# Lecture 1 : Distributed Systems

**Definition** : A set of Nodes/process working together and communicating through a network. It’s a collection of independent computers that appears to its users as a single coherent system.

**Type of distributed systems :**

* Distributed Computing Systems : used for high performance computing tasks
* Distributed Information Systems : systems mainly for management and integration of / enterprise Application Integration
* Distributed Pervasive Systems : mobile and embedded systems / home systems

**Advantages :**

* performance / speedup
* grow incrementally as computers are added
* sharing of resources
* heterogeneity and distributed computing problems
* Each piece of hardware is replaceable should it fail

**Disadvantages :**

* insecure communication
* loss of control over parts of the system
* parts of the system are down or inaccessible

**Communication** in distributed system is based on message passing (protocols are use to communicate : UDP/TCP).

**Group communication** offers a service whereby a message is sent to a group and then this message is delivered to all members of the group.

**Why do we synchronize need clocks?** To determine when one thing happened before another (events can be ordered using (logical) clocks).

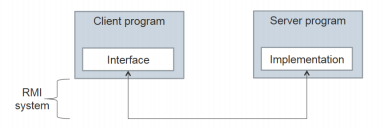
# 

# Lecture 2 : Remote Method Invocation

# 

# Definition : Instantiate an object on another machine

Invoke methods on the remote object



RMI is for Java only / is object-oriented

# 

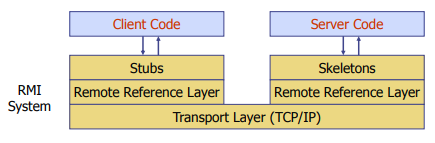
# Why not just use sockets? More error prone and tedious (fastidieux).

**Advantages :**

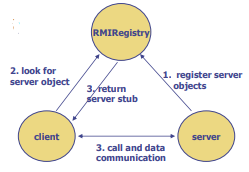
* provide user with thin client
* good performance on low end clients
* maximize $ investment over many clients

**Primary goal** of RMI is to allow programmers to develop distributed Java programs with the same syntax and semantic used for non-distributed programs.

**What are RMI layers ?** Stub and skeleton, remote reference layer, transport layer



**What are RMI components ?** Registry that list objects, server that implements and give access to object and client.



# Lecture 3 : Fault Tolerance

## Fault Tolerance

# Definition : The characteristic by which a system can mask the occurrence and recovery from failures. A system is fault tolerant if it can continue to operate even in the presence of failures.

**Goal :** mask partial failures

**Partial failure :** One component fails. Some components are affected. Others not.

**Properties should a dependable system satisfy :**

* Availability: the system is ready to be used immediately
* Reliability: the system can run continuously without failure
* Safety: if a system fails, nothing catastrophic will happen
* Maintainability: when a system fails, it can be repaired easily and quickly (and, sometimes, without its users noticing the failure)
* Security: protection of a system from various threats

## Failure

# Definition : A system is said to fail when it cannot meet its properties.

**Fails are caused by errors :** damaged or lost packet, etc.

**Types of faults :**

* Transient fault: occurs once and then disappears
* Intermittent fault: may reappear again and again
* Permanent fault: continues to exist until the faulty

**Why do they fail ?** Faults at design and faults at runtime

**SHEITAN LE RESTE DU CHAPITRE**

# 

# Lecture 4 : Consistency (cohérence)

## Réplication

**Définition** : la réplication est l’action de copier ou reproduire des données ou un process.

La réplication améliore :

* performance
* availability

**Contraintes** : il est difficile de garder les données synchronisée et cohérentes

**Techniques de réplication** :

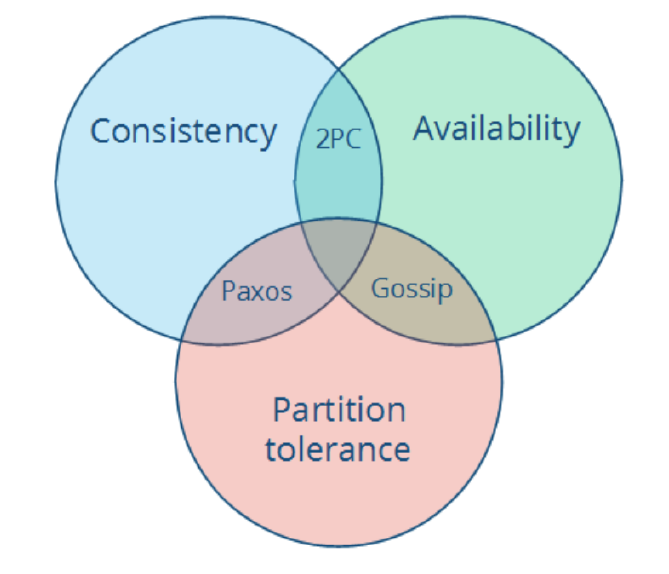
* caching (mise en cache)
* Replicated databases ( copie de données depuis une bdd)

**Théorème CAP** (Consistency Availablity Partion tolerance)

Chaque système distribué à du mal à fournir :

* Consistency (cohérence) : tous les clients ne voit pas les même données en même temps
* Availability (disponibilité) :garantie que toutes les requêtes reçoivent une réponse
* Partition tolerance : aucune panne moins importante qu'une coupure totale du réseau ne doit empêcher le système de répondre correctement

Il est impossible de créer un système qui respecte les 3 conditions.



