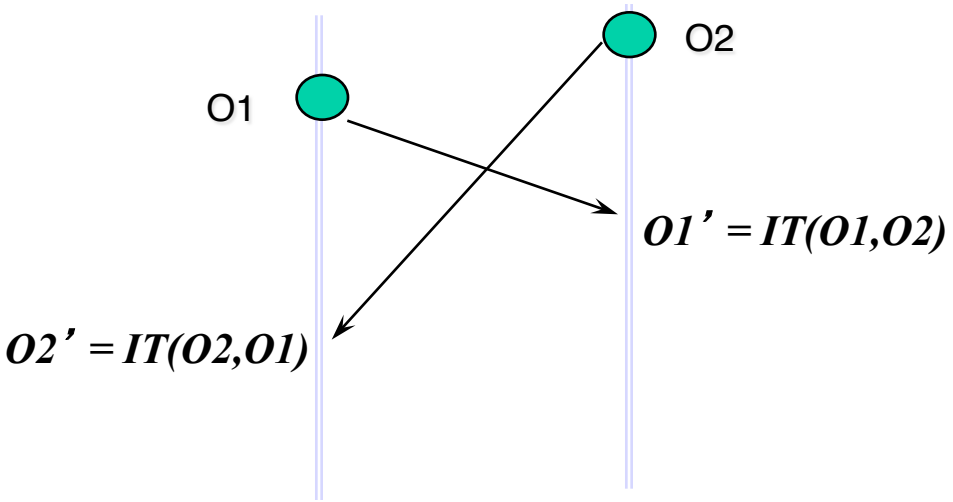
- Séquence des opérations
  - Chaque site stocke les opérations dans une structure donnée appelée log (ou histoire)
- Algorithme d'intégration
  - Réception, Propagation, Exécution des opérations
  - Causalité (vecteurs d'état)
  - Exemples : dOPT, AdOPTed, SOCT2 et SOCT4
- Algorithme de transformation
  - Sérialisation des opérations concurrentes dans des ordres différents
  - Spécifique à la sémantique de l'objet collaboratif (partagé)

Transformées Opérationnelles (OT)

Condition de convergence TP1

Pour toutes les opérations concurrentes définies sur le même état o1 et o2 :

$$[o1 ; IT(o2,o1)] \equiv [o2 ; IT(o1,o2)]$$



Identité d'états

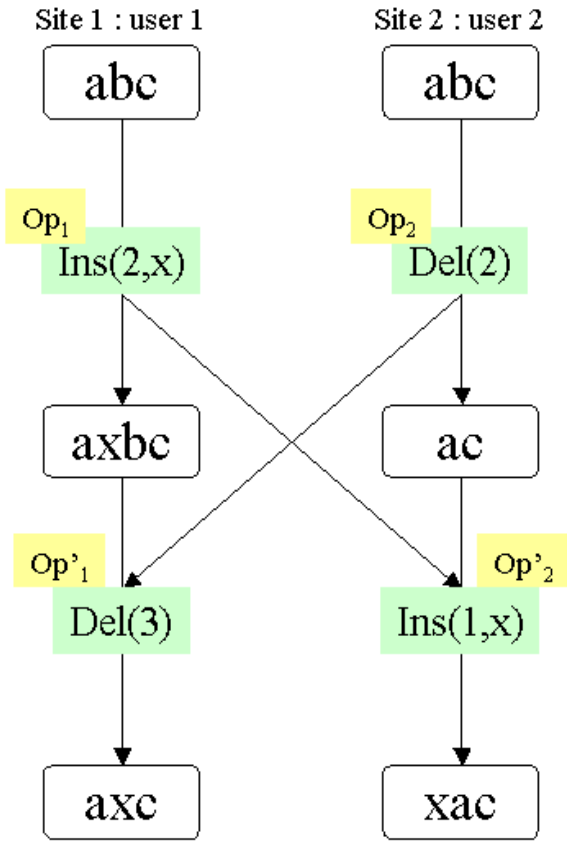
Transformées Opérationnelles (OT)

Exemple de violation de TP1

Algorithme de Transformation de Ellis and Gibb

IT(Ins(p_1, c_1, pr_1), Del(p_2,pr_2)) =
if *($p_1 < p_2$)* **then return** *Ins(p_1, c_1, pr_1)*
else return *Ins(p_1-1, c_1, pr_1)*
endif

IT(Del(p_1, pr_1), Ins(p_2, c_2, pr_2)) =
if *($p_1 < p_2$)* **then return** *Del(p_1, pr_1)*
else return *Del(p_1+1, pr_1)*
endif



Où est l'erreur ?

