Exclusion mutuelle de groupe dans les systèmes distribués

Group mutual exclusion in distributed systems

Ousmane THIARE

Université de Cergy-Pontoise
L.I.C.P (Laboration d'Informatique de Cergy-Pontoise)

25 Juin 2007

Plan de la présentation

1- Systèmes distribués

Systèmes distribués

- Unités de traitements interconnectées
- Coopération pour exécuter une tâche globale
- Les échanges d'informations se font
  - soit par échanges de messages
  - soit par l'intermédiaire de mémoires partagées

Systèmes distribués

- Unités de traitements interconnectées
- Coopération pour exécuter une tâche globale
- Les échanges d'informations se font
  - soit par échanges de messages
  - soit par l'intermédiaire de mémoires partagées

Systèmes distribués

- Unités de traitements interconnectées
- Coopération pour exécuter une tâche globale
- Les échanges d'informations se font
  - soit par échanges de messages
  - soit par l'intermédiaire de mémoires partagées

Systèmes distribués

- Unités de traitements interconnectées
- Coopération pour exécuter une tâche globale
- Les échanges d'informations se font
  - soit par échanges de messages
  - soit par l'intermédiaire de mémoires partagées

Systèmes distribués

- Unités de traitements interconnectées
- Coopération pour exécuter une tâche globale
- Les échanges d'informations se font
  - soit par échanges de messages
  - soit par l'intermédiaire de mémoires partagées

2- Exclusion mutuelle de groupe basée sur les quorums

Algorithme basé sur les quorums

Quorum

Définition (1)

Une collection d'ensembles, Q, est un quorum dans U si :

1. $(\forall G \in Q)[G \neq \emptyset$ *et* $G \subseteq U]$
2. ***(Minimalité) :*** $(\forall G, H \in Q)[G \not\subseteq H]$**.**

Les ensembles $G \in Q$ sont appelés *quorums*. Par exemple, soit $U = \{a, b, c, d\}$. Alors, $Q = \{\{a, b\}, \{b, c\}\}$ est un quorum dans U.

Définition (2)

Un quorum Q est une coterie sur U si la propriété d'intersection est satisfaite, c'est-à-dire, $(\forall G, H \in Q)[G \cap H \neq \emptyset]$.

Algorithme basé sur les quorums

Problèmes fondamentaux liés à l'exclusion mutuelle

- Exclusion mutuelle classique [Dijkstra 1965]
  - Une ressource unique à partager entre N demandeurs potentiels
  - File d'attente
- Le dîner des philosophes [Dijkstra 1978]
- Les philosophes buveurs [Chandy et Misra 1984]
- Lecteurs-Rédacteurs [Courtois, Heymans et Parnas 1971]
  - Au plus un rédacteur
  - Plusieurs lecteurs simultanément
  - Jamais de lectures et d'écritures simultanément

Algorithme basé sur les quorums

Problèmes fondamentaux liés à l'exclusion mutuelle

- Exclusion mutuelle classique [Dijkstra 1965]
  - Une ressource unique à partager entre N demandeurs potentiels
  - File d'attente
- Le dîner des philosophes [Dijkstra 1978]
- Les philosophes buveurs [Chandy et Misra 1984]
- Lecteurs-Rédacteurs [Courtois, Heymans et Parnas 1971]
  - Au plus un rédacteur
  - Plusieurs lecteurs simultanément
  - Jamais de lectures et d'écritures simultanément

Algorithme basé sur les quorums

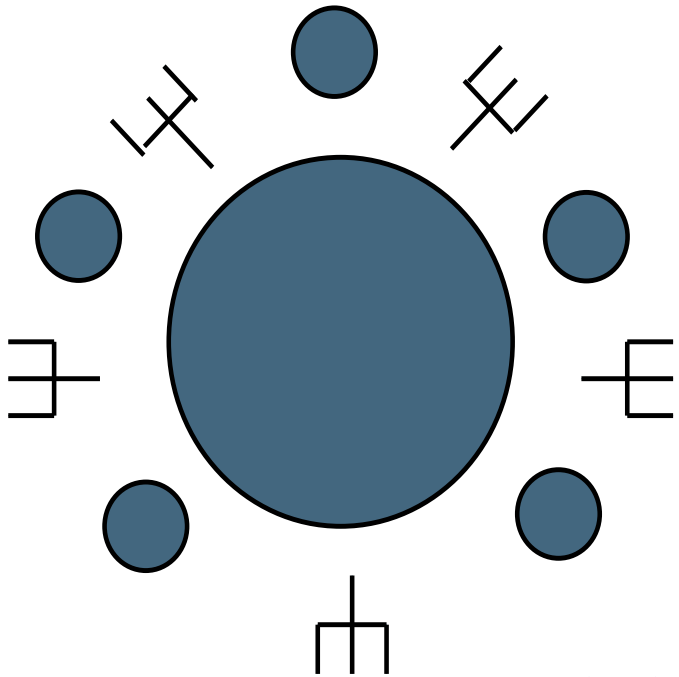
Problèmes fondamentaux liés à l'exclusion mutuelle

- Exclusion mutuelle classique [Dijkstra 1965]
  - Une ressource unique à partager entre N demandeurs potentiels
  - File d'attente
- Le dîner des philosophes [Dijkstra 1978]
- Les philosophes buveurs [Chandy et Misra 1984]
- Lecteurs-Rédacteurs [Courtois, Heymans et Parnas 1971]
  - Au plus un rédacteur
  - Plusieurs lecteurs simultanément
  - Jamais de lectures et d'écritures simultanément

Algorithme basé sur les quorums

Problèmes fondamentaux liés à l'exclusion mutuelle

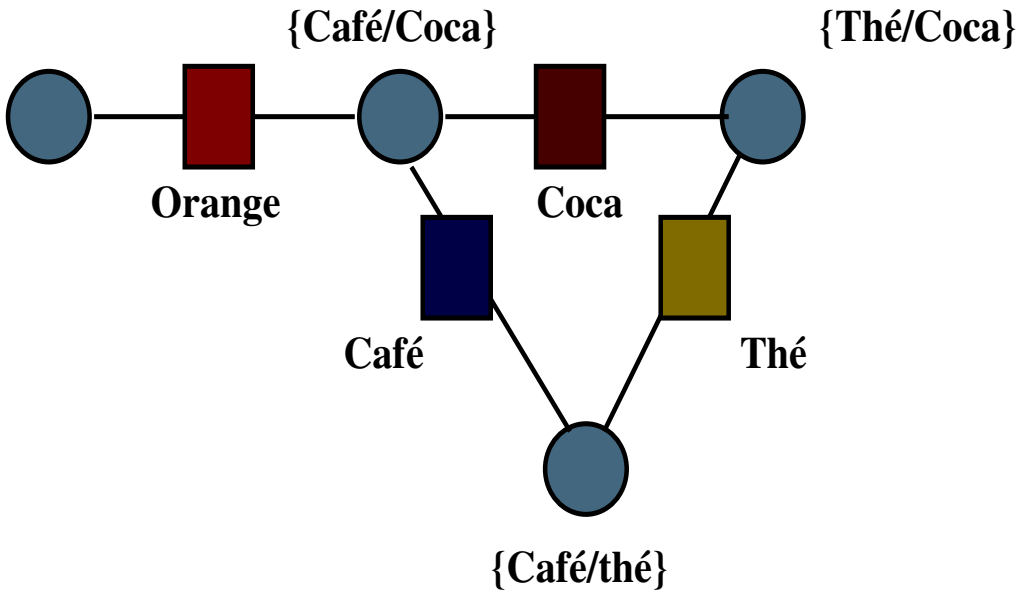
- Exclusion mutuelle classique [Dijkstra 1965]
  - Une ressource unique à partager entre N demandeurs potentiels
  - File d'attente
- Le dîner des philosophes [Dijkstra 1978]



Algorithme basé sur les quorums

Problèmes fondamentaux liés à l'exclusion mutuelle

- Exclusion mutuelle classique [Dijkstra 1965]
  - Une ressource unique à partager entre N demandeurs potentiels
  - File d'attente
- Le dîner des philosophes [Dijkstra 1978]
- Les philosophes buveurs [Chandy et Misra 1984]



Algorithme basé sur les quorums

Problèmes fondamentaux liés à l'exclusion mutuelle

- Exclusion mutuelle classique [Dijkstra 1965]
  - Une ressource unique à partager entre N demandeurs potentiels
  - File d'attente
- Le dîner des philosophes [Dijkstra 1978]
- Les philosophes buveurs [Chandy et Misra 1984]
- Lecteurs-Rédacteurs [Courtois, Heymans et Parnas 1971]
  - Au plus un rédacteur
  - Plusieurs lecteurs simultanément
  - Jamais de lectures et d'écritures simultanément

Algorithme basé sur les quorums

Problèmes fondamentaux liés à l'exclusion mutuelle

- Exclusion mutuelle classique [Dijkstra 1965]
  - Une ressource unique à partager entre N demandeurs potentiels
  - File d'attente
- Le dîner des philosophes [Dijkstra 1978]
- Les philosophes buveurs [Chandy et Misra 1984]
- Lecteurs-Rédacteurs [Courtois, Heymans et Parnas 1971]
  - Au plus un rédacteur
  - Plusieurs lecteurs simultanément
  - Jamais de lectures et d'écritures simultanément

Algorithme basé sur les quorums

Problèmes fondamentaux liés à l'exclusion mutuelle

- Exclusion mutuelle classique [Dijkstra 1965]
  - Une ressource unique à partager entre N demandeurs potentiels
  - File d'attente
- Le dîner des philosophes [Dijkstra 1978]
- Les philosophes buveurs [Chandy et Misra 1984]
- Lecteurs-Rédacteurs [Courtois, Heymans et Parnas 1971]
  - Au plus un rédacteur
  - Plusieurs lecteurs simultanément
  - Jamais de lectures et d'écritures simultanément

Algorithme basé sur les quorums

Problèmes fondamentaux liés à l'exclusion mutuelle

- Exclusion mutuelle classique [Dijkstra 1965]
  - Une ressource unique à partager entre N demandeurs potentiels
  - File d'attente
- Le dîner des philosophes [Dijkstra 1978]
- Les philosophes buveurs [Chandy et Misra 1984]
- Lecteurs-Rédacteurs [Courtois, Heymans et Parnas 1971]
  - Au plus un rédacteur
  - Plusieurs lecteurs simultanément
  - Jamais de lectures et d'écritures simultanément

Algorithme basé sur les quorums
Exclusion Mutuelle de Groupe

- Joung 1998
- Le problème
  - n processus et m sessions
  - Permettre à un nombre quelconque de processus d'accéder concurrement à une même session

Algorithme basé sur les quorums
Exclusion Mutuelle de Groupe

- Joung 1998
- Le problème
  - n processus et m sessions
  - Permettre à un nombre quelconque de processus d'accéder concurrement à une même session

Algorithme basé sur les quorums

Exclusion Mutuelle de Groupe

- Joung 1998
- Le problème
  - n processus et m sessions
  - Permettre à un nombre quelconque de processus d'accéder concurrement à une même session

Algorithme basé sur les quorums

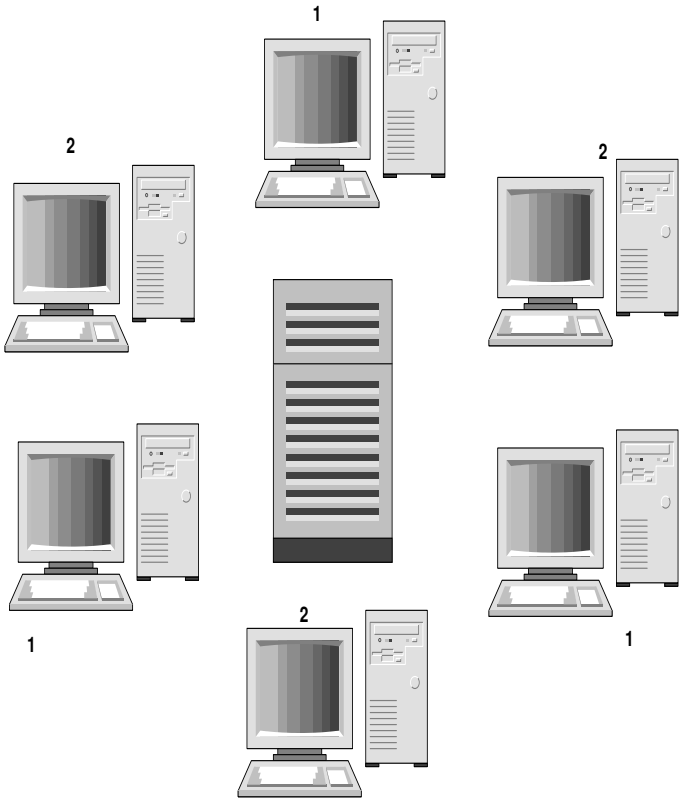
Exclusion Mutuelle de Groupe

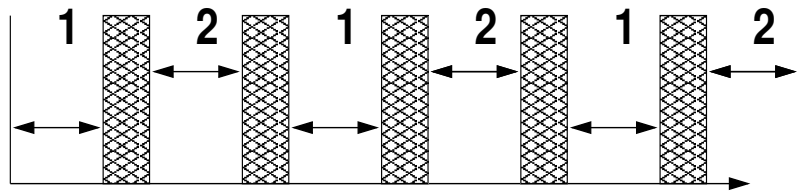
- Joung 1998
- Le problème
  - n processus et m sessions
  - Permettre à un nombre quelconque de processus d'accéder concurrement à une même session

