Conception des systèmes répartis

3ième Année Informatique et Mathématiques Appliquées, Master SLCP
Durée : 1H45 heures — Documents autorisés : Précis et notes de cours

16 Décembre 2005

Toutes les questions valent 2 points

1 Etude d'un algorithme d'exclusion mutuelle

On considère un système composé de N processus pouvant communiquer entre eux (maillage complet). On cherche à développer un algorithme simple d'exclusion mutuelle en utilisant des protocoles ayant des propriétés adéquates pour propager les requêtes d'entrée et sortie d'exclusion mutuelle.

Une solution simple peut reposer sur un ordonnancement total des requêtes d'entrée en exclusion mutuelle. On suppose donc que les événements de requête vont être totalement ordonnés et on note $R_i(p)$ l'événement ayant le numéro d'ordre i (autrement dit la i-ème requête), le paramètre p indiquant le processus ayant émis la requête. Lorsque l'origine de la requête n'a pas d'importance, on utilisera la notation simplifiée R_i.

On note E_i l'événement d'entrée en exclusion associé à la requête i et S_i l'événement de sortie.

Rappel La propriété invariante de maintien de l'exclusion mututelle est : $\forall i, j : P_i.excl \wedge P_j.excl \Rightarrow i = j$

Spécifications

On énonce les relations causales entre événements *devant être respectées* pour garantir l'invariant d'exclusion :

- Total : L'ensemble des requêtes $\mathcal{R}$ est totalement ordonné : $\forall i, j : i < j, R_i, R_j \in \mathcal{R} \Rightarrow R_i \prec R_j$
- Vivacité : Tout événement requête est suivi d'un événement d'entrée : $\forall i : R_i \in \mathcal{R} \Rightarrow \exists E_i \in \mathcal{E} : R_i \prec E_i$
- Term : Tout événement d"entrée est suivi d'un événement de sortie : $\forall i : E_i \in \mathcal{E} \Rightarrow \exists S_i \in \mathcal{S} : E_i \prec S_i$
- Sequence : Tout événement E_i succède à un événement S_{i-1} : $\forall i > 1 : E_i \in \mathcal{E} : \exists S_{i-1} \in \mathcal{S} : S_{i-1} \prec E_i$

Propriété Un processus p est dans l'état *excl* seulement durant les intervalles $[E_i(p) : S_i(p)]$ correspondant aux requêtes qu'il a émises (c'est-à-dire pour lesquelles l'événement $R_i(p)$ existe).

Questions

1. Démontrer que les spécifications données impliquent que l'ensemble des événements d'entrée $\mathcal{E}$ est totalement ordonné.
2. Sur un exemple, montrer que les spécifications données n'impliquent pas un ordre total entre **TOUS** les événements de l'exécution du calcul c'est-à-dire $\mathcal{R} \cup \mathcal{E} \cup \mathcal{S}$. Autrement dit, exhiber deux événements non causalement liés durant une exécution en illustrant ce cas par un chronogramme.
3. Démontrer par l'absurde que les relations causales spécifiées garantissent le maintien de l'invariant.

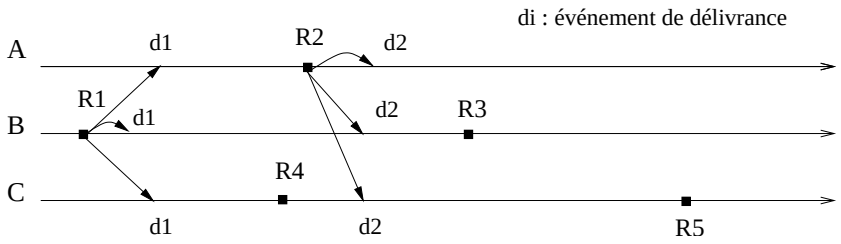


FIG. 1 – Diffusion des requêtes

Implantation

On va donc essayer d'utiliser des protocoles d'échange de messages entre les processus permettant de garantir les propriétés causales énoncées dans les spécifications. Pour ce faire, on décide d'utiliser un protocole de diffusion totalement ordonné pour diffuser les requêtes à tous les autres processus. C'est ce protocole qui implicitement, par l'ordre de délivrance des messages qu'il impose, fixe la numérotation des requêtes. L'identité du processus à l'origine d'une requête sera inclus dans le message diffusé.