Un **protocole de réplication** ou un **protocole de contrôle de réplique** est responsable du mappage des opérations logiques sur les éléments de données, il détermine :

* le niveau de cohérence
* Udpate site
* Update propagation

## model de cohérence

**Définition** : un modèle de cohérence est un ensemble d'événements dans le système

Il existe 2 familles de cohérences :

* cohérence forte : nécessite toutes les copies d’une donnees soient les mêmes après une mise à jour
* cohérence faible : ne nécessite pas que toutes les copies d’une donnees soient les mêmes après une mise à jour

## Les modèles de cohérence forte

### Cohérence séquentielle

**Définition** : les résultats de chaque exécution sont identiques. Les exécutions sont lancées dans un ordre séquentiel.

* chaque séquence d’écriture ou de lecture est acceptable
* tous les process voient la même séquence d’opération
* les opérations d’un process apparaissent dans un ordre spécifié par le process lui même

### Cohérence causal

**Définition** : les opérations sont exécuté dans un ordre qui reflète leur causalité

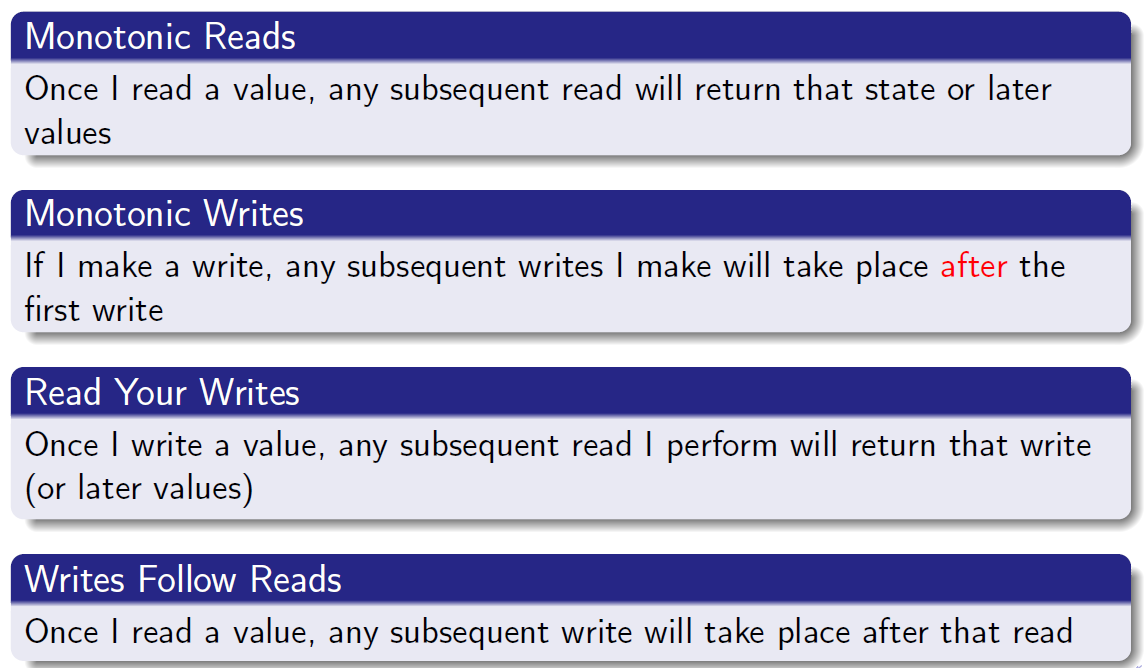
**Suivre la causalité** :

* horloge de Lamport
* la lecture dépend des écritures précédentes (même valeur)

## Les modèles de cohérence faible

### cohérence éventuelle

Garantie la cohérence des données même si aucune mise à jour se produit



# 

# Lecture 5 : Partition tolerance

# 

De nos jours beaucoup d’applications dépendent plus des données que de la puissance de calcul.

Les problèmes majeurs sont :

* la quantité de données
* la complexités des données
* la vitesse à laquelle les données changent

Dans ce type de systèmes, le travail est décomposé en plusieurs tâches qui peuvent être effectuées efficacement par un seul processus / nœud.

Un “network partition” est une erreur de communication entre 2 systèmes.