Algorithme basé sur les quorums

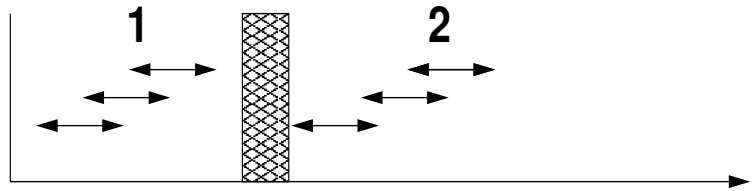
Exclusion Mutuelle de Groupe

- Un exemple, le juke-box





Exclusion mutuelle

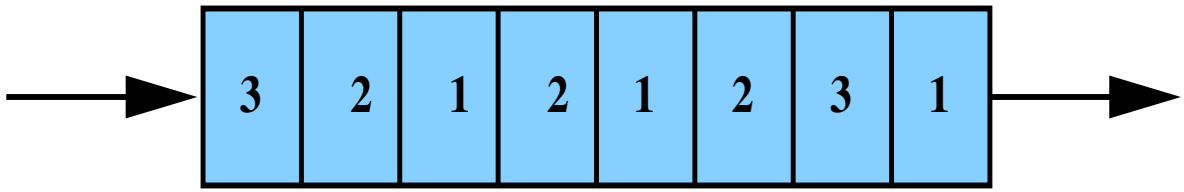


Exclusion mutuelle de groupe

Algorithme basé sur les quorums

Exclusion Mutuelle de Groupe

- Autre exemple, le serveur Web



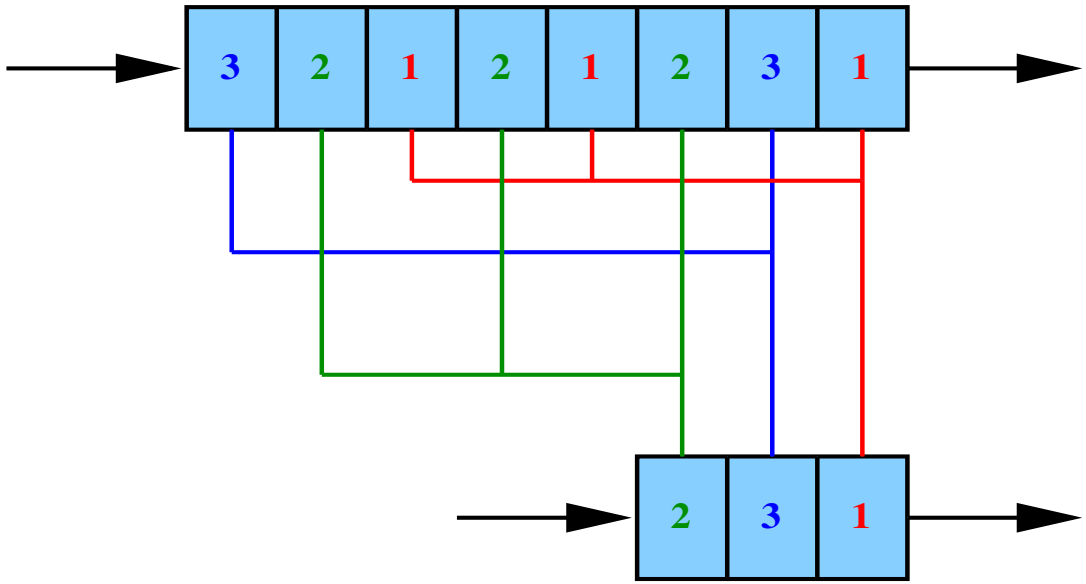
1– Demande pour la page d'accueil du LICP

2– Demande pour Google

3– Demande pour le programme télé

Algorithme basé sur les quorums
Exclusion Mutuelle de Groupe

- Autre exemple, le serveur Web



1– Demande pour la page d'accueil du LICP

2– Demande pour Google

3– Demande pour le programme télé

Algorithme basé sur les quorums

Exclusion Mutuelle de Groupe

- Les spécifications
  - Exclusion mutuelle (sûreté).
    Si deux processus distincts, p et q exécutent simultanément leur section critique respectivement dans les sessions S_p et S_q, alors $S_p = S_q$.
  - Absence de blocage (vivacité).
    Si un processus p veut participer à une session S_p, alors p exécute finalement sa section critique.
  - Entrée concurrente (efficacité).
    Si des processus veulent participer à une session, et qu'aucun autre processus ne veut participer à une autre session différente, alors les processus peuvent y entrer concurremment.

Algorithme basé sur les quorums

Exclusion Mutuelle de Groupe

- Les spécifications
  - Exclusion mutuelle (sûreté).
    Si deux processus distincts, p et q exécutent simultanément leur section critique respectivement dans les sessions S_p et S_q, alors $S_p = S_q$.
  - Absence de blocage (vivacité).
    Si un processus p veut participer à une session S_p, alors p exécute finalement sa section critique.
  - Entrée concurrente (efficacité).
    Si des processus veulent participer à une session, et qu'aucun autre processus ne veut participer à une autre session différente, alors les processus peuvent y entrer concurremment.

Algorithme basé sur les quorums

Exclusion Mutuelle de Groupe

- Les spécifications
  - Exclusion mutuelle (sûreté).
    Si deux processus distincts, p et q exécutent simultanément leur section critique respectivement dans les sessions S_p et S_q, alors $S_p = S_q$.
  - Absence de blocage (vivacité).
    Si un processus p veut participer à une session S_p, alors p exécute finalement sa section critique.
  - Entrée concurrente (efficacité).
    Si des processus veulent participer à une session, et qu'aucun autre processus ne veut participer à une autre session différente, alors les processus peuvent y entrer concurremment.

Algorithme basé sur les quorums

Exclusion Mutuelle de Groupe

- Les spécifications
  - Exclusion mutuelle (sûreté).
    Si deux processus distincts, p et q exécutent simultanément leur section critique respectivement dans les sessions S_p et S_q, alors $S_p = S_q$.
  - Absence de blocage (vivacité).
    Si un processus p veut participer à une session S_p, alors p exécute finalement sa section critique.
  - Entrée concurrente (efficacité).
    Si des processus veulent participer à une session, et qu'aucun autre processus ne veut participer à une autre session différente, alors les processus peuvent y entrer concurremment.

Algorithme basé sur les quorums

- Travaux existants
  - Modèle à mémoire partagée
    - *Joung* 98]
    - [*Keane & Moir* 99]
  - Modèle à passage de messages
    - [*Joung* 99] réseau complet
    - [*Wu & Joung* 99] anneau unidirectionnel
  - 3 algorithmes GMERn, GMERm, GMERnm fondés sur une circulation de jeton dans un anneau unidirectionnel
    - [*Cantarell , Datta , Petit & Villain* 2001]
  - 3 algorithmes GMET1, GMET2, GMET3 dans un arbre
    - [*Beauquier , Cantarell , Datta & Petit* 2003]

Algorithme basé sur les quorums

- Travaux existants
  - Modèle à mémoire partagée
    - *Joung* 98]
    - [*Keane & Moir* 99]
  - Modèle à passage de messages
    - [*Joung* 99] réseau complet
    - [*Wu & Joung* 99] anneau unidirectionnel
  - 3 algorithmes GMERn, GMERm, GMERnm fondés sur une circulation de jeton dans un anneau unidirectionnel
    - [*Cantarell , Datta , Petit & Villain* 2001]
  - 3 algorithmes GMET1, GMET2, GMET3 dans un arbre
    - [*Beauquier , Cantarell , Datta & Petit* 2003]

Algorithme basé sur les quorums

- Travaux existants
  - Modèle à mémoire partagée
    - *Joung* 98]
    - [*Keane & Moir* 99]
  - Modèle à passage de messages
    - [*Joung* 99] réseau complet
    - [*Wu & Joung* 99] anneau unidirectionnel
  - 3 algorithmes GMERn, GMERm, GMERnm fondés sur une circulation de jeton dans un anneau unidirectionnel
    - [*Cantarell , Datta , Petit & Villain* 2001]
  - 3 algorithmes GMET1, GMET2, GMET3 dans un arbre
    - [*Beauquier , Cantarell , Datta & Petit* 2003]

Algorithme basé sur les quorums

- Travaux existants
  - Modèle à mémoire partagée
    - *Joung* 98]
    - [*Keane & Moir* 99]
  - Modèle à passage de messages
    - [*Joung* 99] réseau complet
    - [*Wu & Joung* 99] anneau unidirectionnel
  - 3 algorithmes GMERn, GMERm, GMERnm fondés sur une circulation de jeton dans un anneau unidirectionnel
    - [*Cantarell , Datta , Petit & Villain* 2001]
  - 3 algorithmes GMET1, GMET2, GMET3 dans un arbre
    - [*Beauquier , Cantarell , Datta & Petit* 2003]

Algorithme basé sur les quorums

- Travaux existants
  - Modèle à mémoire partagée
    - *Joung* 98]
    - [*Keane & Moir* 99]
  - Modèle à passage de messages
    - [*Joung* 99] réseau complet
    - [*Wu & Joung* 99] anneau unidirectionnel
  - 3 algorithmes GMERn, GMERm, GMERnm fondés sur une circulation de jeton dans un anneau unidirectionnel
    - [*Cantarell , Datta , Petit & Villain* 2001]
  - 3 algorithmes GMET1, GMET2, GMET3 dans un arbre
    - [*Beauquier , Cantarell , Datta & Petit* 2003]

Algorithme basé sur les quorums

- Travaux existants
  - Modèle à mémoire partagée
    - *Joung* 98]
    - [*Keane & Moir* 99]
  - Modèle à passage de messages
    - [*Joung* 99] réseau complet
    - [*Wu & Joung* 99] anneau unidirectionnel
  - 3 algorithmes GMERn, GMERm, GMERnm fondés sur une circulation de jeton dans un anneau unidirectionnel
    - [*Cantarell , Datta , Petit & Villain* 2001]
  - 3 algorithmes GMET1, GMET2, GMET3 dans un arbre
    - [*Beauquier , Cantarell , Datta & Petit* 2003]

Algorithme basé sur les quorums

- Travaux existants
  - Modèle à mémoire partagée
    - *Joung* 98]
    - [*Keane & Moir* 99]
  - Modèle à passage de messages
    - [*Joung* 99] réseau complet
    - [*Wu & Joung* 99] anneau unidirectionnel
  - 3 algorithmes GMERn, GMERm, GMERnm fondés sur une circulation de jeton dans un anneau unidirectionnel
    - [*Cantarell , Datta , Petit & Villain* 2001]
  - 3 algorithmes GMET1, GMET2, GMET3 dans un arbre
    - [*Beauquier , Cantarell , Datta & Petit* 2003]

Algorithme basé sur les quorums

- Travaux existants
  - Modèle à mémoire partagée
    - *Joung* 98]
    - [*Keane & Moir* 99]
  - Modèle à passage de messages
    - [*Joung* 99] réseau complet
    - [*Wu & Joung* 99] anneau unidirectionnel
  - 3 algorithmes GMERn, GMERm, GMERnm fondés sur une circulation de jeton dans un anneau unidirectionnel
    - [*Cantarell , Datta , Petit & Villain* 2001]
  - 3 algorithmes GMET1, GMET2, GMET3 dans un arbre
    - [*Beauquier , Cantarell , Datta & Petit* 2003]

Algorithme basé sur les quorums

- Travaux existants
  - Modèle à mémoire partagée
    - *Joung* 98]
    - [*Keane & Moir* 99]
  - Modèle à passage de messages
    - [*Joung* 99] réseau complet
    - [*Wu & Joung* 99] anneau unidirectionnel
  - 3 algorithmes GMERn, GMERm, GMERnm fondés sur une circulation de jeton dans un anneau unidirectionnel
    - [*Cantarell , Datta , Petit & Villain* 2001]
  - 3 algorithmes GMET1, GMET2, GMET3 dans un arbre
    - [*Beauquier , Cantarell , Datta & Petit* 2003]

Algorithme basé sur les quorums

- Travaux existants
  - Modèle à mémoire partagée
    - *Joung* 98]
    - [*Keane & Moir* 99]
  - Modèle à passage de messages
    - [*Joung* 99] réseau complet
    - [*Wu & Joung* 99] anneau unidirectionnel
  - 3 algorithmes GMERn, GMERm, GMERnm fondés sur une circulation de jeton dans un anneau unidirectionnel
    - [*Cantarell , Datta , Petit & Villain* 2001]
  - 3 algorithmes GMET1, GMET2, GMET3 dans un arbre
    - [*Beauquier , Cantarell , Datta & Petit* 2003]

Algorithme basé sur les quorums

- Travaux existants
  - Modèle à mémoire partagée
    - *Joung* 98]
    - [*Keane & Moir* 99]
  - Modèle à passage de messages
    - [*Joung* 99] réseau complet
    - [*Wu & Joung* 99] anneau unidirectionnel
  - 3 algorithmes GMERn, GMERm, GMERnm fondés sur une circulation de jeton dans un anneau unidirectionnel
    - [*Cantarell* , *Datta* , *Petit & Villain* 2001]
  - 3 algorithmes GMET1, GMET2, GMET3 dans un arbre
    - [*Beauquier* , *Cantarell* , *Datta & Petit* 2003]

