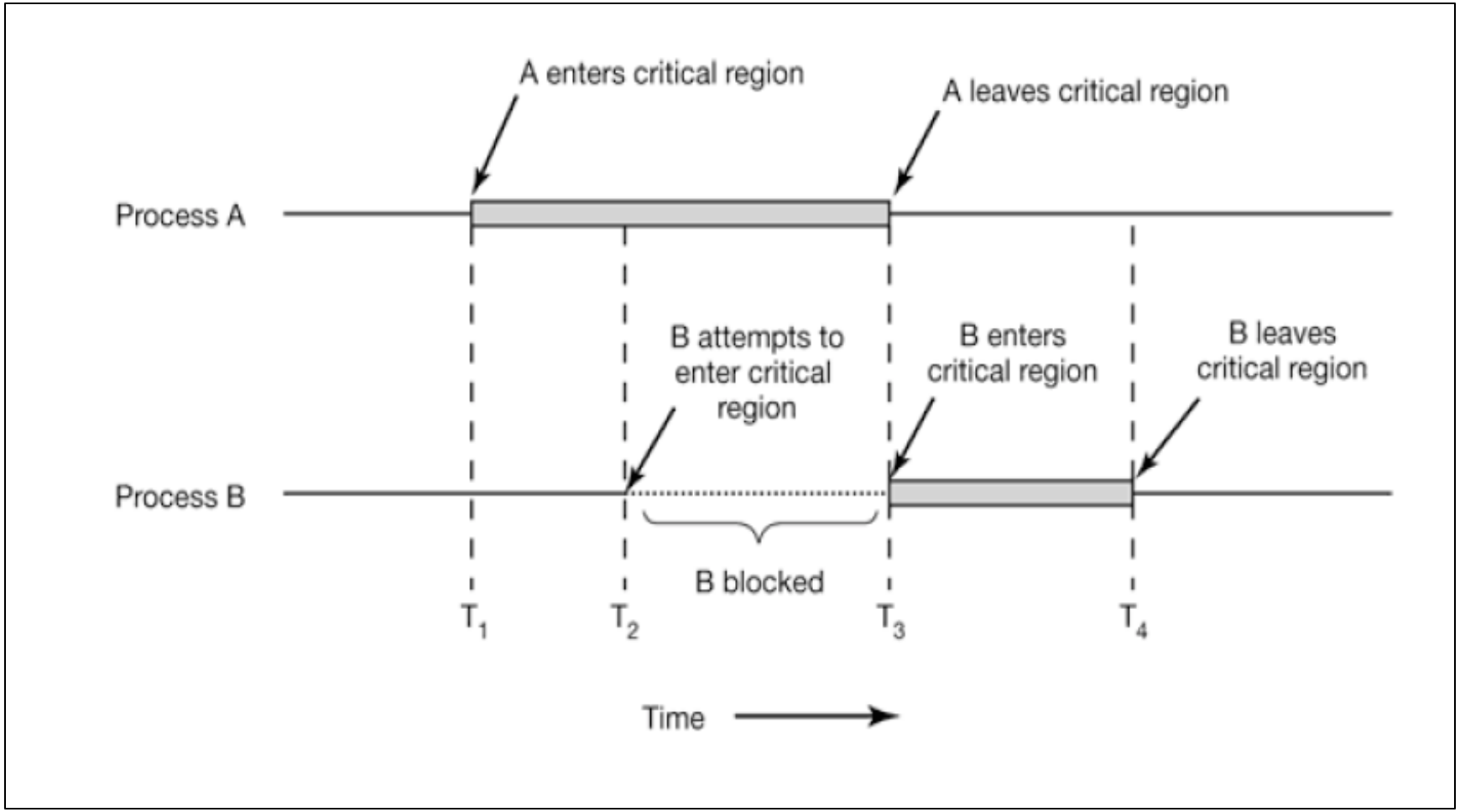
Cours 3

Applications de datation logique

- Horloge linière.

2.3. Application de l'horloge logique aux problèmes de synchronisation : 1. Exclusion mutuelle
Rappel



Exclusion mutuelle (Rappel) cont

{État, = dehors) acquérir {état, = demandeur)