Questions

4. Compléter le chronogramme de la figure 1 en supposant que les requêtes sont donc diffusées par un protocole de diffusion totalement ordonné de manière à ce que l'ordre de délivrance des requêtes reflète leur numérotation dans le chronogramme (**Utiliser la feuille jointe au sujet**).
5. Expliquez pourquoi le message de requête $R_i(p)$ issu d'un processus p peut très bien voir sa délivrance à p retardée par des requêtes issues d'autres processus (Illustrer par un chronogramme).

On décide d'utiliser le principe d'un jeton pour autoriser les processus à entrer en exclusion selon l'ordre fixé par les requêtes. On utilise un canal fiable point à point pour transmettre le jeton d'un processus à un autre. Initialement, un processus unique doit posséder le jeton. Idéalement ce doit être le processus diffusant la première requête.

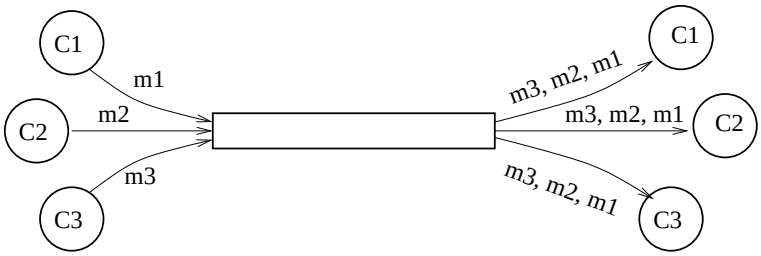
Questions

6. Montrer que l'on peut effectivement affecter le jeton au processus diffusant la première requête en précisant comment chaque processus décide s'il en est possesseur : préciser, d'une part, à quel moment de leur exécution (≡ après quel événement précis) ce test peut être exécuté et, d'autre part, quel prédidat est testé.
7. Après quel événement un processus possesseur du jeton connaît-il l'identité du processus auquel il devra transmettre son jeton ?
8. Compléter le chronogramme de la figure 1 en ajoutant les messages de transfert du jeton d'un processus à l'autre (**Utiliser la feuille jointe au sujet**). Attention à bien tenir compte **du moment** de l'événement d'émission de ces messages (voir question précédente).

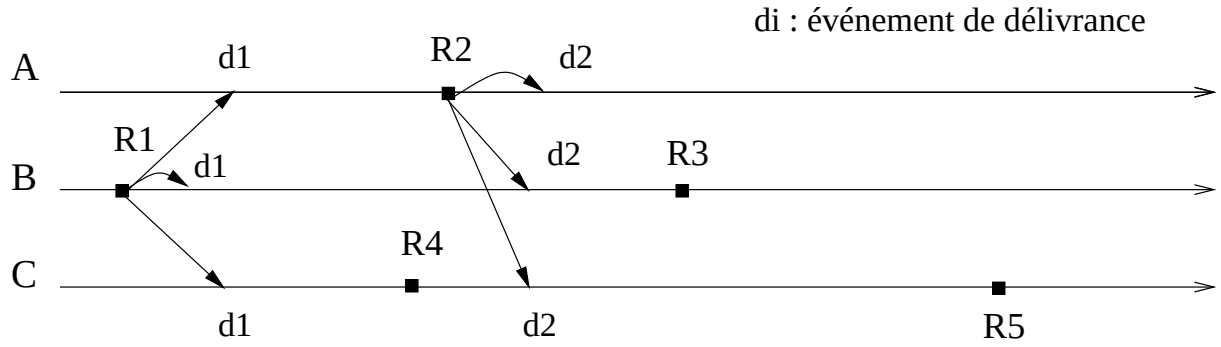
Analyse de la tolérance aux fautes

9. Monter que l'algorithme est tolérant à l'arrêt d'un processus sous certaines conditions portant sur l'état du processus au moment de son arrêt.
10. Comment un processus pourrait-il détecter la perte du jeton ? Proposer un protocole en précisant quel(s) processus serai(en)t concerné(s), pourquoi, quand, etc.

Rappel : Protocole de diffusion totalement ordonnée



Question 4



Question 8