Algorithme basé sur les quorums

Exemple

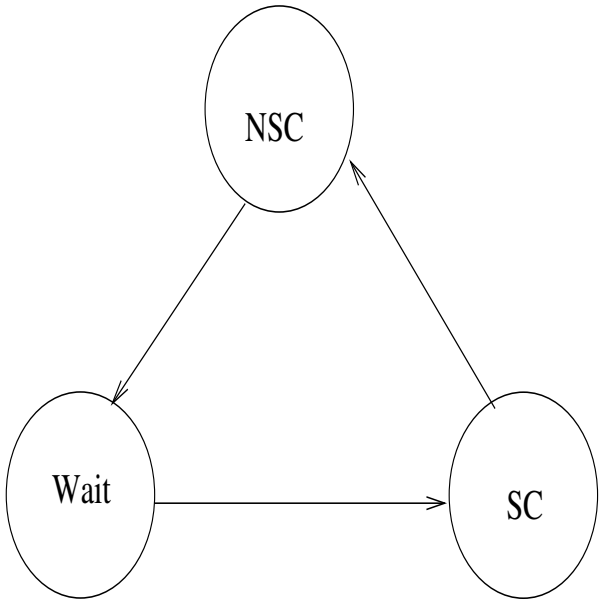


Figure: Etat des processus

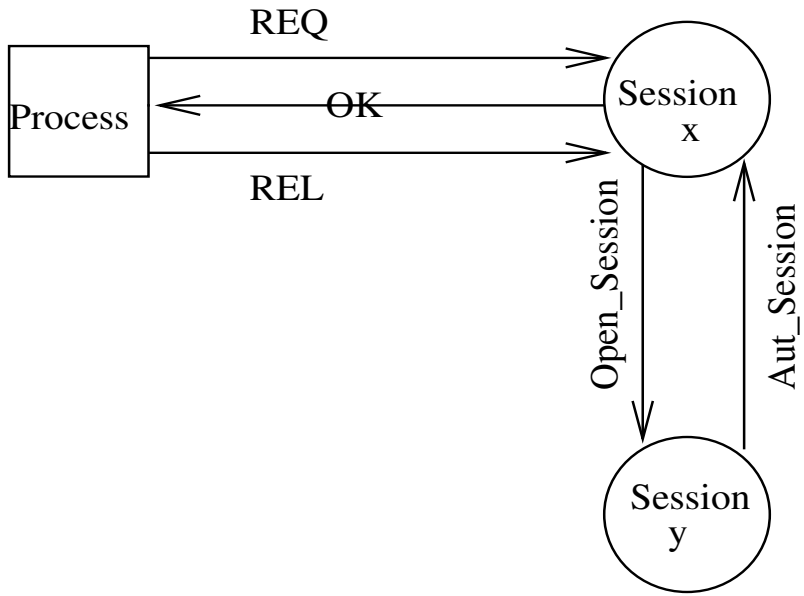


Figure: Messages processus/sessions et entre les

Algorithme basé sur les quorums

Principe de l'algorithme

Quand un processus $P_i \in Q$ veut participer à une session x :

- Envoie d'un message REQ() à la session x.
- x envoie un message *Open_Session*() à son groupe.
- Si x reçoit toutes les permissions de son groupe c'est-à-dire $|g(x)| - 1$ messages *Aut_Session*(), elle envoie un message OK() au processus P_i.
- P_i peut maintenant ouvrir la session x.
- Lors de la sortie de la section critique, P_i envoie un message REL() à x.

Algorithme basé sur les quorums

Principe de l'algorithme

Quand un processus $P_i \in Q$ veut participer à une session x :

- Envoie d'un message REQ() à la session x.
- x envoie un message *Open_Session*() à son groupe.
- Si x reçoit toutes les permissions de son groupe c'est-à-dire $|g(x)| - 1$ messages *Aut_Session*(), elle envoie un message OK() au processus P_i.
- P_i peut maintenant ouvrir la session x.
- Lors de la sortie de la section critique, P_i envoie un message REL() à x.

Algorithme basé sur les quorums

Principe de l'algorithme

Quand un processus $P_i \in Q$ veut participer à une session x :

- Envoie d'un message REQ() à la session x.
- x envoie un message *Open_Session*() à son groupe.
- Si x reçoit toutes les permissions de son groupe c'est-à-dire $|g(x)| - 1$ messages *Aut_Session*(), elle envoie un message OK() au processus P_i.
- P_i peut maintenant ouvrir la session x.
- Lors de la sortie de la section critique, P_i envoie un message REL() à x.

Algorithme basé sur les quorums

Principe de l'algorithme

Quand un processus $P_i \in Q$ veut participer à une session x :

- Envoie d'un message REQ() à la session x.
- x envoie un message *Open_Session*() à son groupe.
- Si x reçoit toutes les permissions de son groupe c'est-à-dire $|g(x)| - 1$ messages *Aut_Session*(), elle envoie un message OK() au processus P_i.
- P_i peut maintenant ouvrir la session x.
- Lors de la sortie de la section critique, P_i envoie un message REL() à x.

Algorithme basé sur les quorums

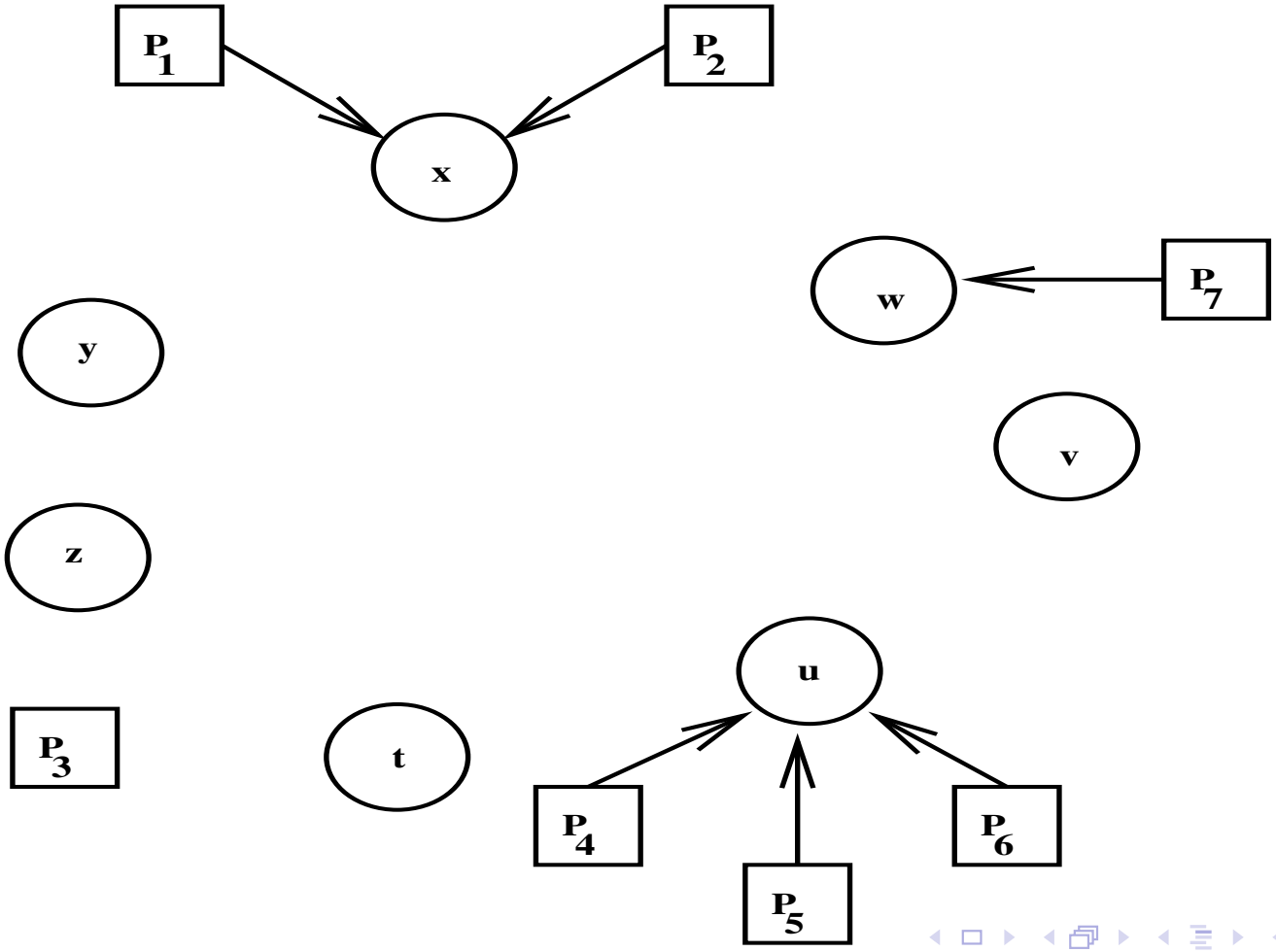
Principe de l'algorithme

Quand un processus $P_i \in Q$ veut participer à une session x :

- Envoie d'un message REQ() à la session x.
- x envoie un message *Open_Session*() à son groupe.
- Si x reçoit toutes les permissions de son groupe c'est-à-dire $|g(x)| - 1$ messages *Aut_Session*(), elle envoie un message OK() au processus P_i.
- P_i peut maintenant ouvrir la session x.
- Lors de la sortie de la section critique, P_i envoie un message REL() à x.