# Lecture 6 : Peer to peer

un **système centralisé** est fragile car :

* requiert une grande capacité de stockage
* requiert une grande capacité de calcul
* vulnérable aux attaques

un **système distribué** est robuste car :

* chaque noeud est indépendant
* la puissance de calcul est partagée
* le réseau est robuste

**un système peer to peer** : système distribué dans lequel les noeuds organise la topologie de façon autonome. Chaque noeud participe au stockage et au calcul. Quand le nombre de noeud augmente les capacités de stockage et de calcul augmentent.

Dans un système en peer to peer, la recherche de donnée nécessite un algorithme efficace si il y a beaucoup de noeud.

## type de peer to peer

### peer to peer non structuré

Dans un réseau peer to peer non structuré, un noeud rejoint le réseau en suivant certaines règles.

**avantages du peer to peer non structuré** :

* il supporte les requêtes complexes
* il s’organise lui même
* simple à mettre en place

**contraintes du peer to peer non structuré** :

* on ne peut pas être sûre de la position d’un objet dans le réseau
* pas évolutif en raison de la surcharge de messages élevée

### Le peer to peer structuré

Le peer to peer structuré :

* impose une contrainte de typologie
* quand un noeud rejoint le réseau il doit suivre des règles strictes
* les contraintes imposées permettent une recherche efficace des données

**avantages du peer to peer structuré** :

* garantie l'existence des données recherchées dans le système, les données sont trouvées avec un temps O(log n) n = nombre de noeuds

**contraintes du peer to peer structuré** :

* demande beaucoup d’effort pour maintenir le système

### Le peer to peer hybride

utilise des techniques de **flooding** pour rechercher des données et le **DHT** pour localiser les données rarement utilisées.

**avantage** :

Ils peuvent gérer efficacement les demandes de ressources hautement répliquées et

à ceux rarement demandés

**contraintes** :

Il peut être difficile de distinguer les ressources rarement utilisées des ressources populaires

### Hierarchical P2P

introduction de couche qui permettent de réduire la charge réseau.

super peer :

* peer avec de grosses ressources physiques (cpu gpu ..)
* ils agissent comme un proxy pour les noeuds ordinaires

Leaf-nodes :

* noeud avec des capacités limitées
* ils sont connectés à un super noeud

## protocoles de peer to peer

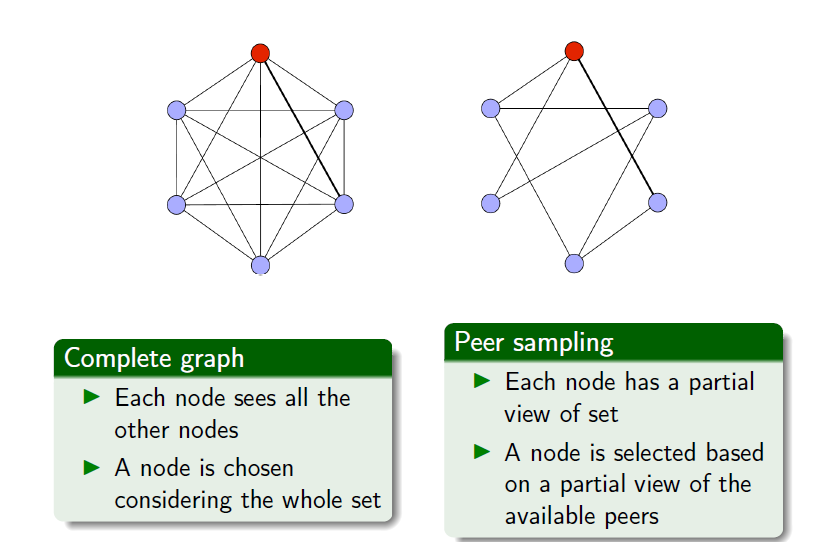
### Gossiping

Désigne l'échange probabiliste d'informations entre deux nœuds.

**contraintes** :

* la sélection d’une paire est aléatoire
* seul les informations locales sont disponibles dans chaque noeud
* chaque noeud exécute le même protocole

### Random sampling



### Distributed Hash Table (DHT)

* c’est un système décentralisé qui fournit un système de recherche similaire à une table de hachage
* chaque noeud participe à la recherche grâce à leur clef
* la clef est obtenue grâce à une fonction de hachage (sha1 sha2 sha256..)

**avantages** :

* Prise en charge de requêtes de correspondance exacte efficaces
* Prend en charge une répartition uniforme des charges entre les nœuds, garantissant une bonne performance

**contraintes** :

* La latence de recherche dans les superpositions basées sur DHT peut affecter la performances des applications s'exécutant sur celles-ci
* Ils ne fournissent pas l'anonymat

### Chord

* c’est un protocole de stockage et de recherche de données
* les noeuds sont organisés en cercle
* les ID des noeuds sont codés sur 2m-1 avec m le nombre de noeud