{état, = dedans) libérer {état, = *dehors)*

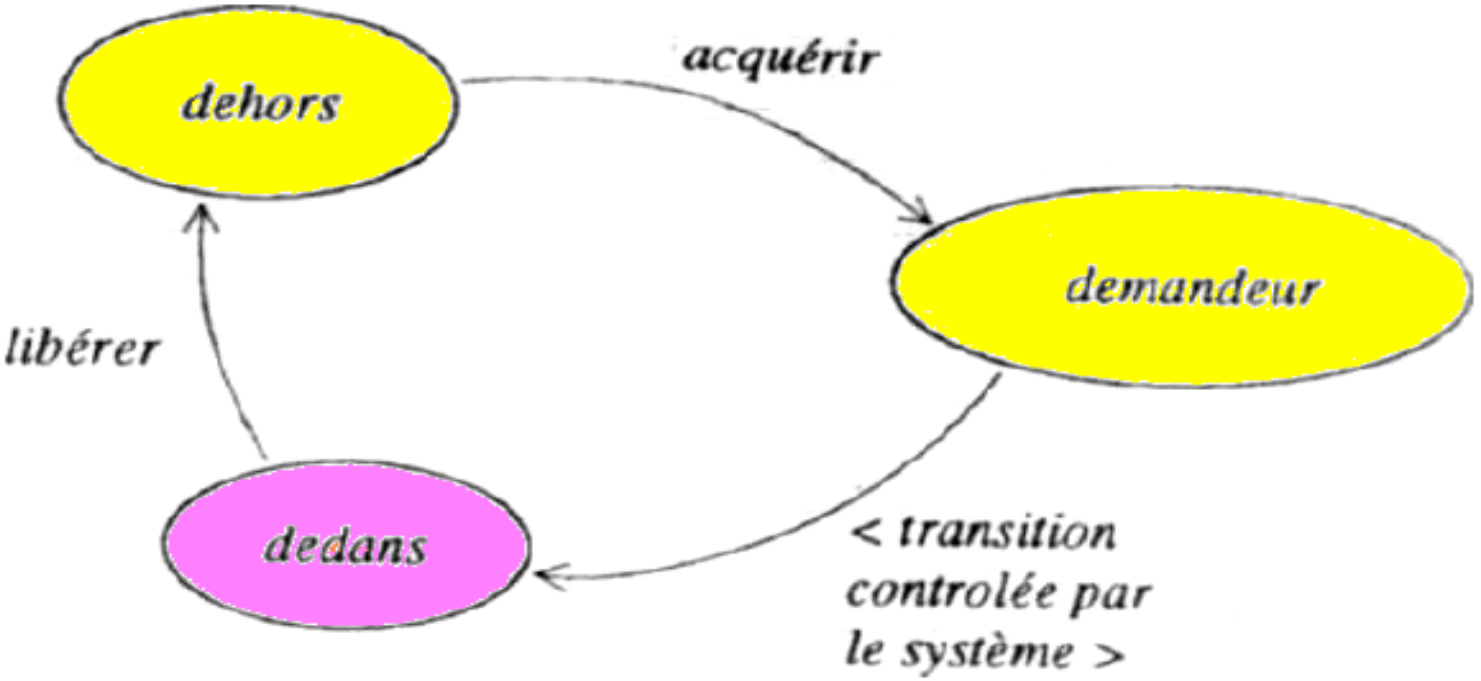


Figure 1.1 : Transitions d'un processus

Cont..

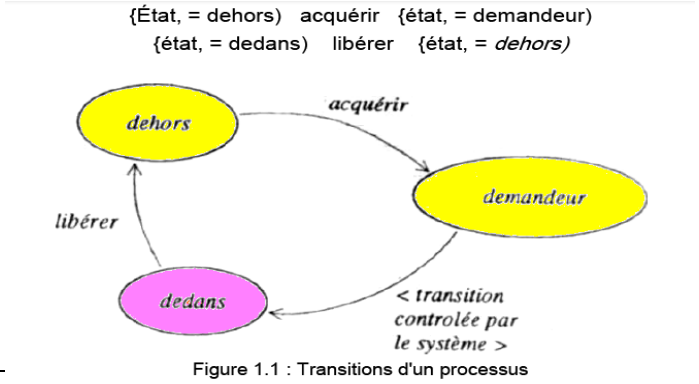


Figure 1.1 : Transitions d'un processus

- **Une solution au problème de l'exclusion mutuelle consiste à offrir un protocole (encore appelé règles de comportement ou algorithme) capable de faire passer les processus, qui se trouvent dans l'état demandeur, à l'état dedans et tel que les 2 propriétés suivantes soient vérifiées :**
  - **propriété de sûreté (safety)**
  - **propriété de vivacité (liveness)**

Cont..

1. **propriété de sûreté (safety) : à tout instant il y a au plus un processus Pi tel que état(Pi) = dedans.**
2. **propriété de vivacité (liveness) : tout processus qui se trouve dans l'état demandeur doit passer au bout d'un temps fini dans l'état dedans.**

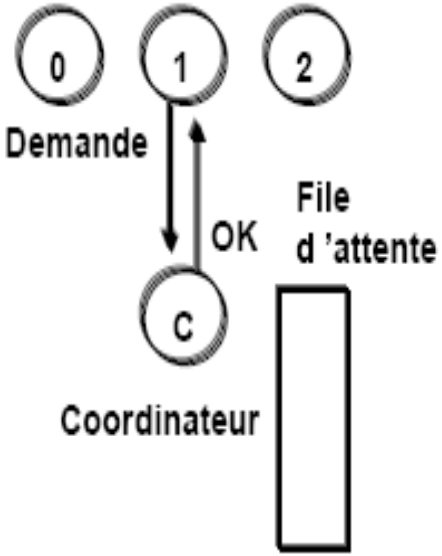
Cont..