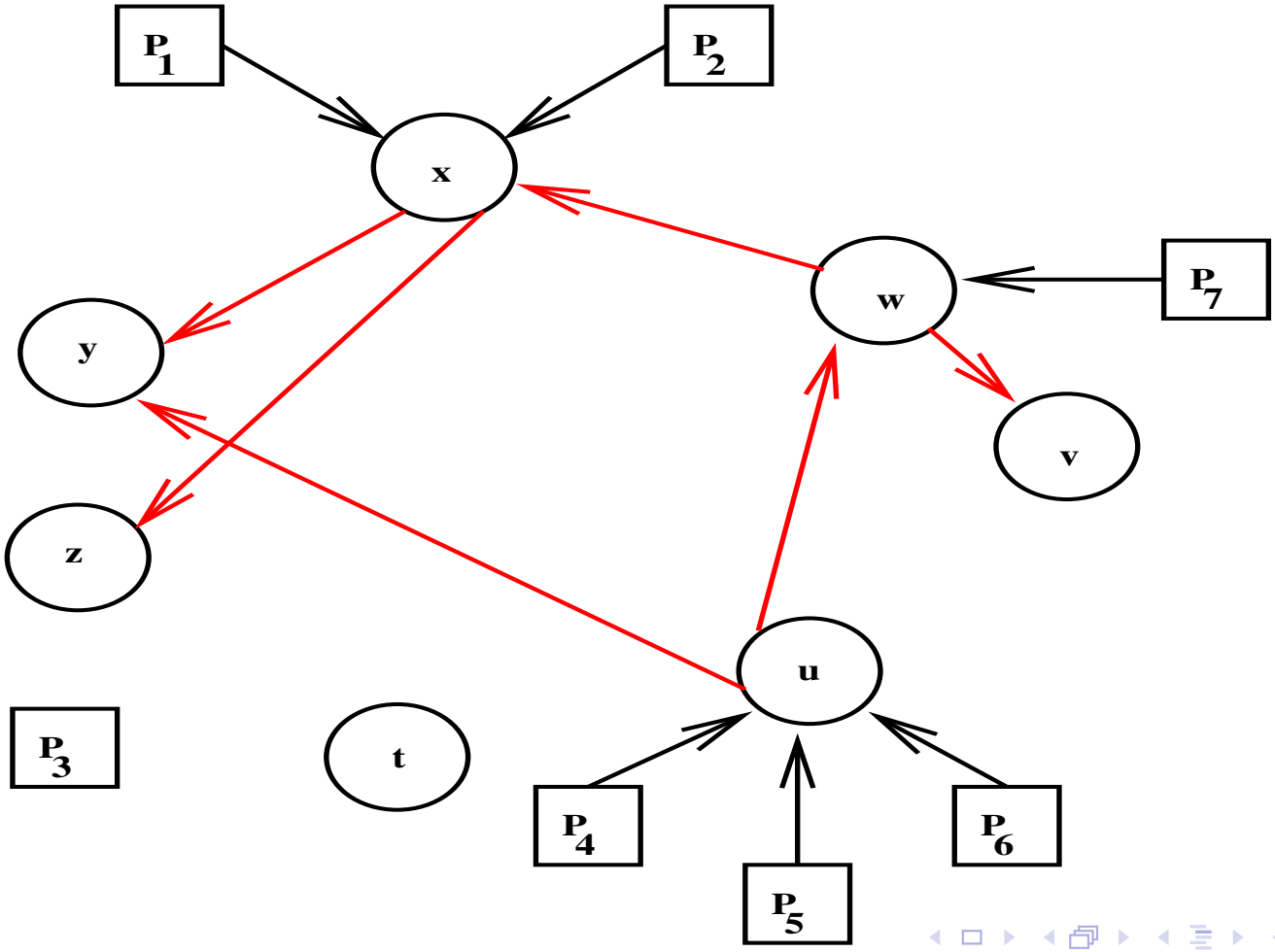
Algorithme basé sur les quorums

Exemple



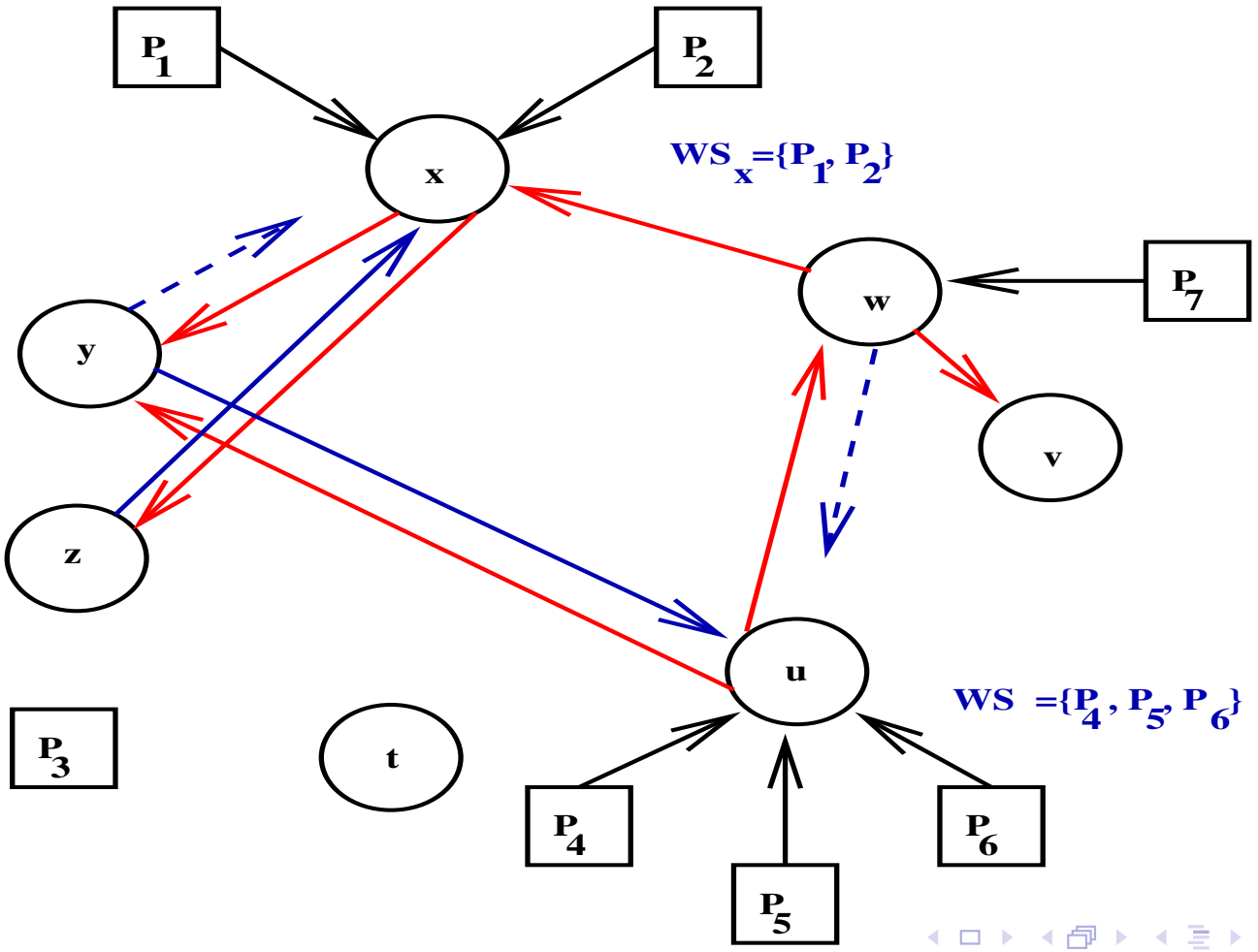
Algorithme basé sur les quorums

Exemple



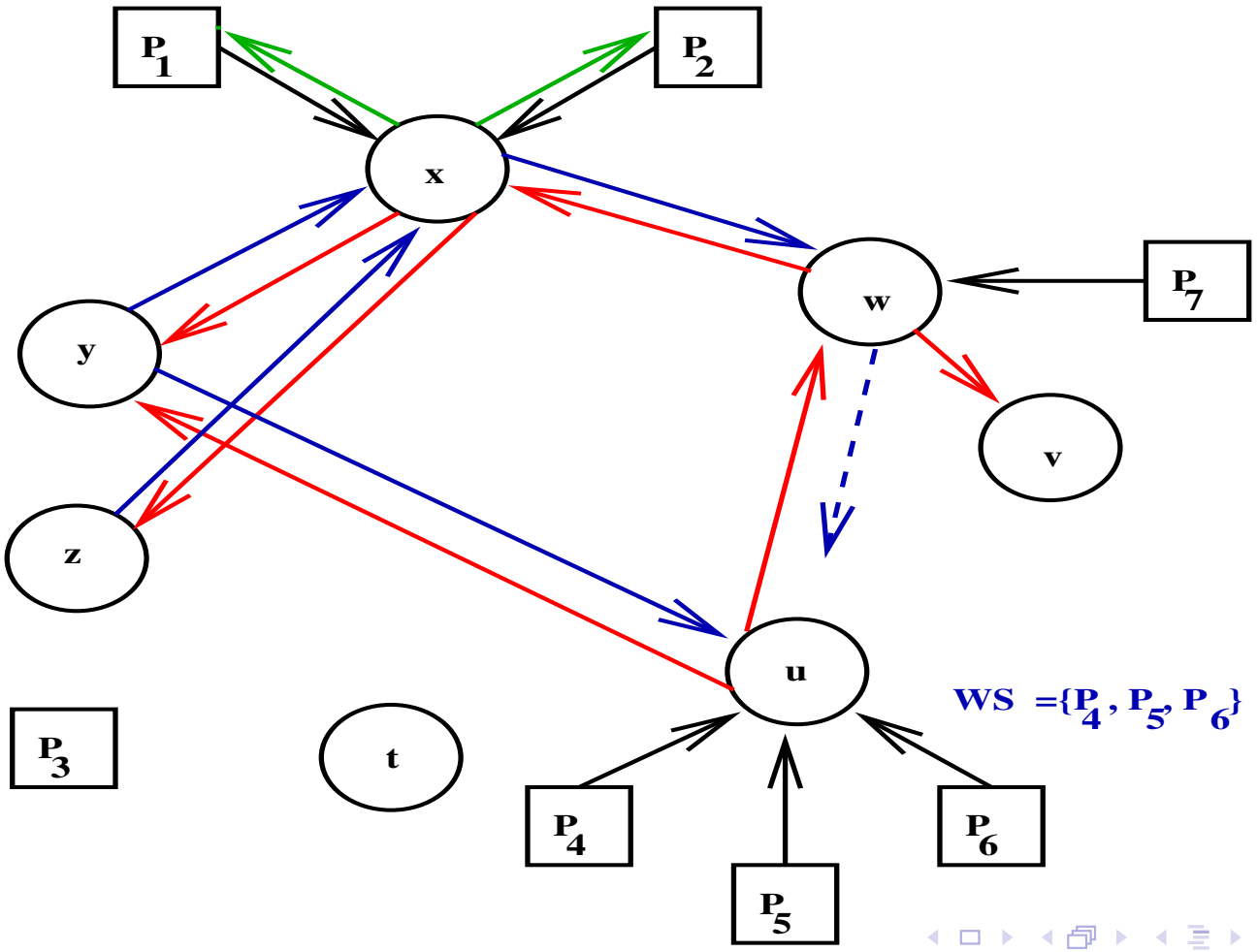
Algorithme basé sur les quorums

Exemple



Algorithme basé sur les quorums

Exemple



Algorithme basé sur les quorums

Preuve

Théorème (1)

Cet algorithme garantit l'exclusion mutuelle.

Si deux processus P_i et P_j sont en section critique, deux cas possibles :

- P_i et P_j sont dans la même session x_i
- P_i et P_j sont dans deux sessions différentes x_i et x_j $(i \neq j)$.

Algorithme basé sur les quorums

Preuve

Théorème (1)

Cet algorithme garantit l'exclusion mutuelle.

Si deux processus P_i et P_j sont en section critique, deux cas possibles :

- P_i et P_j sont dans la même session x_i
- P_i et P_j sont dans deux sessions différentes x_i et x_j $(i \neq j)$.

Algorithme basé sur les quorums

Preuve

Théorème (2)

L'algorithme est exempt de famine.

Supposer que la famine existe pour une session x. Un message Aut_Session() sera envoyé à x quand sa requête aura la plus haute priorité. Ceci a lieu après $|g(x)| - 1$ requêtes sont traitées car toute requête suivante sera de plus haute priorité que x. Donc, au bout d'un temps fini, x recevra Aut_Session() et sera ouverte.

Algorithme basé sur les quorums

Preuve

Théorème (3)

L'algorithme est exempt de blocage.

Il n'y a pas d'attente circulaire ([Maekawa 85]).

Algorithme basé sur les quorums

Complexité

$O(n + |Q|)$ messages par entrée en section critique et le nombre de messages requis entre un processus et une session est de 3 pour chaque entrée en section critique.

3- Exclusion mutuelle de groupe basée sur les accès concurrents

Algorithme basé sur les accès concurrents

Exemple

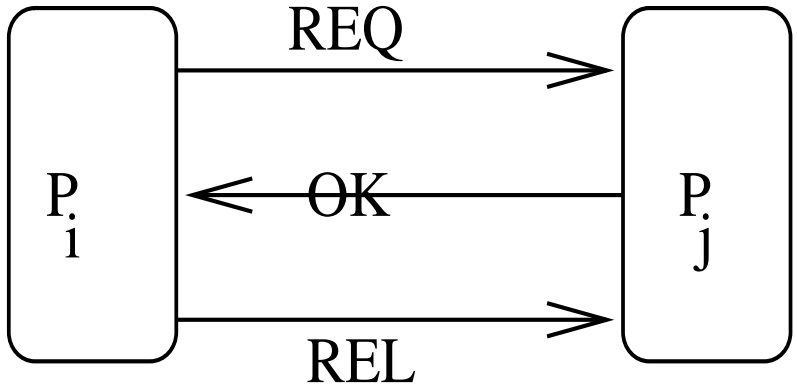


Figure: Messages échangés entre les processus

Algorithme basé sur les accès concurrents

Principe de l'algorithme

Un processus $P_i \in g(P_i)$ qui veut participer à une session x

- envoie un message REQ() à tous les membres de son groupe et attend des messages OK().
- Si le nombre de messages OK() reçus est $|g(P_i)| - 1$ alors le processus P_i envoie un message $\Theta()$ à son groupe.
- Ainsi P_i peut participer à la session x avec les membres de son groupe.

Algorithme basé sur les accès concurrents

Principe de l'algorithme

Un processus $P_i \in g(P_i)$ qui veut participer à une session x

- envoie un message REQ() à tous les membres de son groupe et attend des messages OK().
- Si le nombre de messages OK() reçus est $|g(P_i)| - 1$ alors le processus P_i envoie un message $\Theta()$ à son groupe.
- Ainsi P_i peut participer à la session x avec les membres de son groupe.

Algorithme basé sur les accès concurrents

Principe de l'algorithme

Un processus $P_i \in g(P_i)$ qui veut participer à une session x

- envoie un message REQ() à tous les membres de son groupe et attend des messages OK().
- Si le nombre de messages OK() reçus est $|g(P_i)| - 1$ alors le processus P_i envoie un message $\Theta()$ à son groupe.
- Ainsi P_i peut participer à la session x avec les membres de son groupe.

Algorithme basé sur les accès concurrents

Exemple

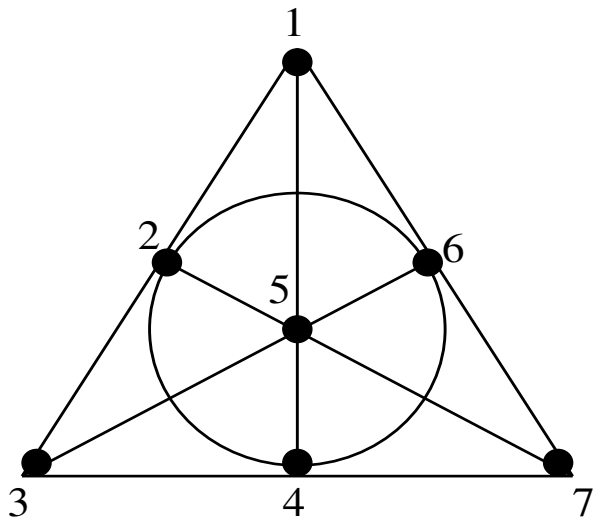


Figure: Plan projectif fini d'ordre 2 (Fano plane)

$g(P_1) = \{1, 2, 3\}$, $g(P_2) = \{2, 4, 6\}$, $g(P_3) = \{3, 5, 6\}$
$g(P_4) = \{4, 1, 5\}$, $g(P_5) = \{5, 2, 7\}$, $g(P_6) = \{6, 1, 7\}$
$g(P_7) = \{7, 4, 3\}$. Soient deux sessions x_1 et x_2.

