

Les systèmes répartis

- [Qu'est-ce qu'un système réparti ?](#)
 - [Structure centralisée](#)
 - [Structure répartie](#)
 - [Structure mixte](#)
 - [La notion du temps](#)
 - [Exemple](#)
 - [L'ordre partiel](#)
 - [Utilisation de l'ordre partiel](#)
 - [Ordre Total Strict](#)
 - [Exemple](#)
 - [Ordonancement au moyen d'estampilles](#)
 - [L'exclusion mutuelle - Algorithme](#)
-

Qu'est-ce qu'un système réparti ?

Exemples de systèmes :

- service de courrier électronique ;
- systèmes de fichiers distants montés localement (NFS) ;
- systèmes de réservations (voyage + hotel + vehicule) ;
- ...

Soit un système informatique constitué d'un ensemble de stations reliées entre elles par un moyen de communication.

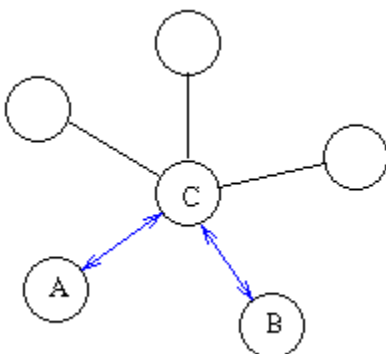
On veut implémenter un système de messagerie.

Un usager doit pouvoir émettre ou retirer des messages de n'importe quelle station.

Plusieurs implémentations possibles.

- structure **centralisée** ;
- structure **décentralisée - ou répartie** ;
- structure **mixte**.

Structure centralisée



Tous les courriers sont stockés sur C (station centrale).

1 usager = 1 boîte aux lettres sur C.

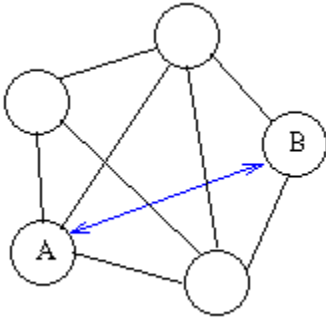
⇒ volume de stockage important sur C.

⇒ disponibilité du service = disponibilité de C.

⇒ 1 opération = 1 transfert d'informations.

Structure répartie

1 usager = 1 boîte aux lettres = 1 station de rattachement



Retrait d'un message sur station de rattachement = opération locale.

Depot ou retrait sur une autre station ⇒ échange d'informations.

⇒ dictionnaire Usager/Station-de-rattachement.

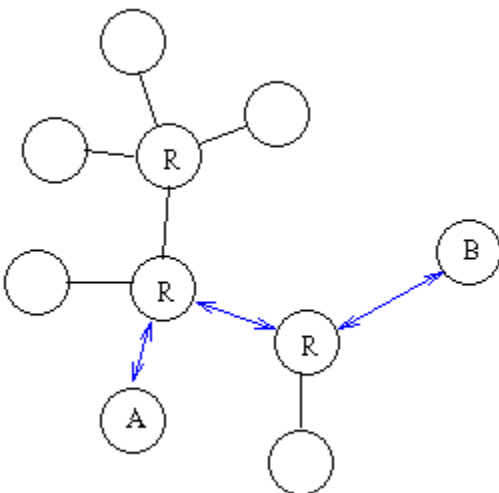
Disponibilité du système accrue.

⇒ Panne d'une station = empêche la réception des messages pour les usagers de cette station.

⇒ Volume de stockage global réparti sur l'ensemble des stations.

⇒ Une copie du répertoire sur chaque machine (attention à la mise-à-jour).

Structure mixte



On ajoute des stations **relais**.

Une station est reliée à une station relai et une seule.

Le relai stocke les messages des stations qui lui sont rattachées.

Tout usager dépend donc d'un seul relai.

Un message passe forcément par un relai.

Seuls les relais possèdent un dictionnaire.

Panne d'un relai ➡ pénalise un ensemble d'usagers.

Moins de répertoires à mettre à jour.

Avantage : si une fraction importante du nombre de messages est échangée entre les utilisateurs d'un même relai.

Un station peut être un simple terminal.

En résumé : on préfère définir un système réparti par les services qu'il offre :

- Structure modulaire.
- Disponibilité accrue (même si un module tombe en panne, le service est rendu).
- Décentralisation, destinée à accroître la localité du traitement avec la création de copies multiples, la multiplication des allocateurs et le changement d'implémentation du code et des données.

La notion du temps

Un processus P_1 sur une station A veut coopérer avec un processus P_2 sur une station B .

Comment faire pour savoir qu'un événement e de P_1 s'est déroulé avant (ou après) un événement e' de P_2 ?

➡ Il n'existe pas d'**horloge commune** (ou temps global).

À un instant donné, quel est l'**état du système** ?

➡ Il faut que les processus échangent des messages.

Exemple

Soit un parking. Les usagers sont en **compétition** pour l'utilisation des places.