- **Dans les systèmes centralisés**
  - **Sémaphores**
  - **Moniteurs**
  - **..**
- **Contrainte dans un système distribué**
  - **Les mécanismes basés sur une mémoire commune ne sont plus valables.**

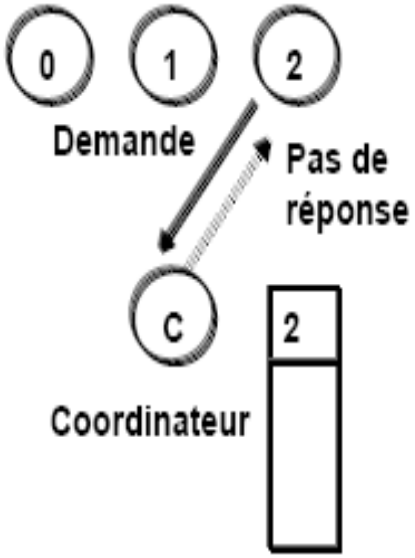
Exclusion mutuelle dans les systèmes distribués
Principe

- Supposons qu'un ensemble de processus partagent une unique ressource. On veut un algorithme de synchronisation qui fait en sorte que :
  - Un processus qui a obtenu l'autorisation d'accès à la ressource la libère avant qu'un autre processus puisse obtenir cette autorisation.
  - Les différentes demande d'accès sont servies dans l'ordre où elles ont été émises (ce qui n'est pas forcément l'ordre dans lequel elles ont été reçues).
  - Si tout processus qui a eu la ressource finit par la libérer, alors tout processus qui demande la ressource finira par l'obtenir.

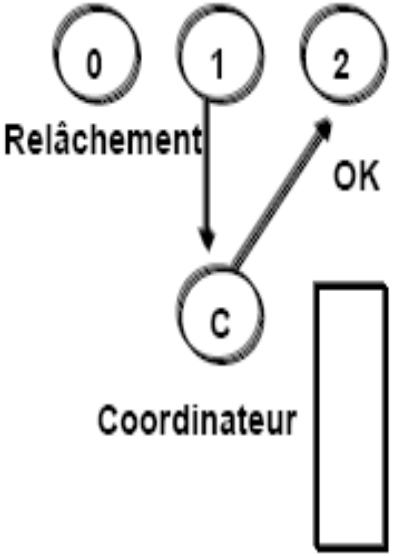
Exclusion mutuelle
a- Un algorithme centralisé