Algorithme basé sur les accès concurrents

exemple

- Les processus 1, 2 et 3 $\in g(P_1)$.
- 1 envoie un message $REQ(1, x_1, H_1)$ aux processus 2 et 3.
- 1 attend un message $OK()$ des membres de son groupe, c'est-à-dire 2 et 3.
- Les processus 2 et 3 reçoivent un message $REQ(P_i, x, H_i)$ du processus 1.
- Si les processus 2 et 3 n'ont pas demandé à participer à une autre session x_2, ils peuvent envoyer un message $OK()$ au processus 1.
- Le processus 1 a reçu tous les messages $OK()$ $(nOK = |g(P_i)| - 1)$ des membres de son groupe.
- Le processus 1 envoie un message $\Theta(x_1)$ aux processus 2 et 3 afin d'accéder à la session x_1.

Algorithme basé sur les accès concurrents

exemple

- Les processus 1, 2 et 3 $\in g(P_1)$.
- 1 envoie un message $REQ(1, x_1, H_1)$ aux processus 2 et 3.
- 1 attend un message $OK()$ des membres de son groupe, c'est-à-dire 2 et 3.
- Les processus 2 et 3 reçoivent un message $REQ(P_i, x, H_i)$ du processus 1.
- Si les processus 2 et 3 n'ont pas demandé à participer à une autre session x_2, ils peuvent envoyer un message $OK()$ au processus 1.
- Le processus 1 a reçu tous les messages $OK()$ $(nOK = |g(P_i)| - 1)$ des membres de son groupe.
- Le processus 1 envoie un message $\Theta(x_1)$ aux processus 2 et 3 afin d'accéder à la session x_1.

Algorithme basé sur les accès concurrents

exemple

- Les processus 1, 2 et 3 $\in g(P_1)$.
- 1 envoie un message $REQ(1, x_1, H_1)$ aux processus 2 et 3.
- 1 attend un message $OK()$ des membres de son groupe, c'est-à-dire 2 et 3.
- Les processus 2 et 3 reçoivent un message $REQ(P_i, x, H_i)$ du processus 1.
- Si les processus 2 et 3 n'ont pas demandé à participer à une autre session x_2, ils peuvent envoyer un message $OK()$ au processus 1.
- Le processus 1 a reçu tous les messages $OK()$ $(nOK = |g(P_i)| - 1)$ des membres de son groupe.
- Le processus 1 envoie un message $\Theta(x_1)$ aux processus 2 et 3 afin d'accéder à la session x_1.

Algorithme basé sur les accès concurrents

exemple

- Les processus 1, 2 et 3 $\in g(P_1)$.
- 1 envoie un message $REQ(1, x_1, H_1)$ aux processus 2 et 3.
- 1 attend un message $OK()$ des membres de son groupe, c'est-à-dire 2 et 3.
- Les processus 2 et 3 reçoivent un message $REQ(P_i, x, H_i)$ du processus 1.
- Si les processus 2 et 3 n'ont pas demandé à participer à une autre session x_2, ils peuvent envoyer un message $OK()$ au processus 1.
- Le processus 1 a reçu tous les messages $OK()$ ($nOK = |g(P_i)| - 1$) des membres de son groupe.
- Le processus 1 envoie un message $\Theta(x_1)$ aux processus 2 et 3 afin d'accéder à la session x_1.

Algorithme basé sur les accès concurrents

exemple

- Les processus 1, 2 et 3 $\in g(P_1)$.
- 1 envoie un message $REQ(1, x_1, H_1)$ aux processus 2 et 3.
- 1 attend un message $OK()$ des membres de son groupe, c'est-à-dire 2 et 3.
- Les processus 2 et 3 reçoivent un message $REQ(P_i, x, H_i)$ du processus 1.
- Si les processus 2 et 3 n'ont pas demandé à participer à une autre session x_2, ils peuvent envoyer un message $OK()$ au processus 1.
- Le processus 1 a reçu tous les messages $OK()$ ($nOK = |g(P_i)| - 1$) des membres de son groupe.
- Le processus 1 envoie un message $\Theta(x_1)$ aux processus 2 et 3 afin d'accéder à la session x_1.

Algorithme basé sur les accès concurrents
exemple

- Les processus 1, 2 et 3 $\in g(P_1)$.
- 1 envoie un message $REQ(1, x_1, H_1)$ aux processus 2 et 3.
- 1 attend un message $OK()$ des membres de son groupe, c'est-à-dire 2 et 3.
- Les processus 2 et 3 reçoivent un message $REQ(P_i, x, H_i)$ du processus 1.
- Si les processus 2 et 3 n'ont pas demandé à participer à une autre session x_2, ils peuvent envoyer un message $OK()$ au processus 1.
- Le processus 1 a reçu tous les messages $OK()$ ($nOK = |g(P_i)| - 1$) des membres de son groupe.
- Le processus 1 envoie un message $\Theta(x_1)$ aux processus 2 et 3 afin d'accéder à la session x_1.

Algorithme basé sur les accès concurrents

exemple

- Les processus 1, 2 et 3 $\in g(P_1)$.
- 1 envoie un message $REQ(1, x_1, H_1)$ aux processus 2 et 3.
- 1 attend un message $OK()$ des membres de son groupe, c'est-à-dire 2 et 3.
- Les processus 2 et 3 reçoivent un message $REQ(P_i, x, H_i)$ du processus 1.
- Si les processus 2 et 3 n'ont pas demandé à participer à une autre session x_2, ils peuvent envoyer un message $OK()$ au processus 1.
- Le processus 1 a reçu tous les messages $OK()$ $(nOK = |g(P_i)| - 1)$ des membres de son groupe.
- Le processus 1 envoie un message $\Theta(x_1)$ aux processus 2 et 3 afin d'accéder à la session x_1.

Algorithme basé sur les accès concurrents

Preuve

Lemme (1)

Au plus une seule session peut être ouverte à la fois.

P_1 et P_2 exécutent leurs sections critiques respectives dans les sessions x_1 et x_2. Soit $g(P_1) \cap g(P_2) = \{P_k\}$. Puisque P_k demande une seule session à la fois, alors il n'enverra pas de OK() simultanément à P_1 et P_2. Ce qui est absurde.

Algorithme basé sur les accès concurrents

Preuve

Lemme (2)

l'algorithme est exempt de famine.

Preuve par l'absurde

Soit $P_i \in g(P_i)$ pour x_i ayant la plus haute priorité avec H_i. P_i recevra OK() si aucun processus n'est en SC et aucun autre processus (excepté P_i) n'a pas fait de demande. Considérons le cas où quelques processus pour x_i' entrent et sortent de SC indéfiniment. Puisque chaque processus envoie OK() à tout processus dont la priorité est supérieure à H_i, il est impossible que d'autres processus entrent et sortent indéfiniment de la SC.

Algorithme basé sur les accès concurrents

Preuve

Théorème (1)

L'algorithme proposé résout le problème de l'exclusion mutuelle de groupe.

Lemmes (1), (2) et (3).

Complexité

$O(\sqrt{n})$ pour le plan projectif fini et $O(2\sqrt{n}-1)$ pour une grille.

4- Exclusion mutuelle de groupe basée sur le modèle client-serveur

Algorithme basé sur le modèle client-serveur

Exemple

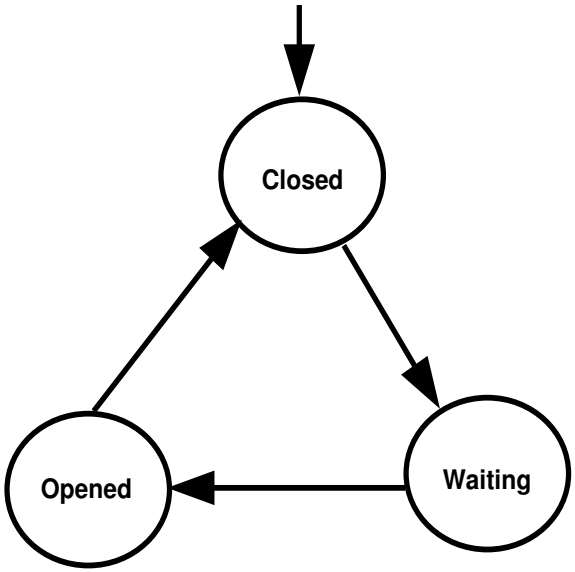


Figure: Etat des sessions

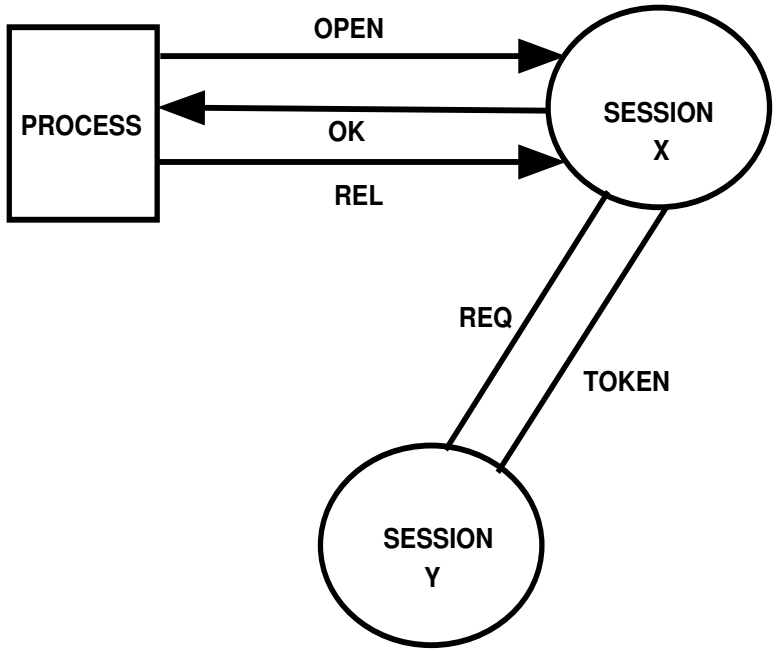


Figure: Messages

Algorithme basé sur le modèle client-serveur

Principe de l'algorithme

Initialement la session racine possède le jeton. Chaque processus P_i se comporte comme un client. Quand une session x reçoit un message OPEN() d'un processus P_i, plusieurs cas sont possibles :

- x a le jeton et il n'existe pas de session en attente, alors x envoie OK() à P_i.
- Sinon s'il existe une session en attente, P_i est ajoutée à la file d'attente.
- Si x reçoit OPEN() de P_i et ne possède pas le jeton
  - x envoie REQ() au leader pour avoir le jeton
  - si y reçoit REQ() de x, elle envoie TOKEN() à x et x envoie OK() à tous les processus de sa file d'attente et attend les messages REL().
  - si tous les messages REL() sont reçus, x envoie le jeton à la prochaine session.

Algorithme basé sur le modèle client-serveur

Principe de l'algorithme

Initialement la session racine possède le jeton. Chaque processus P_i se comporte comme un client. Quand une session x reçoit un message OPEN() d'un processus P_i, plusieurs cas sont possibles :

- x a le jeton et il n'existe pas de session en attente, alors x envoie OK() à P_i.
- Sinon s'il existe une session en attente, P_i est ajoutée à la file d'attente.
- Si x reçoit OPEN() de P_i et ne possède pas le jeton
  - x envoie REQ() au leader pour avoir le jeton
  - si y reçoit REQ() de x, elle envoie TOKEN() à x et x envoie OK() à tous les processus de sa file d'attente et attend les messages REL().
  - si tous les messages REL() sont reçus, x envoie le jeton à la prochaine session.

Algorithme basé sur le modèle client-serveur

Principe de l'algorithme

Initialement la session racine possède le jeton. Chaque processus P_i se comporte comme un client. Quand une session x reçoit un message OPEN() d'un processus P_i, plusieurs cas sont possibles :

- x a le jeton et il n'existe pas de session en attente, alors x envoie OK() à P_i.
- Sinon s'il existe une session en attente, P_i est ajoutée à la file d'attente.
- Si x reçoit OPEN() de P_i et ne possède pas le jeton
  - x envoie REQ() au leader pour avoir le jeton
  - si y reçoit REQ() de x, elle envoie TOKEN() à x et x envoie OK() à tous les processus de sa file d'attente et attend les messages REL().
  - si tous les messages REL() sont reçus, x envoie le jeton à la prochaine session.

Algorithme basé sur le modèle client-serveur

Principe de l'algorithme

Initialement la session racine possède le jeton. Chaque processus P_i se comporte comme un client. Quand une session x reçoit un message OPEN() d'un processus P_i, plusieurs cas sont possibles :

- x a le jeton et il n'existe pas de session en attente, alors x envoie OK() à P_i.
- Sinon s'il existe une session en attente, P_i est ajoutée à la file d'attente.
- Si x reçoit OPEN() de P_i et ne possède pas le jeton
  - x envoie REQ() au leader pour avoir le jeton
  - si y reçoit REQ() de x, elle envoie TOKEN() à x et x envoie OK() à tous les processus de sa file d'attente et attend les messages REL().
  - si tous les messages REL() sont reçus, x envoie le jeton à la prochaine session.

Algorithme basé sur le modèle client-serveur

Principe de l'algorithme

Initialement la session racine possède le jeton. Chaque processus P_i se comporte comme un client. Quand une session x reçoit un message OPEN() d'un processus P_i, plusieurs cas sont possibles :

- x a le jeton et il n'existe pas de session en attente, alors x envoie OK() à P_i.
- Sinon s'il existe une session en attente, P_i est ajoutée à la file d'attente.
- Si x reçoit OPEN() de P_i et ne possède pas le jeton
  - x envoie REQ() au leader pour avoir le jeton
  - si y reçoit REQ() de x, elle envoie TOKEN() à x et x envoie OK() à tous les processus de sa file d'attente et attend les messages REL().
  - si tous les messages REL() sont reçus, x envoie le jeton à la prochaine session.