1. Parking avec un accès unique, surveillé par un seul gardien : connaissance partielle de la situation

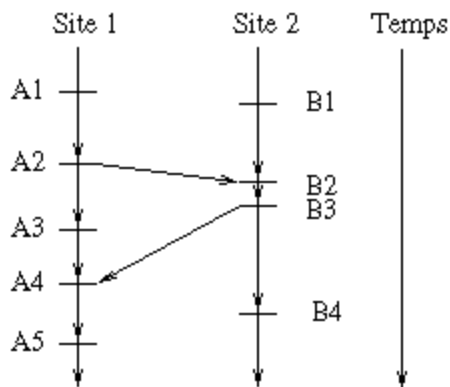
➡ refus d'entrer alors qu'une voiture est en route vers la sortie.

2. Plusieurs accès avec un gardien à chaque accès : chaque gardien connaît avec retard les actions des autres gardiens

➡ 2 voitures sont autorisées à rentrer alors qu'il y a 1 seule place libre.

➡ les gardiens doivent **coopérer**.

L'ordre partiel



Ordre local des évènements :

$A1 \rightarrow A2 \rightarrow A3 \rightarrow A4 \rightarrow A5$
 $B1 \rightarrow B2 \rightarrow B3 \rightarrow B4$

Échanges de messages :

$A2 \rightarrow B2 \text{ et } B3 \rightarrow A4$

Transitivité :

$A1 \rightarrow A2 \rightarrow B2 \rightarrow B3 \rightarrow B4$
 $B1 \rightarrow B2 \rightarrow B3 \rightarrow A4 \rightarrow A5$
 $A1 \rightarrow A2 \rightarrow B2 \rightarrow B3 \rightarrow A4 \rightarrow A5$

Exemples d'évènements incomparables :

$B1$ et $A1, A2, A3$
 $A3$ et $B2, B3, B4$

Utilisation de l'ordre partiel

Hypothèses :

1. tout site communique avec tout autre site (maillage logique) ;
2. pas d'erreur de transmission ni de perte / duplication de messages ;
3. ordre de réception = ordre d'émission ;
4. une panne d'un site est détectée et signalée aux autres sites.

Producteur - Consommateur

Le producteur P et le consommateur C sont sur des sites différents (S_1 et S_2).

NP = nombre de productions et NC = nombre de consommations.

C peut consommer ssi $NP - NC > 0$.

P peut produire ssi $NP - NC < N$.

Implémentation :

- sur S_1 , NP = le nombre de productions faites et NC' , image de NC , incrémenté à chaque réception d'un message de C .
- sur S_2 , NC = le nombre de consommations faites et NP' , image de NP , incrémenté à chaque réception d'un message de P .

Utilisation des **compteurs d'évènements** 1 compteur = variable entière non-décroissante, associée à une classe E .

Primitives :

- $avancer(E)$: augmente de 1 la valeur du compteur (arrivée d'un évènement de classe E) ;
- $consulter(E)$: fournit la valeur du compteur ;
- $attendre(E, n)$: suspend le processus tant que la valeur du compteur est inférieure à n .

- 2 compteurs d'évènements NP' et NC' initialisés à 0.

- 2 variables entières NP et NC initialisées à 0.

Producteur P	Consommateur C
$attendre(NC', NP-N+I)$; { passage lorsque $NP-NC'$ } production ; $avancer(NP')$; $NP := NP+I$; 	$attendre(NP', NC+I)$; { passage lorsque $NP'-NC'>0$ } consommation ; $avancer(NC')$; $NC := NC+I$;

Ordre Total Strict

Ordre partiel dès fois insuffisant.

On doit pouvoir avoir un **ordre total strict**.

Par exemple : allocation de ressources.

En **centralisé** : les requêtes et avis de libération sont ordonnés dans une file manipulée en Exclusion-Mutuelle, puis traitées en séquence.

En **réparti** :

- si un seul message à la fois arrive, l'ordonnement est strict ;
- si plusieurs messages arrivent en même temps, il faut les ranger en EM dans une file locale ;
- si on veut conserver ordre de réception = ordre d'émission, il faut pouvoir ordonner globalement l'émission des messages.

Exemple

Parking avec 3 gardiens $G1$, $G2$ et $G3$. état initial = 100 places libres.

Les trois gardiens diffusent les messages suivants :

- $M1$: 20 places de plus sont libres ;
- $M2$: 10 places de plus sont occupées ;
- $M3$: je réserve 10% des places pour le nettoyage.

Résultat (exemples) :

Ordre d'envoi	Séquence 1 (msg, valeur)	Séquence 2 (msg, valeur)	Séquence 3 (msg, valeur)	Séquence 4 (msg, valeur)
init	-, 100	-, 100	-, 100	-, 100
1	M1, 120	M1, 120	M3, 90	M2, 90
2	M3, 108	M2, 110	M1, 110	M3, 81
3	M2, 98	M3, 99	M2, 100	M1, 101
final	-, 98	-, 99	-, 100	-, 101

Ordonancement au moyen d'estampilles

À chaque message, on associe un numéro appelé **estampille**.