Situation 1

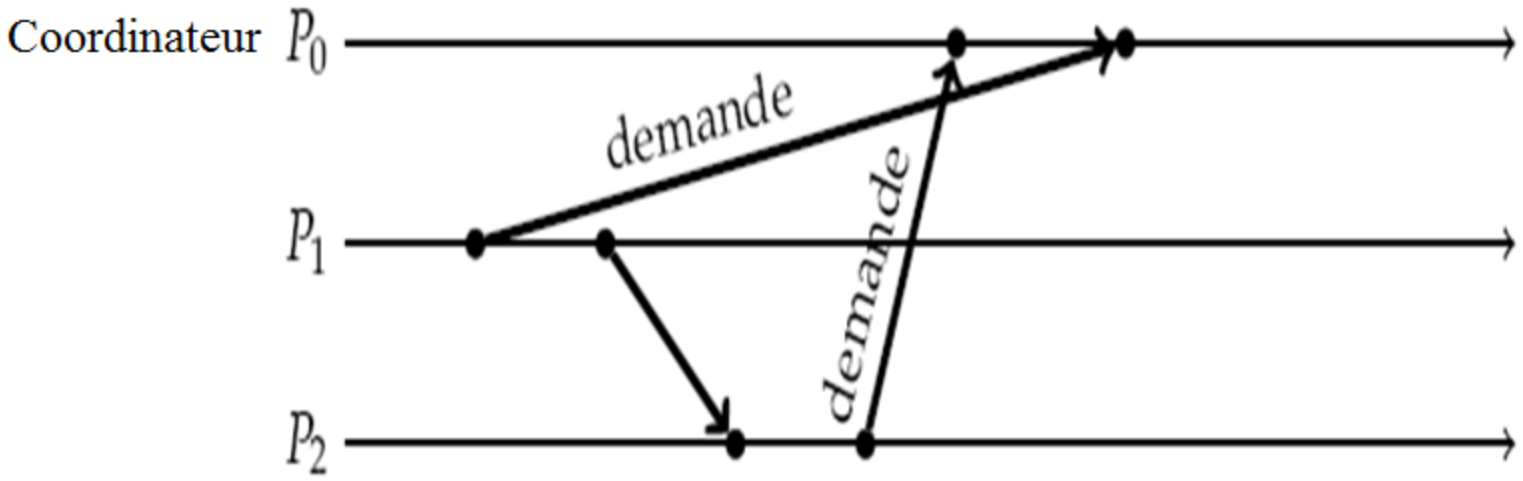


Situation 2

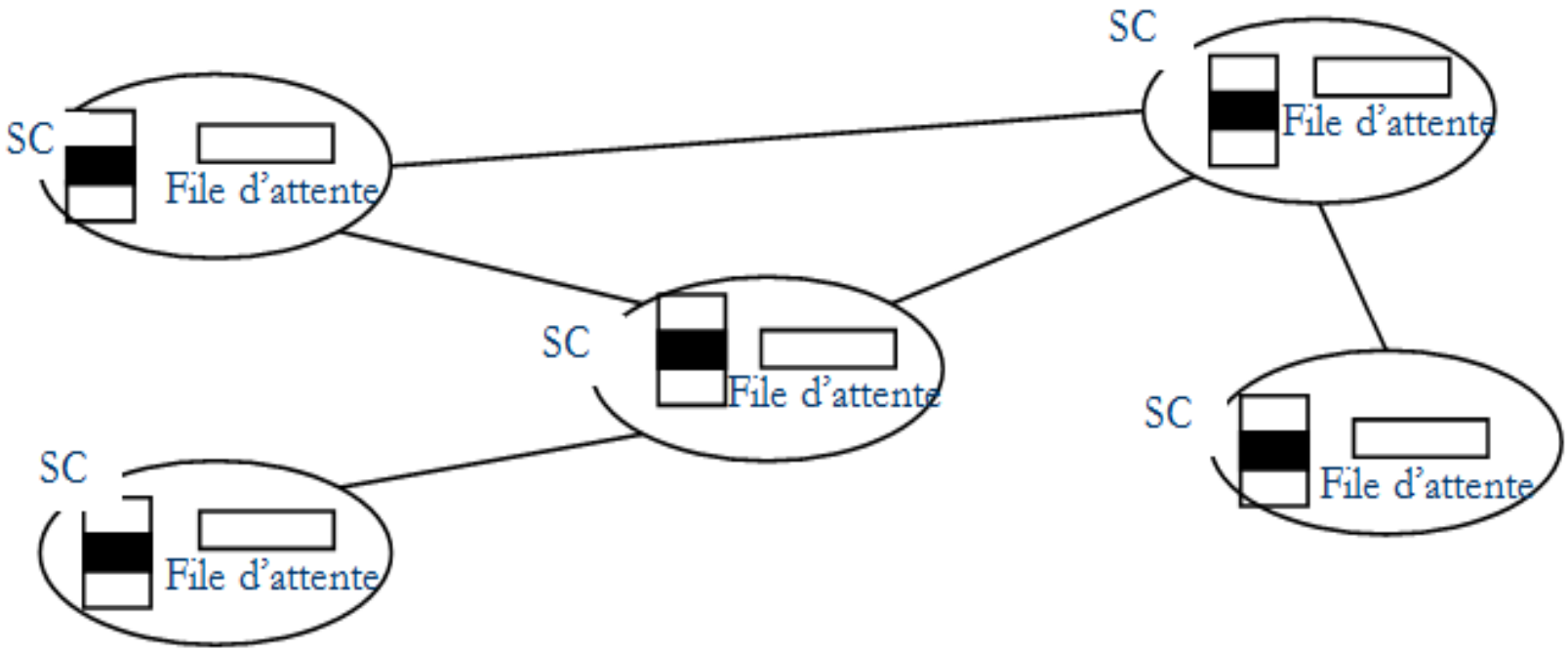


Situation 3

Exclusion mutuelle
a- Un algorithme centralisé -problème-



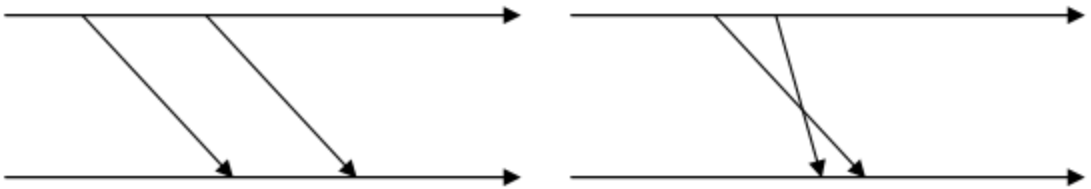
Exclusion mutuelle
b- Algorithme de Lamport