Algorithme basé sur le modèle client-serveur

Principe de l'algorithme

Initialement la session racine possède le jeton. Chaque processus P_i se comporte comme un client. Quand une session x reçoit un message OPEN() d'un processus P_i, plusieurs cas sont possibles :

- x a le jeton et il n'existe pas de session en attente, alors x envoie OK() à P_i.
- Sinon s'il existe une session en attente, P_i est ajoutée à la file d'attente.
- Si x reçoit OPEN() de P_i et ne possède pas le jeton
  - x envoie REQ() au leader pour avoir le jeton
  - si y reçoit REQ() de x, elle envoie TOKEN() à x et x envoie OK() à tous les processus de sa file d'attente et attend les messages REL().
  - si tous les messages REL() sont reçus, x envoie le jeton à la prochaine session.

Algorithme basé sur le modèle client-serveur

Exemple

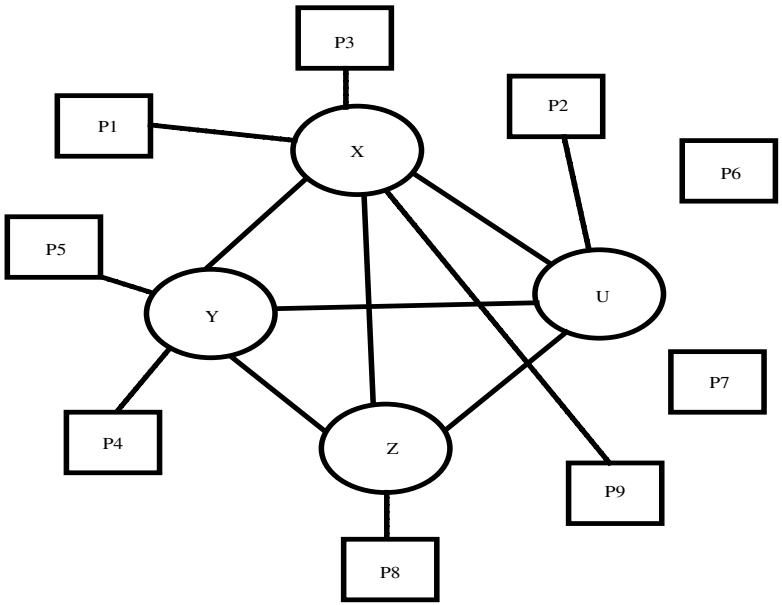


Figure: Graphe processus/sessions

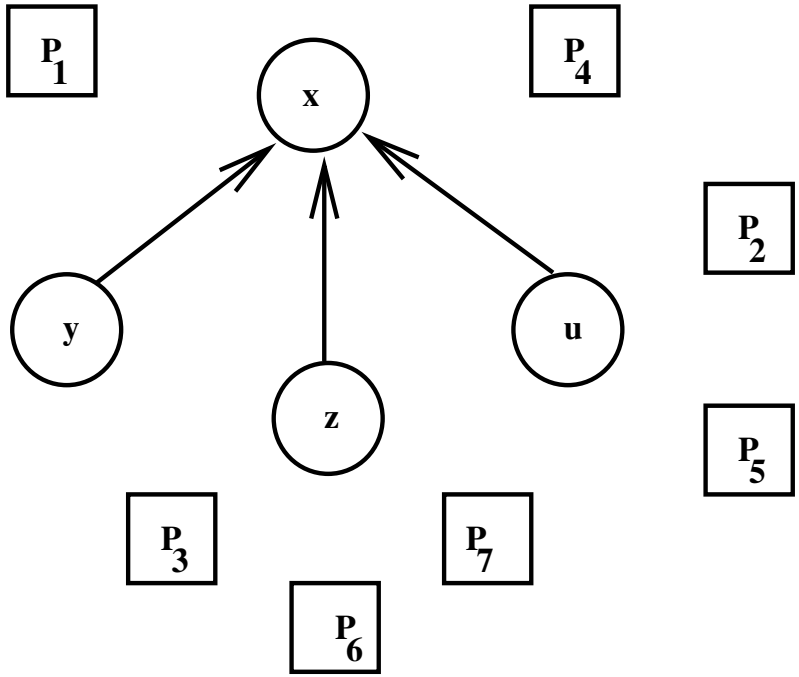


Figure: Arbre enraciné initial

Algorithme basé sur le modèle client-serveur

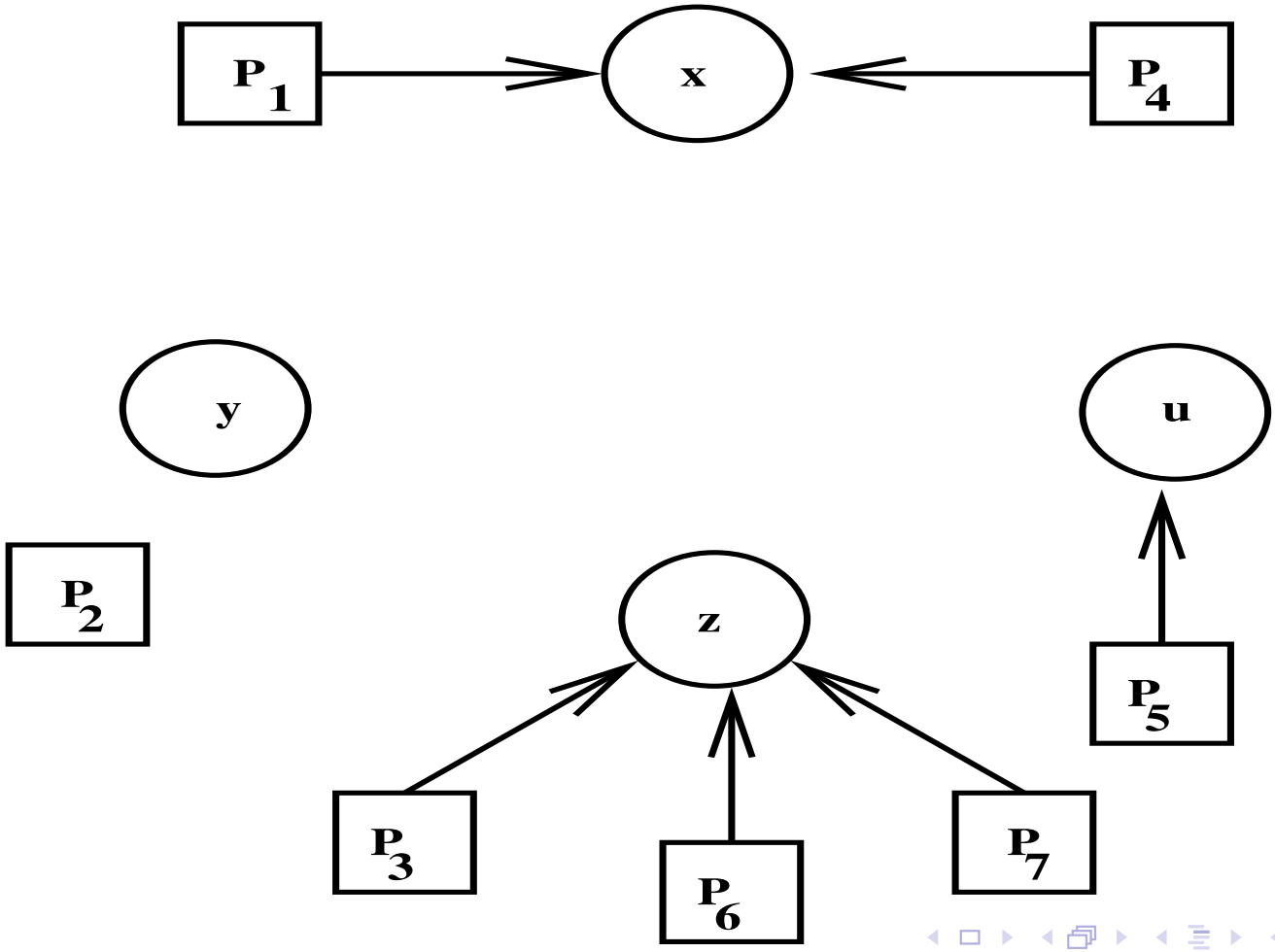
Exemple

Variable	u	x	y	z
Leader	x	*Nil*	x	x
Next	*Nil*	*Nil*	*Nil*	*Nil*
HT	*False*	*True*	*False*	*False*
RS	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
Nrel	0	0	0	0

Table: Etat initial du système

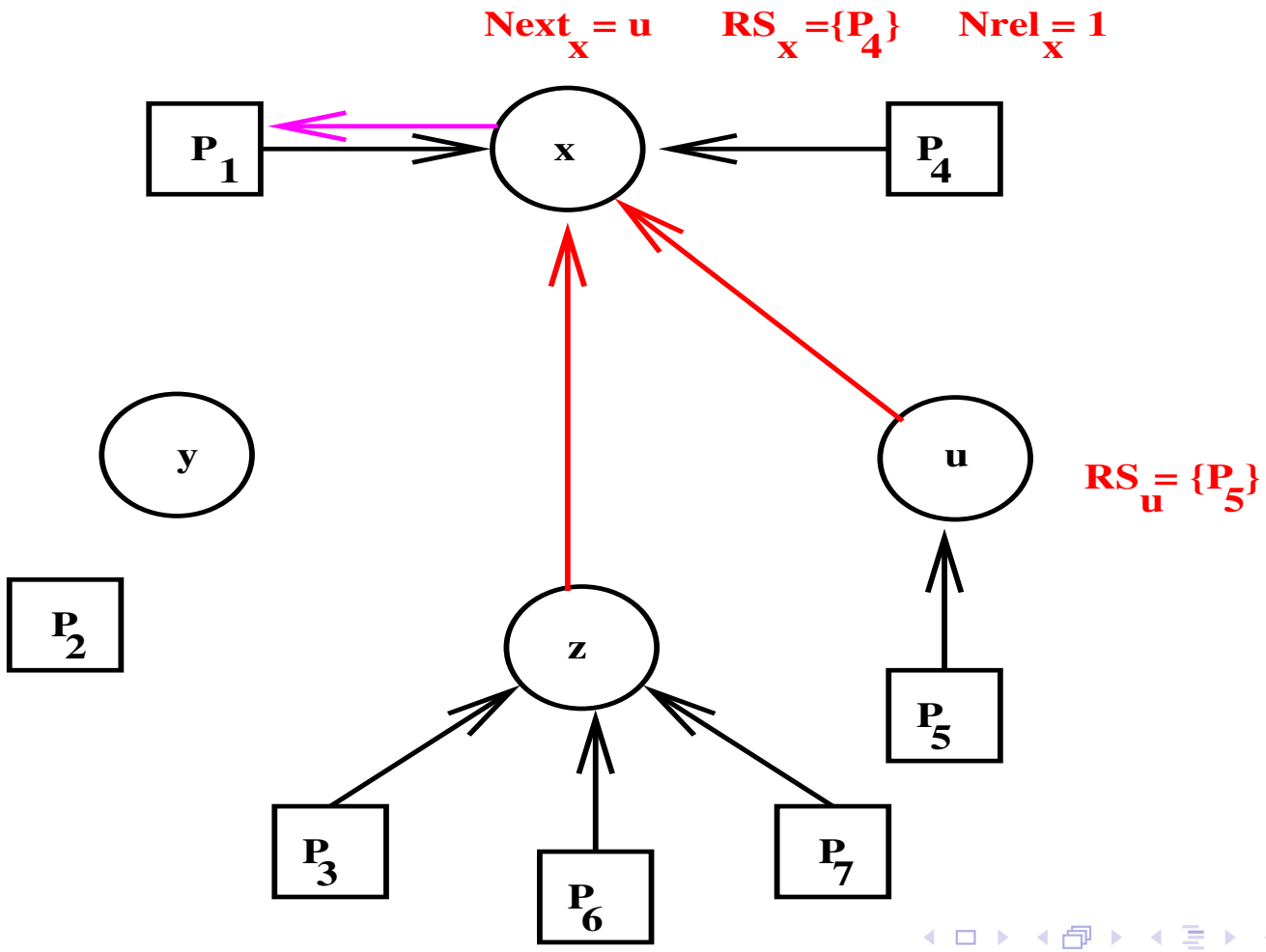
Algorithme basé sur le modèle client-serveur