Transformées Opérationnelles (OT)

IT(Ins(p_1, c_1, pr_1), Del(p_2,pr_2)) =
if *($p_1 < p_2$)* **then return** *Ins(p_1, c_1, pr_1)*
else retu *Ins(p_1-1, c_1, pr_1)*
endi

IT(Del
if *($p_1 < p_2$)* **then return** *Del(p_1, pr_1)*
else return *Del(p_1+1, pr_1)*
endif ;

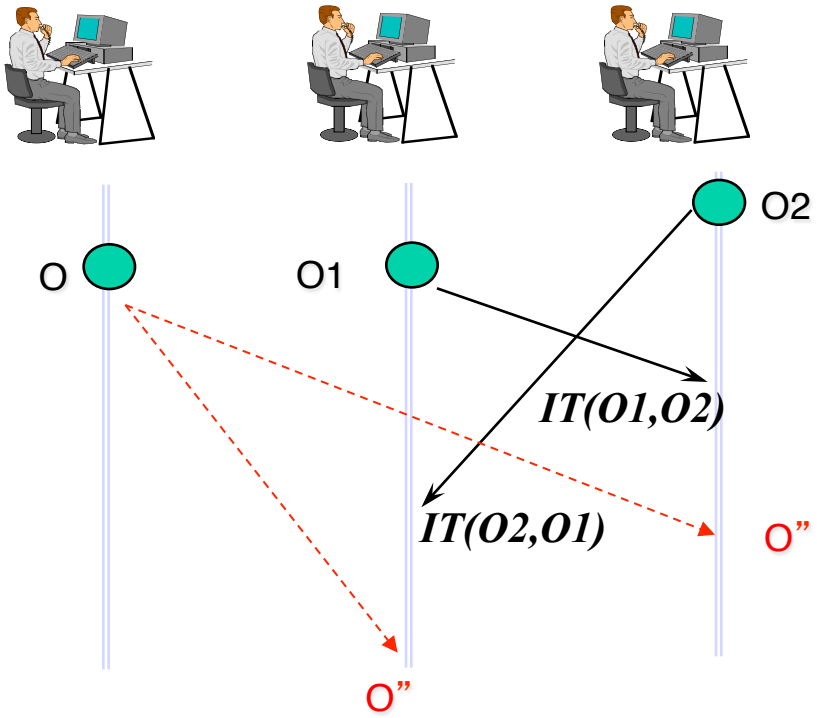
La source d'erreur. Elle devrait être réécrite $p_1 \leq p_2$

Transformées Opérationnelles (OT)

Condition de convergence TP2

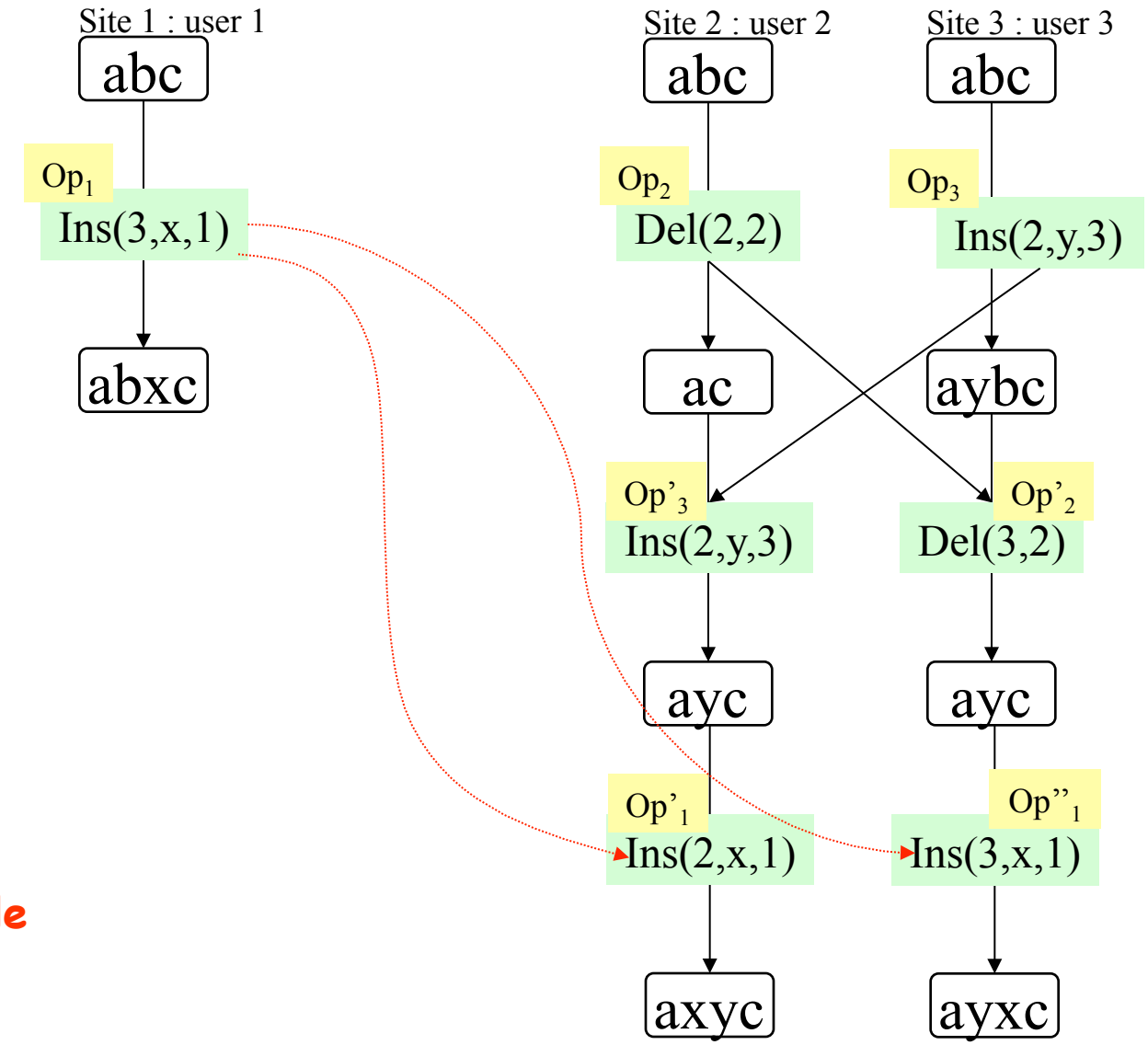
Soient 3 opérations concurrentes définies sur le même état o, o1 et o2 :