SC : Section Critique

Hypothèse

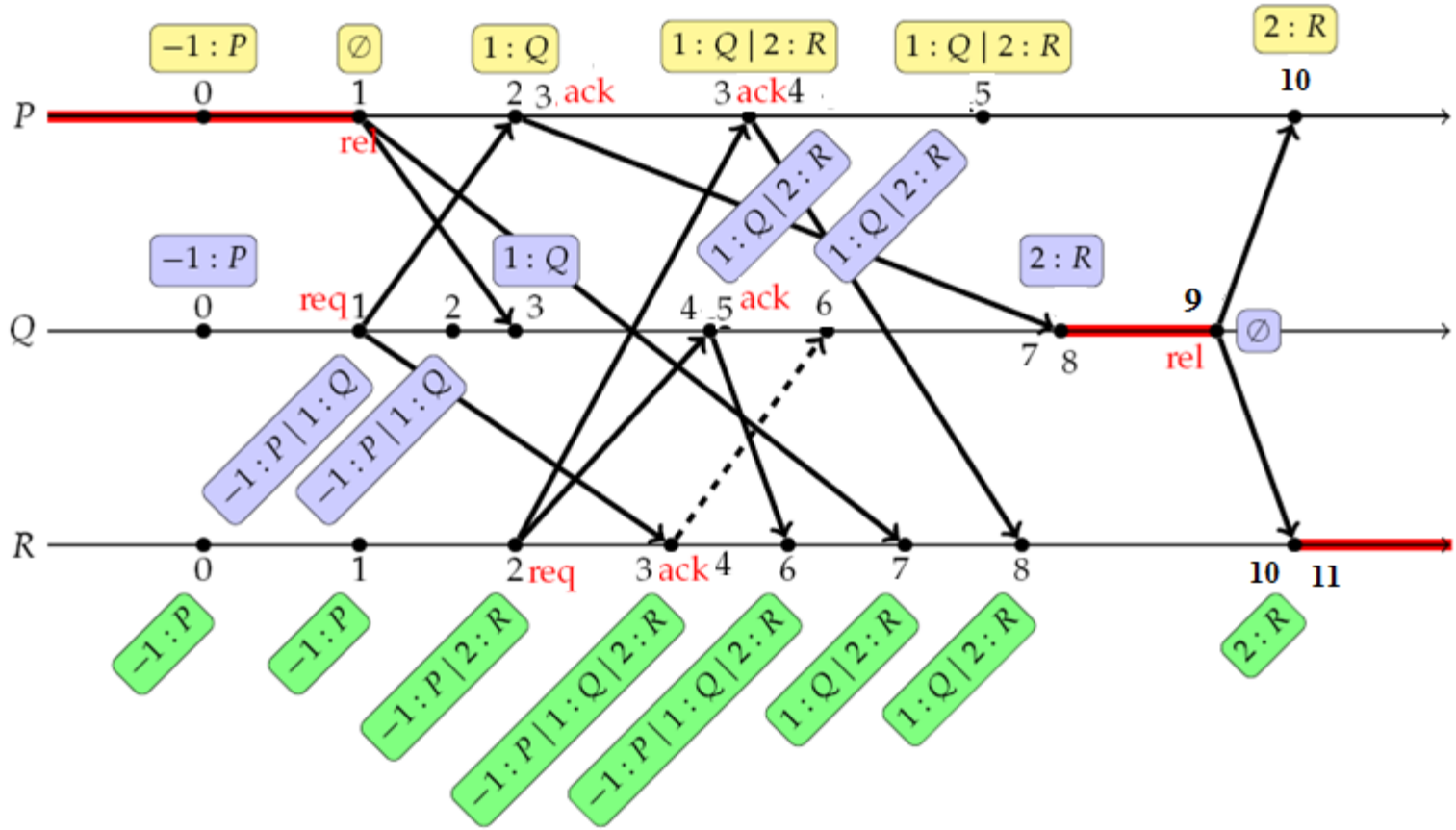
- **le nombre N des sites est connu**
- **les canaux sont FIFO**
- **ordre total réalisé par horloge logique**



Exclusion mutuelle
b- Algorithme de Lamport

- **Un process Pi voulant accéder à sa SC, envoie un message de demande *(Request message)* , contenant aussi son estampille LC(i), à tous les autres sites (DIFFUSION) y compris lui même.**
- **Un process Pj recevant la demande de Pi met à jour sa file locale et répond par un message d'acquittement (*Reply message*) contenant aussi son estampille.**
- **Pi peut entrer à sa SC si :**
  - **il a reçu tous les acquittements de tous les (N-1) process.**
  - **sa demande est la plus ancienne parmi les autres demandes d'accès dans sa file d'attente.**
- **À la sortie Pi diffuse un message de sortie (*Release Message*) contenant aussi son estampille à tous les autres process.**

Exclusion mutuelle
b- Algorithme de Lamport -exemple-



Diffusion

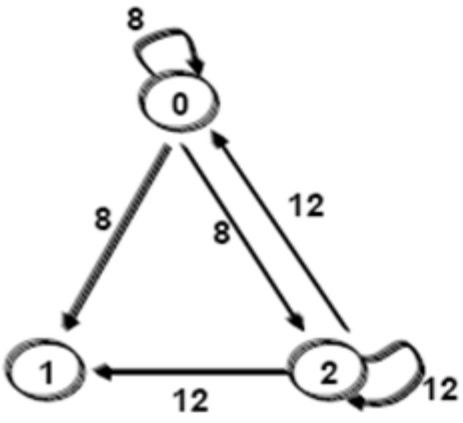
- Un seul événement, une seule date.
- Exemple des messages : req, lib.