Cette estampille est la valeur instantannée, sur le site d'émission, d'une horloge logique locale à ce site.

Les horloges des différents sites sont recalées grâce à un dialogue.

Construction d'un **ordre total strict** [Lamport 78] :

Chaque site s est munie d'un compteur à valeurs entières H_s , appelé **horloge logique**.

H_s est incrémenté entre deux évènements successifs.

Un site e qui émet un message le marque d'une estampille E égale à la valeur courante de H_e .

À la réception du message, le site récepteur r met-à-jour H_r ainsi :

si $H_r < E$ **alors** $H_r := E + 1$ **fin**

L'évènement << réception du message >> est daté par H_r .

On a une relation d'ordre **total** ⇨

Soient a et b deux évènements des sites (resp.) i et j alors :

$a \rightarrow b \equiv H_i(a) < H_j(b)$

Pour avoir un ordre **total et strict** ⇨

On associe le numéro du site à l'estampille.

$a \rightarrow b \equiv (H_i(a) < H_j(b))$ ou $(H_i(a) < H_j(b) \text{ et } i < j)$

L'exclusion mutuelle - Algorithme

Solution centralisée : un site central reçoit les requêtes d'EM, les met dans une file FIFO.

On veut répartir l'algorithme (*i.e.* pas de site central).

⇨ une file par site.

Chaque site reçoit les requêtes et avis de libération de tous les autres sites.

On veut un ordre total strict (estampillage par horloge logique + numéro de site).

Pour qu'un site puisse prendre sa décision, il doit avoir reçu un message de chaque autre site (pas de message en transit).

Les envois de messages :

1. Messages (REQ, H_i, i) (de i vers tous) = le site i veut entrer en SC.
2. Messages (REL, H_i, i) (de i vers tous) = le site i sort de SC.
3. Messages (ACQ, H_i, i) (de i à j) = le site i a reçu du site j un (REQ, H_j, j) .

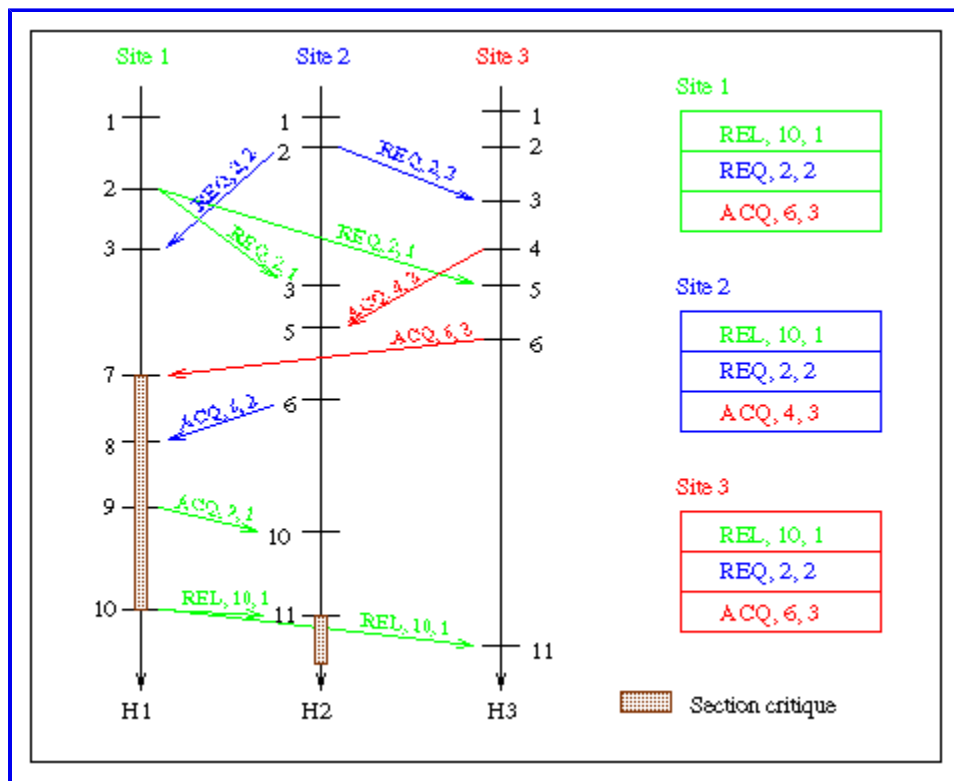
Chaque site i maintient une file de messages avec 1 message par site. Au départ, chaque file contient $M_i = (REL, H_{init}, i)$.

H_{init} est la même pour tous les sites.

Si un site reçoit (REQ, H_i, i) ou (REL, H_i, i) , ce message remplace M_i .

Si un site reçoit (ACQ, H_i, i) , ce message remplace M_i sauf si M_i est un (REQ, H_i, i) .

Le site i peut rentrer en SC si sa requête (REQ, H_i, i) précède tous les autres messages de la file d'attente.



[Retour au sommaire.](#)