Exemple



Algorithme basé sur le modèle client-serveur

Exemple



Algorithme basé sur le modèle client-serveur

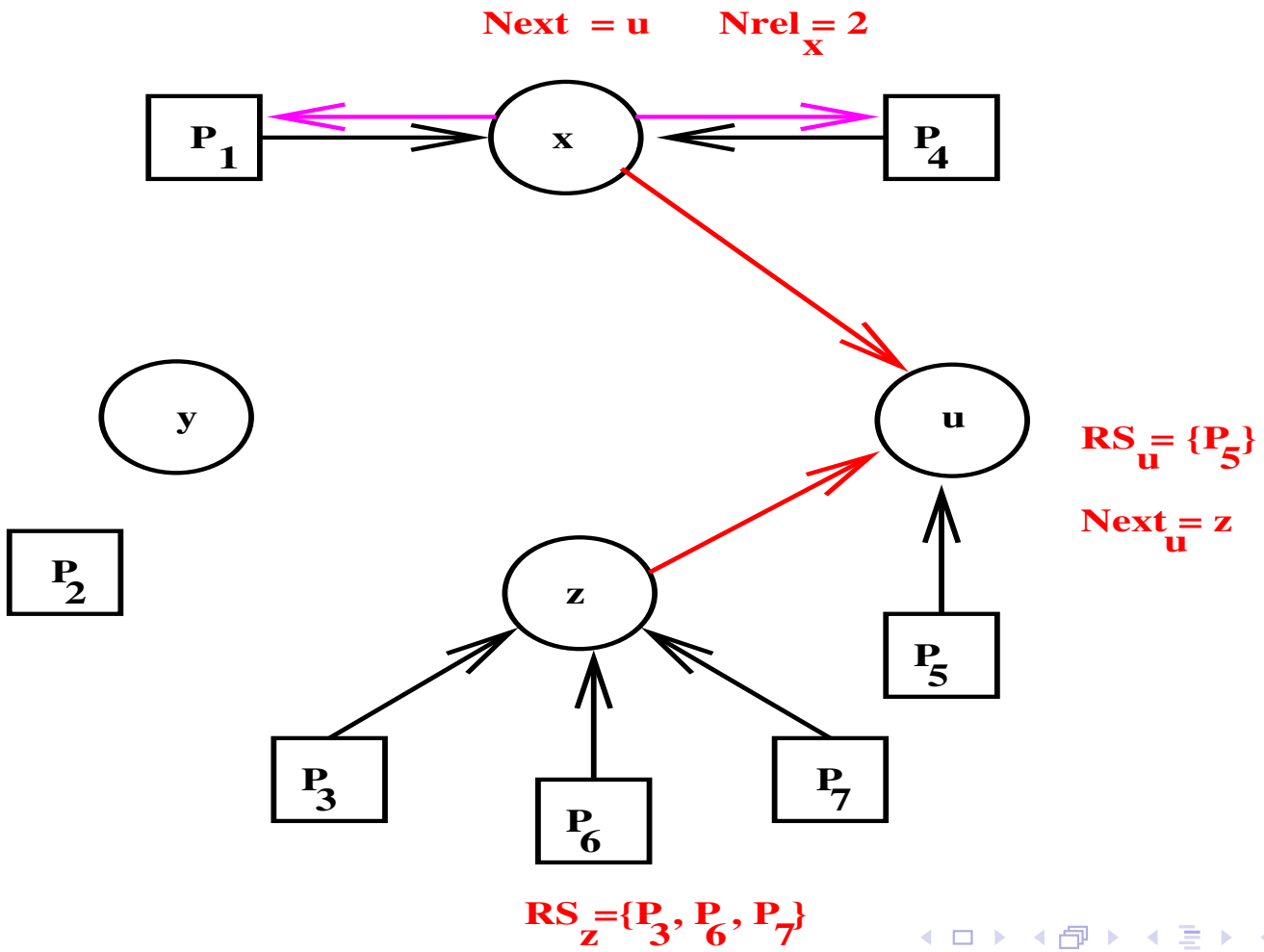
Exemple

Variable	u	x	y	z
Leader	*Nil*	u	x	u
Next	*Nil*	z	*Nil*	u
HT	*False*	*True*	*False*	*False*
RS	$\{P_5\}$	$\{P_4\}$	$\emptyset$	$\{P_3, P_6, P_7\}$
Nrel	0	1	0	0

Table: Etat global du système

Algorithme basé sur le modèle client-serveur

Exemple



Algorithme basé sur le modèle client-serveur

Preuve

Lemme (1)

Quelque soient les sessions x et y, ($HT_x \wedge HT_y$)=False est un invariant.

Théorème (1)

L'algorithme assure l'exclusion mutuelle (au plus une seule session est ouverte à la fois).

Algorithme basé sur le modèle client-serveur

Preuve

L'algorithme présenté est exempt de famine :

Lemme (2)

Une requête est transmise à une session pour laquelle (Leader = Nil) au bout d'un temps fini.

Lemme (3)

A tout moment, si nous traversons le long d'une quelconque session x, nous atteindrons une session y qui est la racine de l'arbre.

Lemme (4)

$(Next_x \neq Nil) \Leftrightarrow (Leader_x \neq Nil) \wedge (RS_x \neq \emptyset)$.

La preuve découle des lemmes (2), (3) et (4).

Algorithme basé sur le modèle client-serveur

Complexité

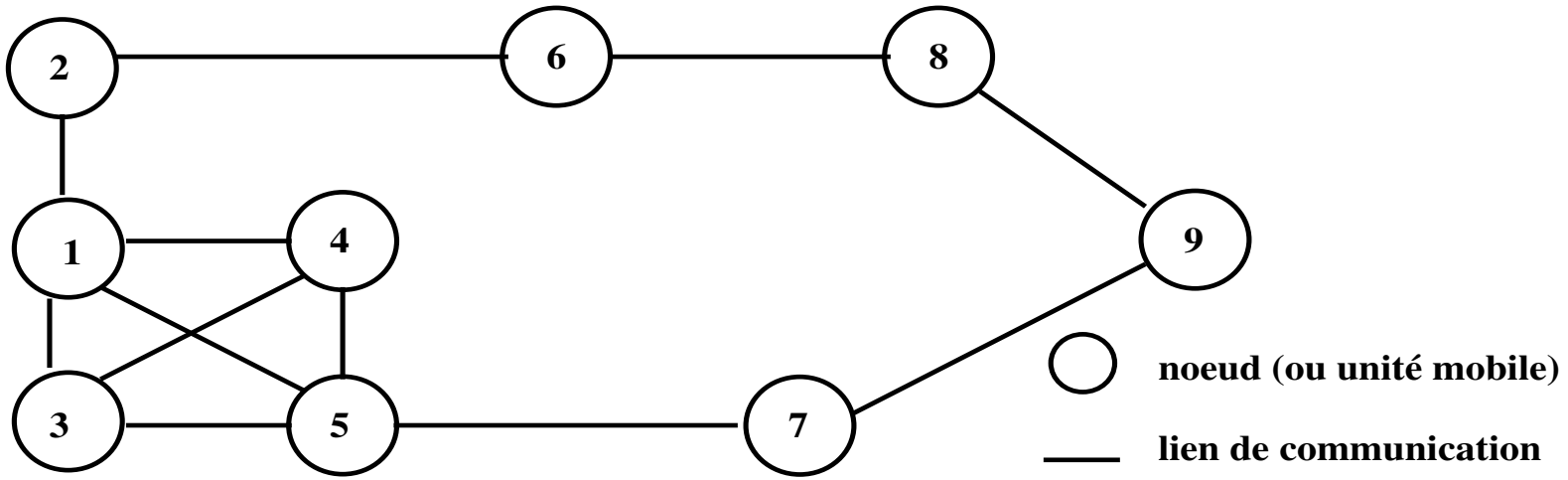
Cet algorithme a une complexité moyenne de $O(Log(m))$ messages et le degré de concurrence est n.

5- Exclusion mutuelle de groupe basée pour les réseaux mobiles ad hoc

Algorithme pour les réseaux mobiles ad hoc

Définition

Un réseau ad hoc (MANET) peut être modélisé par un graphe $G_t = (V_t, E_t)$ où
$V_t = \{noeuds\ mobiles\}$
$E_t = \{connexions\ entre\ noeuds\}$ (voir figure).
Si $e = (u, v) \in E_t$, alors les noeuds u et v sont en mesure de communiquer directement à l'instant t.



Algorithme pour les réseaux mobiles ad hoc

Modèle

- Adaptation de l'algorithme RL basé sur celui de Raymond.
- On a n noeuds $0, 1, \cdots, n-1$ et m ressources $0, 1, \cdots, m-1$.
- Unique jeton dans le réseau détenu par le noeud 0.
- Communications bidirectionnelles et FIFO.
- Utilisation de la technique d'inversion partielle $\Rightarrow$ GAD orienté selon le jeton.
- Les noeuds sont totalement ordonnés et chacun d'entre eux maintient une queue contenant les identificateurs des noeuds voisins pour lesquels il a reçu une requête.

Algorithme pour les réseaux mobiles ad hoc

Modèle

- Adaptation de l'algorithme RL basé sur celui de Raymond.
- On a n noeuds $0, 1, \cdots, n-1$ et m ressources $0, 1, \cdots, m-1$.
- Unique jeton dans le réseau détenu par le noeud 0.
- Communications bidirectionnelles et FIFO.
- Utilisation de la technique d'inversion partielle $\Rightarrow$ GAD orienté selon le jeton.
- Les noeuds sont totalement ordonnés et chacun d'entre eux maintient une queue contenant les identificateurs des noeuds voisins pour lesquels il a reçu une requête.

Algorithme pour les réseaux mobiles ad hoc

Modèle

- Adaptation de l'algorithme RL basé sur celui de Raymond.
- On a n noeuds $0, 1, \cdots, n-1$ et m ressources $0, 1, \cdots, m-1$.
- Unique jeton dans le réseau détenu par le noeud 0.
- Communications bidirectionnelles et FIFO.
- Utilisation de la technique d'inversion partielle $\Rightarrow$ GAD orienté selon le jeton.
- Les noeuds sont totalement ordonnés et chacun d'entre eux maintient une queue contenant les identificateurs des noeuds voisins pour lesquels il a reçu une requête.

Algorithme pour les réseaux mobiles ad hoc

Modèle

- Adaptation de l'algorithme RL basé sur celui de Raymond.
- On a n noeuds $0, 1, \cdots, n-1$ et m ressources $0, 1, \cdots, m-1$.
- Unique jeton dans le réseau détenu par le noeud 0.
- Communications bidirectionnelles et FIFO.
- Utilisation de la technique d'inversion partielle $\Rightarrow$ GAD orienté selon le jeton.
- Les noeuds sont totalement ordonnés et chacun d'entre eux maintient une queue contenant les identificateurs des noeuds voisins pour lesquels il a reçu une requête.

Algorithme pour les réseaux mobiles ad hoc

Modèle

- Adaptation de l'algorithme RL basé sur celui de Raymond.
- On a n noeuds $0, 1, \cdots, n-1$ et m ressources $0, 1, \cdots, m-1$.
- Unique jeton dans le réseau détenu par le noeud 0.
- Communications bidirectionnelles et FIFO.
- Utilisation de la technique d'inversion partielle $\Rightarrow$ GAD orienté selon le jeton.
- Les noeuds sont totalement ordonnés et chacun d'entre eux maintient une queue contenant les identificateurs des noeuds voisins pour lesquels il a reçu une requête.

Algorithme pour les réseaux mobiles ad hoc

Modèle

- Adaptation de l'algorithme RL basé sur celui de Raymond.
- On a n noeuds $0, 1, \cdots, n-1$ et m ressources $0, 1, \cdots, m-1$.
- Unique jeton dans le réseau détenu par le noeud 0.
- Communications bidirectionnelles et FIFO.
- Utilisation de la technique d'inversion partielle $\Rightarrow$ GAD orienté selon le jeton.
- Les noeuds sont totalement ordonnés et chacun d'entre eux maintient une queue contenant les identificateurs des noeuds voisins pour lesquels il a reçu une requête.

Algorithme pour les réseaux mobiles ad hoc

Principe de l'algorithme

Quand un noeud i veut accéder à une ressource R :

- $Q \leftarrow Q \cup \{Request(i, R)\}$ et $status \leftarrow Attente$
- Si i n'a pas le jeton et $Q = \{Request(i, R)\} \Rightarrow$ Procédure SendRequest() pour envoyer Request(i,R).
- Si i possède le jeton alors $Q \leftarrow Q - \{Request(i, R)\}$ et $status \leftarrow CS$ pour accéder à la ressource R.
  - Si i reçoit Request(j,S), comme i est en SC, il envoie un message SubToken(i,R) à tous les noeuds dans sa file Q.
    - Si $S \neq R \Rightarrow Q \leftarrow Q \cup \{Request(j, S)\}$
    - Si $S = R \Rightarrow$ les noeuds peuvent entrer en SC.
- Panne ou formation de lien $\Rightarrow$ Réearrangement partiel du GAD pour rerouter les requêtes.

Algorithme pour les réseaux mobiles ad hoc

Principe de l'algorithme

Quand un noeud i veut accéder à une ressource R :

- $Q \leftarrow Q \cup \{Request(i, R)\}$ et $status \leftarrow Attente$
- Si i n'a pas le jeton et $Q = \{Request(i, R)\} \Rightarrow$ Procédure SendRequest() pour envoyer Request(i,R).
- Si i possède le jeton alors $Q \leftarrow Q - \{Request(i, R)\}$ et $status \leftarrow CS$ pour accéder à la ressource R.
  - Si i reçoit Request(j,S), comme i est en SC, il envoie un message SubToken(i,R) à tous les noeuds dans sa file Q.
  - Si $S \neq R \Rightarrow Q \leftarrow Q \cup \{Request(j, S)\}$
  - Si $S = R \Rightarrow$ les noeuds peuvent entrer en SC.
- Panne ou formation de lien $\Rightarrow$ Réearrangement partiel du GAD pour rerouter les requêtes.