Algorithme d'exclusion mutuelle de Leslie Lamport -implémentation *incomplète*-

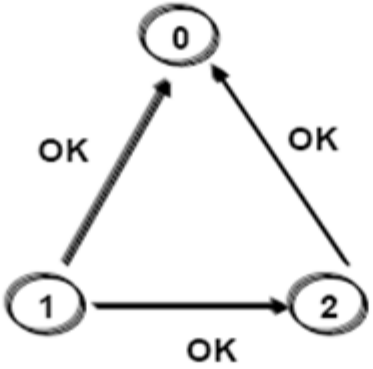
- **Structure de données sur chaque site : tableau de N messages (Mij), 1 par site.**
- **Initialement sur Si : pour tout j Mij := (lib, 0, Sj)**
- **Demande d'entrée d'un site Si en section critique : diffusion de (req, H(req), i) à tous les sites, Si compris.**
- **Réception de M = (req, H(req), j) ou (lib, H(lib), Sj) => Mij := M**
- **Réception de M = (acq, H(acq), Sj) => si Mij ≠ (req, H(req), j) alors Mij := M (puisque il faut juste comparer les estampilles des requêtes et pas des ack ni de lib)**
- **Si entre en section critique quand sa demande Mii << Mij pour tout j**

Exclusion mutuelle
c- Un algorithme distribuée(R & Agrawala-1981)

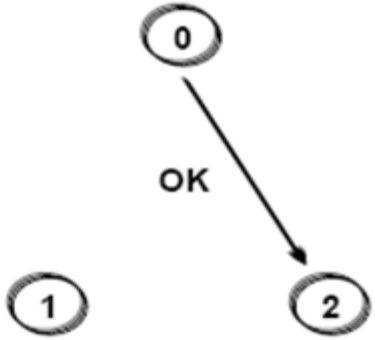
- **Hypothèse**
- **le nombre N des sites est connu**
- **les canaux sont quelconques (hypothèse FIFO non nécessaire)**
- **ordre total réalisé par horloge logique**