IT(IT(o,o1), IT(o2,o1)) = IT(IT(o,o2) , IT(o1,o2))



Identité d'opérations

Transformées Opérationnelles (OT)



TP2 Puzzle

Transformées Opérationnelles (OT)

Critère de Consistance

Un système répliqué basé sur OT est consistant ssi :

- Préservation de causalité
  - Si o1 happens-before o2 alors o1 doit être exécutée avant o2 sur tous les sites
- Convergence
  - Quand les sites ont exécutés le même ensemble d'opérations, les réplicas de l'objet partagé doivent être identiques

Transformées Opérationnelles (OT)

Algorithme d'intégration

Quand un site i reçoit une opération o, l'algorithme d'intégration procède comme suit :

1. Déterminer à partir de l'histoire locale L une histoire équivalente $L = L_h ; L_c$ tels que :
   - L_h contient les opérations qui précèdent causalement o
   - L_c contient les opérations concurrentes à o
2. Exécuter l'algorithme de transformation IT pour déterminer l'opération o' tel que :
   - l'opération o' est la transformation de o par rapport à L_c
3. Exécuter l'opération o' sur l'état courant
4. Ajouter o' à l'histoire local L

Transformées Opérationnelles (OT)

Exercice : Tampon de mémoire

Soit un tampon mémoire TM partagé et dont l'état est altéré par l'opération suivante :

SetNat(n) qui assigne l'entier n à TM

Soit l'algorithme de transformation IT :

IT (SetNat(n1), SetNat(n2)) = SetNat(n1)

1. IT ne satisfait pas TP1. Pourquoi ?
2. Corriger IT pour vérifier TP1.
3. Vérifier si la version corrigée satisfait TP2

Références Bibliographiques

1. V. Martins, E. Pacitti and P. Valduriez
« Survey of Data Replication in P2P System » Rapport de recherche INRIA N°6083, December, 2006

2. C.A. Ellis and S. Gibb « Concurrency Control in Groupware Systems » SigMod 1989

3. C. Sun and C.A. Ellis « Operational Transformation in Real-Time Group Editors » CSCW 1998

4. A. Imine, P. Molli, G. Oster and M. Rusinowitch « Proving Correctness of Transformation Functions in Real-Time Groupware» ECSCW 2003

5. H. Bouchneb and A. Imine « Experiments in Model-Checking Optimistic Replication Algorithms », Rapport de Recherche INRIA N° 6510, Avril 2008

Travail à faire

Proposer un algorithme de transformation IT pour les objets suivants :

- Une pile dont les opérations sont :
  - Empiler un élément e : Push(e)
  - Dépiler : Pop
- Une file dont les opérations sont :
  - Insérer un élément e : Enqueue(e)
  - Supprimer un élément e : Dequeue(e)
- Un ensemble dont les opérations sont :
  - Insérer un élément e : Ins(e)
  - Supprimer un élément e : Del(e)
- Un arbre dont les opérations sont :
  - Insérer un nœud
  - Détruire un nœud

1) Mettre en évidence toutes les situations possibles de conflit
2) Donner la preuve que IT vérifie TP1 et TP2