Algorithme pour les réseaux mobiles ad hoc

Principe de l'algorithme

Quand un noeud i veut accéder à une ressource R :

- $Q \leftarrow Q \cup \{Request(i, R)\}$ et $status \leftarrow Attente$
- Si i n'a pas le jeton et $Q = \{Request(i, R)\} \Rightarrow$ Procédure SendRequest() pour envoyer Request(i,R).
- Si i possède le jeton alors $Q \leftarrow Q - \{Request(i, R)\}$ et $status \leftarrow CS$ pour accéder à la ressource R.
  - Si i reçoit Request(j,S), comme i est en SC, il envoie un message SubToken(i,R) à tous les noeuds dans sa file Q.
  - Si $S \neq R \Rightarrow Q \leftarrow Q \cup \{Request(j, S)\}$
  - Si $S = R \Rightarrow$ les noeuds peuvent entrer en SC.
- Panne ou formation de lien $\Rightarrow$ Réearrangement partiel du GAD pour rerouter les requêtes.

Algorithme pour les réseaux mobiles ad hoc

Principe de l'algorithme

Quand un noeud i veut accéder à une ressource R :

- $Q \leftarrow Q \cup \{Request(i, R)\}$ et $status \leftarrow Attente$
- Si i n'a pas le jeton et $Q = \{Request(i, R)\} \Rightarrow$ Procédure SendRequest() pour envoyer Request(i,R).
- Si i possède le jeton alors $Q \leftarrow Q - \{Request(i, R)\}$ et $status \leftarrow CS$ pour accéder à la ressource R.
  - Si i reçoit Request(j,S), comme i est en SC, il envoie un message SubToken(i,R) à tous les noeuds dans sa file Q.
  - Si $S \neq R \Rightarrow Q \leftarrow Q \cup \{Request(j, S)\}$
  - Si $S = R \Rightarrow$ les noeuds peuvent entrer en SC.
- Panne ou formation de lien $\Rightarrow$ Réearrangement partiel du GAD pour rerouter les requêtes.

Algorithme pour les réseaux mobiles ad hoc

Principe de l'algorithme

Quand un noeud i veut accéder à une ressource R :

- $Q \leftarrow Q \cup \{Request(i, R)\}$ et $status \leftarrow Attente$
- Si i n'a pas le jeton et $Q = \{Request(i, R)\} \Rightarrow$ Procédure SendRequest() pour envoyer Request(i,R).
- Si i possède le jeton alors $Q \leftarrow Q - \{Request(i, R)\}$ et $status \leftarrow CS$ pour accéder à la ressource R.
  - Si i reçoit Request(j,S), comme i est en SC, il envoie un message SubToken(i,R) à tous les noeuds dans sa file Q.
  - Si $S \neq R \Rightarrow Q \leftarrow Q \cup \{Request(j, S)\}$
  - Si $S = R \Rightarrow$ les noeuds peuvent entrer en SC.
- Panne ou formation de lien $\Rightarrow$ Réearrangement partiel du GAD pour rerouter les requêtes.

Algorithme pour les réseaux mobiles ad hoc

Principe de l'algorithme

Quand un noeud i veut accéder à une ressource R :

- $Q \leftarrow Q \cup \{Request(i, R)\}$ et $status \leftarrow Attente$
- Si i n'a pas le jeton et $Q = \{Request(i, R)\} \Rightarrow$ Procédure SendRequest() pour envoyer Request(i,R).
- Si i possède le jeton alors $Q \leftarrow Q - \{Request(i, R)\}$ et $status \leftarrow CS$ pour accéder à la ressource R.
  - Si i reçoit Request(j,S), comme i est en SC, il envoie un message SubToken(i,R) à tous les noeuds dans sa file Q.
  - Si $S \neq R \Rightarrow Q \leftarrow Q \cup \{Request(j, S)\}$
  - Si $S = R \Rightarrow$ les noeuds peuvent entrer en SC.
- Panne ou formation de lien $\Rightarrow$ Réearrangement partiel du GAD pour rerouter les requêtes.

Algorithme pour les réseaux mobiles ad hoc

Principe de l'algorithme

Quand un noeud i veut accéder à une ressource R :

- $Q \leftarrow Q \cup \{Request(i, R)\}$ et $status \leftarrow Attente$
- Si i n'a pas le jeton et $Q = \{Request(i, R)\} \Rightarrow$ Procédure SendRequest() pour envoyer Request(i,R).
- Si i possède le jeton alors $Q \leftarrow Q - \{Request(i, R)\}$ et $status \leftarrow CS$ pour accéder à la ressource R.
  - Si i reçoit Request(j,S), comme i est en SC, il envoie un message SubToken(i,R) à tous les noeuds dans sa file Q.
  - Si $S \neq R \Rightarrow Q \leftarrow Q \cup \{Request(j, S)\}$
  - Si $S = R \Rightarrow$ les noeuds peuvent entrer en SC.
- Panne ou formation de lien $\Rightarrow$ Réearrangement partiel du GAD pour rerouter les requêtes.

Algorithme pour les réseaux mobiles ad hoc

Preuve

Théorème (1)

L'algorithme assure la propriété de l'exclusion mutuelle.

Théorème (2)

L'algorithme vérifie la propriété d'entrée concurrente.

Théorème (3)

L'algorithme est exempt de famine.

6-Conclusion et perspectives

- Contribution à l'exclusion mutuelle de groupe
- Solutions à l'EMG basée sur les quorums
- Solution basée sur une arborescence
- Réduction du nombre de messages par rapports aux algorithmes existants
- Solution pour les réseaux mobiles ad hoc
- Perspectives
  - Résoudre l'EMG pour les autres structures de quorums et étude de complexités.
  - Faire des simulations pour les solutions proposées
  - Utilisation d'algorithme pour bloquer le jeton sur un site s'il n'existe pas de requête en attente
  - Voir le problème d'économie d'énergie dans les réseaux ad hoc et adaption de la méthode de clusterisation
  - Situations plus compliquées de panne de liens

- Contribution à l'exclusion mutuelle de groupe
- Solutions à l'EMG basée sur les quorums
- Solution basée sur une arborescence
- Réduction du nombre de messages par rapports aux algorithmes existants
- Solution pour les réseaux mobiles ad hoc
- Perspectives
  - Résoudre l'EMG pour les autres structures de quorums et étude de complexités.
  - Faire des simulations pour les solutions proposées
  - Utilisation d'algorithme pour bloquer le jeton sur un site s'il n'existe pas de requête en attente
  - Voir le problème d'économie d'énergie dans les réseaux ad hoc et adaption de la méthode de clusterisation
  - Situations plus compliquées de panne de liens

- Contribution à l'exclusion mutuelle de groupe
- Solutions à l'EMG basée sur les quorums
- Solution basée sur une arborescence
- Réduction du nombre de messages par rapports aux algorithmes existants
- Solution pour les réseaux mobiles ad hoc
- Perspectives
  - Résoudre l'EMG pour les autres structures de quorums et étude de complexités.
  - Faire des simulations pour les solutions proposées
  - Utilisation d'algorithme pour bloquer le jeton sur un site s'il n'existe pas de requête en attente
  - Voir le problème d'économie d'énergie dans les réseaux ad hoc et adaption de la méthode de clusterisation
  - Situations plus compliquées de panne de liens

- Contribution à l'exclusion mutuelle de groupe
- Solutions à l'EMG basée sur les quorums
- Solution basée sur une arborescence
- Réduction du nombre de messages par rapports aux algorithmes existants
- Solution pour les réseaux mobiles ad hoc
- Perspectives
  - Résoudre l'EMG pour les autres structures de quorums et étude de complexités.
  - Faire des simulations pour les solutions proposées
  - Utilisation d'algorithme pour bloquer le jeton sur un site s'il n'existe pas de requête en attente
  - Voir le problème d'économie d'énergie dans les réseaux ad hoc et adaption de la méthode de clusterisation
  - Situations plus compliquées de panne de liens

- Contribution à l'exclusion mutuelle de groupe
- Solutions à l'EMG basée sur les quorums
- Solution basée sur une arborescence
- Réduction du nombre de messages par rapports aux algorithmes existants
- Solution pour les réseaux mobiles ad hoc
- Perspectives
  - Résoudre l'EMG pour les autres structures de quorums et étude de complexités.
  - Faire des simulations pour les solutions proposées
  - Utilisation d'algorithme pour bloquer le jeton sur un site s'il n'existe pas de requête en attente
  - Voir le problème d'économie d'énergie dans les réseaux ad hoc et adaption de la méthode de clusterisation
  - Situations plus compliquées de panne de liens

- Contribution à l'exclusion mutuelle de groupe
- Solutions à l'EMG basée sur les quorums
- Solution basée sur une arborescence
- Réduction du nombre de messages par rapports aux algorithmes existants
- Solution pour les réseaux mobiles ad hoc
- Perspectives
  - Résoudre l'EMG pour les autres structures de quorums et étude de complexités.
  - Faire des simulations pour les solutions proposées
  - Utilisation d'algorithme pour bloquer le jeton sur un site s'il n'existe pas de requête en attente
  - Voir le problème d'économie d'énergie dans les réseaux ad hoc et adaption de la méthode de clusterisation
  - Situations plus compliquées de panne de liens

- Contribution à l'exclusion mutuelle de groupe
- Solutions à l'EMG basée sur les quorums
- Solution basée sur une arborescence
- Réduction du nombre de messages par rapports aux algorithmes existants
- Solution pour les réseaux mobiles ad hoc
- Perspectives
  - Résoudre l'EMG pour les autres structures de quorums et étude de complexités.
  - Faire des simulations pour les solutions proposées
  - Utilisation d'algorithme pour bloquer le jeton sur un site s'il n'existe pas de requête en attente
  - Voir le problème d'économie d'énergie dans les réseaux ad hoc et adaption de la méthode de clusterisation
  - Situations plus compliquées de panne de liens

- Contribution à l'exclusion mutuelle de groupe
- Solutions à l'EMG basée sur les quorums
- Solution basée sur une arborescence
- Réduction du nombre de messages par rapports aux algorithmes existants
- Solution pour les réseaux mobiles ad hoc
- Perspectives
  - Résoudre l'EMG pour les autres structures de quorums et étude de complexités.
  - Faire des simulations pour les solutions proposées
  - Utilisation d'algorithme pour bloquer le jeton sur un site s'il n'existe pas de requête en attente
  - Voir le problème d'économie d'énergie dans les réseaux ad hoc et adaption de la méthode de clusterisation
  - Situations plus compliquées de panne de liens

- Contribution à l'exclusion mutuelle de groupe
- Solutions à l'EMG basée sur les quorums
- Solution basée sur une arborescence
- Réduction du nombre de messages par rapports aux algorithmes existants
- Solution pour les réseaux mobiles ad hoc
- Perspectives
  - Résoudre l'EMG pour les autres structures de quorums et étude de complexités.
  - Faire des simulations pour les solutions proposées
  - Utilisation d'algorithme pour bloquer le jeton sur un site s'il n'existe pas de requête en attente
  - Voir le problème d'économie d'énergie dans les réseaux ad hoc et adaption de la méthode de clusterisation
  - Situations plus compliquées de panne de liens

- Contribution à l'exclusion mutuelle de groupe
- Solutions à l'EMG basée sur les quorums
- Solution basée sur une arborescence
- Réduction du nombre de messages par rapports aux algorithmes existants
- Solution pour les réseaux mobiles ad hoc
- Perspectives
  - Résoudre l'EMG pour les autres structures de quorums et étude de complexités.
  - Faire des simulations pour les solutions proposées
  - Utilisation d'algorithme pour bloquer le jeton sur un site s'il n'existe pas de requête en attente
  - Voir le problème d'économie d'énergie dans les réseaux ad hoc et adaption de la méthode de clusterisation
  - Situations plus compliquées de panne de liens

- Contribution à l'exclusion mutuelle de groupe
- Solutions à l'EMG basée sur les quorums
- Solution basée sur une arborescence
- Réduction du nombre de messages par rapports aux algorithmes existants
- Solution pour les réseaux mobiles ad hoc
- Perspectives
  - Résoudre l'EMG pour les autres structures de quorums et étude de complexités.
  - Faire des simulations pour les solutions proposées
  - Utilisation d'algorithme pour bloquer le jeton sur un site s'il n'existe pas de requête en attente
  - Voir le problème d'économie d'énergie dans les réseaux ad hoc et adaption de la méthode de clusterisation
  - Situations plus compliquées de panne de liens

1. O. THIARE, M. NAIMI, M. GUEROUI, A Group Mutual Exclusion Algorithm for Mobile Ad Hoc Networks, IEEE 2nd International Conference on Systems, Computing Sciences and Software Engineering (SCS^2'06) CISSE 2006 December 4-14 Bridgeport, USA (2006)
2. O. THIARE, M. NAIMI, A. GUEROUI, Distributed Groups Mutual Exclusion Based on Clients/Servers Model, 7th IEEE International Conference on Parallel and Distributed Computing Applications and Technologies (PDCAT'06), pages 67-73, 4-6 December 2006 Taipei, Taiwan (2006)

Publications issues de cette thèse

3. O. THIARE, M. NAIMI, M. GUEROUI, Access Concurrents Sessions Based on Quorums, IEEE 2nd International Conference on Systems, Computing Sciences and Software Engineering (SCS^2'06) CISSE 2006 December 4-14 Bridgeport, USA (2006)
4. O. THIARE, M. NAIMI, A Quorum-Based Algorithm for Group Mutual Exclusion, ISCA 19th International Conference on Parallel and Distributed Computing Systems (PDCS'06) pages 63-69, September 20-22 San Francisco California, USA (2006)

Merci de votre attention !