P0 et P2 veulent entrer dans une même section critique

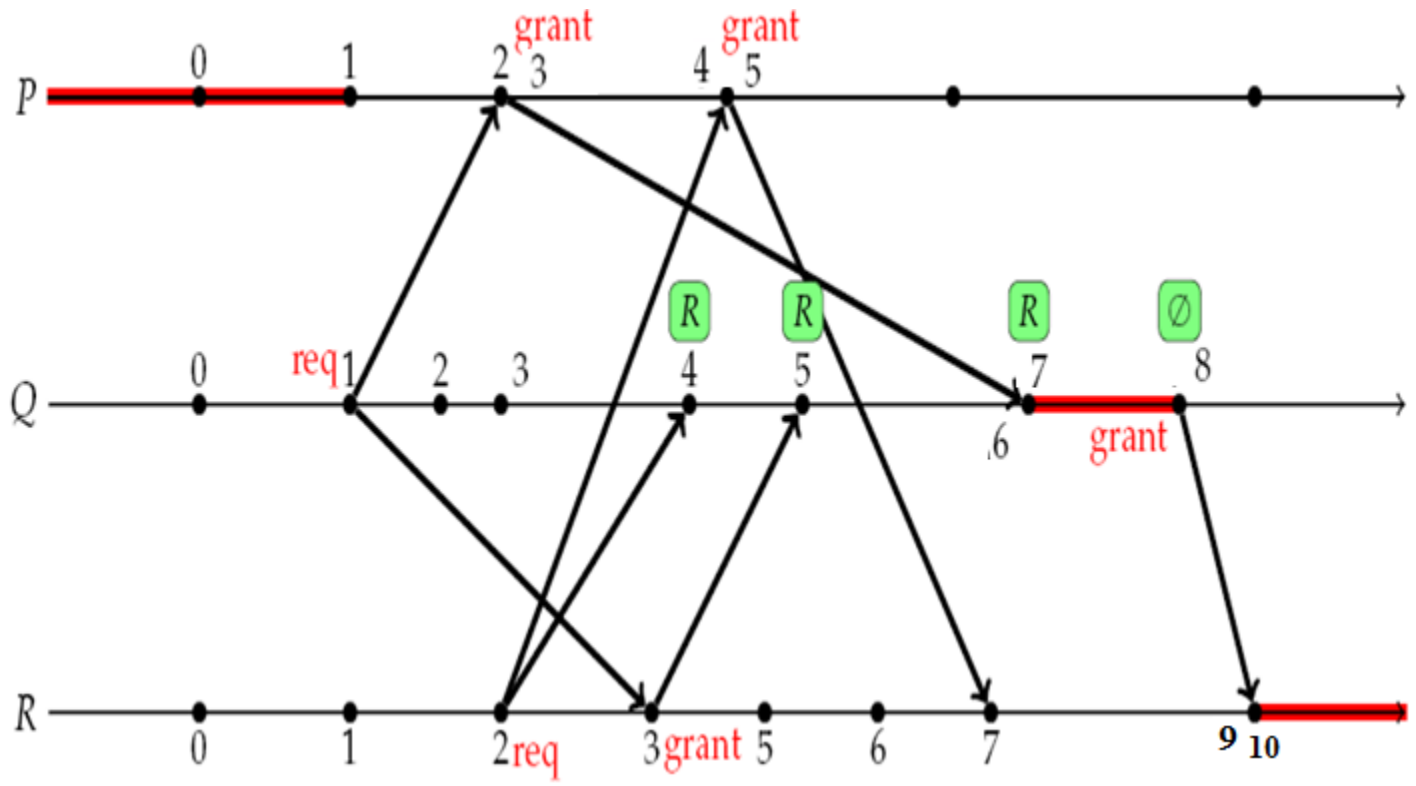


P0 est plus ancien, il gagne et entre en section critique

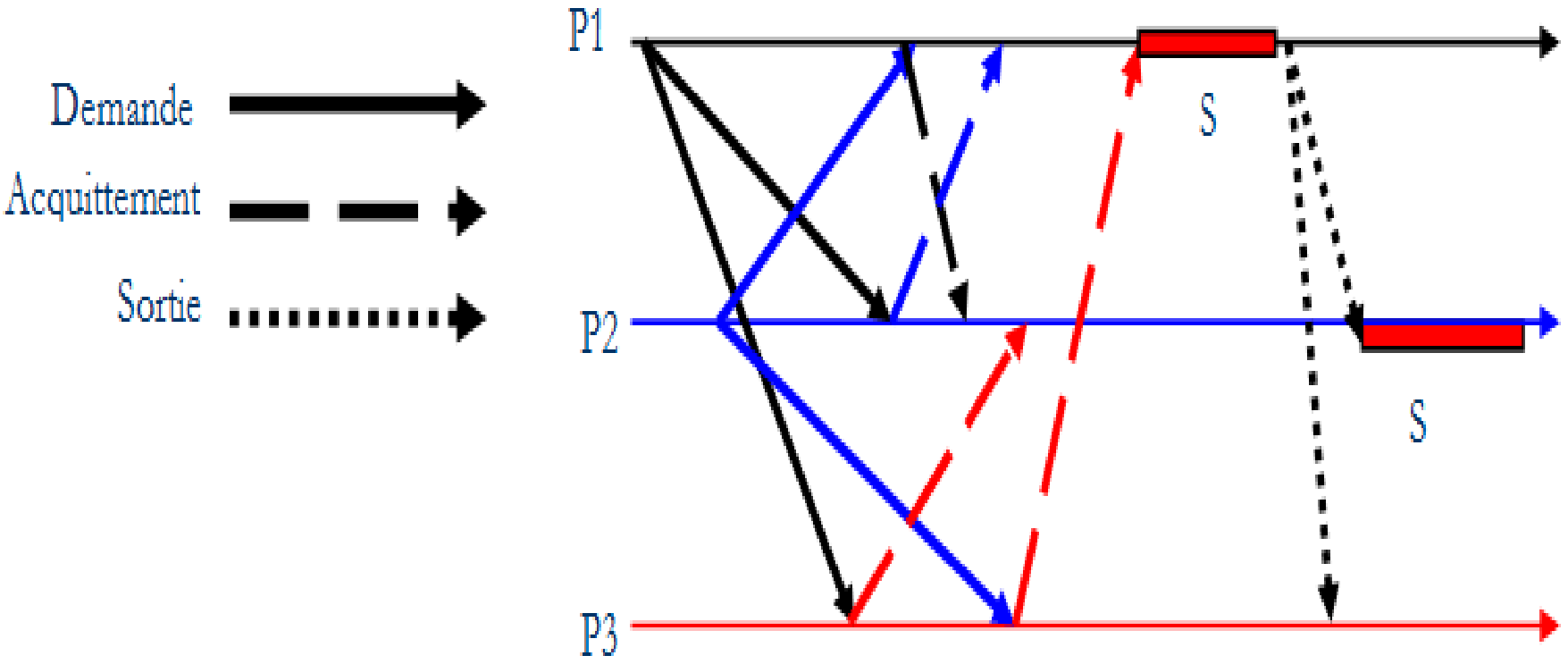


P2 entre en section critique

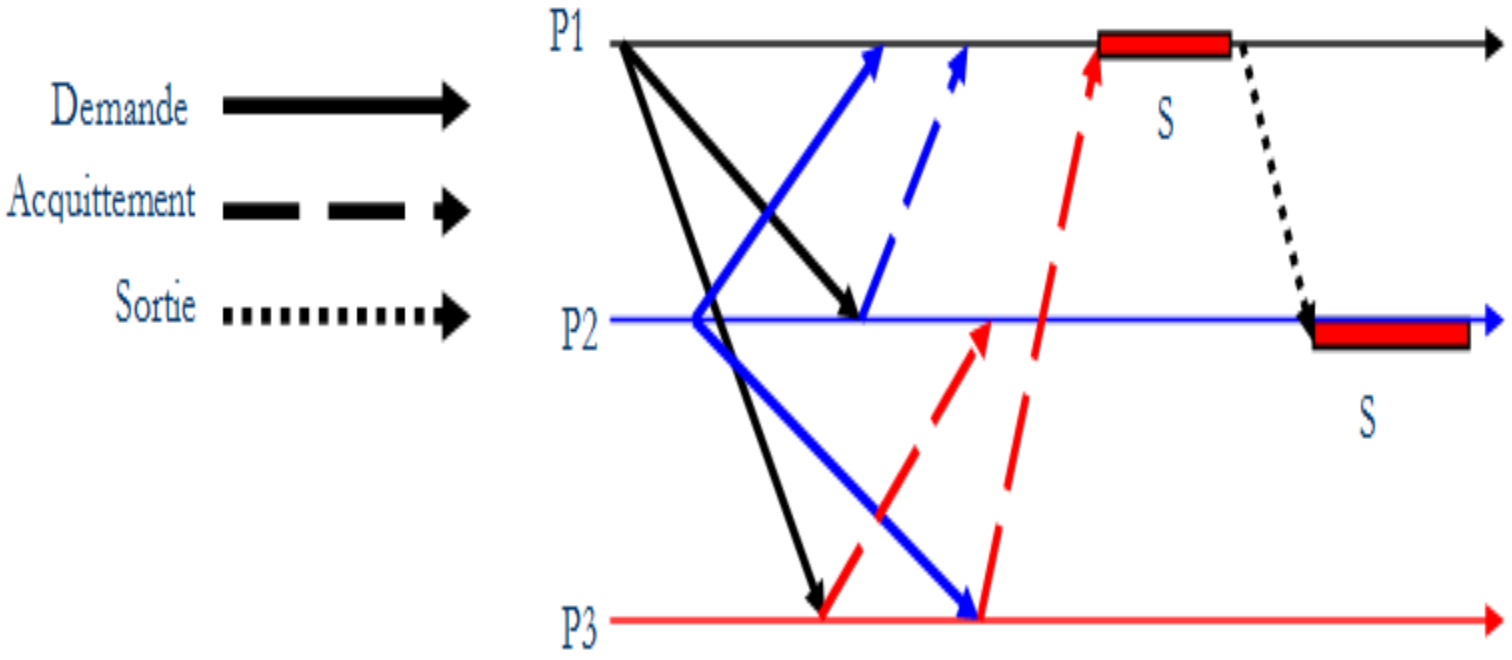
Exclusion mutuelle
c- R & Agrawala



Algorithme de Lamport Exemple
Complexité 3(n-1) messages



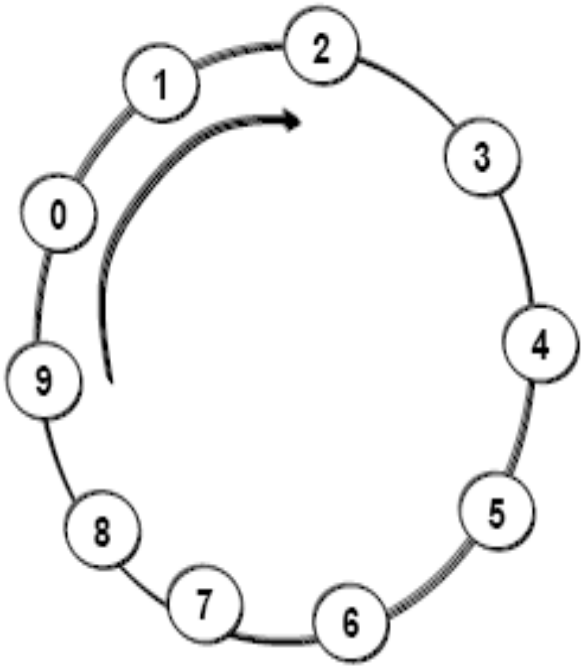
Un algorithme distribuée(R & Agrawala-1981) Exemple Complexité 2(n-1) messages



Question

- Discuter les hypothèses des deux algorithmes.

Exclusion mutuelle, d'autres algorithmes
d- Un algorithme de type anneau à jeton



- établir un anneau logique entre les processus
- introduire un jeton dans l 'anneau
- le processus possédant le jeton peut entrer en section critique. Il garde le jeton jusqu'à sa sortie de la section critique. Il passe ensuite le jeton au processus suivant dans l 'anneau logique
- sinon, il passe le jeton au processus suivant dans l 'anneau logique

Problèmes

- la perte de jeton -> algorithmes de détection et régénérer le jeton
- la panne d 'un processus -> régénérer l 'anneau ou l 'algorithme de détection de la mort du voisin

Exclusion mutuelle
Conclusion et comparaison

Type d'algorithme	Nombre de messages par entrée/sortie	Temps avant entrée	Problèmes
Centralisé	3	2	Plantage du coordinateur
Distribué	3(n-1) 2(n-1)	3(n-1) 2(n-1)	Plantage d 'un processus
Anneau à jeton	1 à ∞	0 à n-1	Perte de jeton, plantage de processus

L 'algorithme distribué est beaucoup plus sensible au plantage que l 'algorithme centralisé !