Conception des systèmes répartis

3ième Année Informatique et Mathématiques Appliquées, Master SLCP

Correction de l'examen du 16 décembre 2005

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

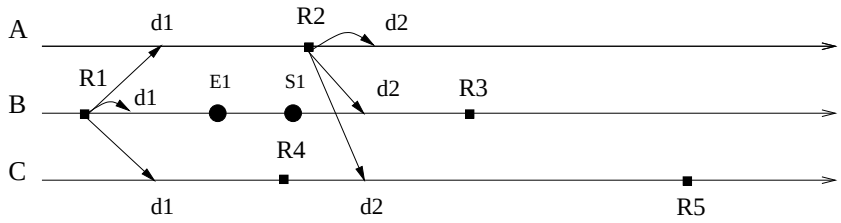
1. Démontrer que les spécifications données impliquent que l'ensemble des événements d'entrée $\mathcal{E}$ est totalement ordonné.

   Réponse : Il suffit d'utiliser les propriétés Term et Sequence :

   $$\mathsf{Term}, \mathsf{Sequence} \vdash \forall i > 0 : E_i \prec S_i \prec E_{i+1} \Rightarrow \forall i > 0 : E_i \prec E_{i+1}$$

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.

   Réponse : Les spécifications n'imposent pas forcément une causalité entre un événement d'entrée E ou sortie S et un événement de requète R comme le montre le chronogramme ci-dessous :

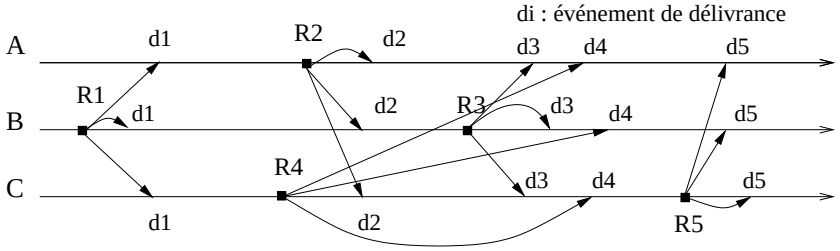


3. Démontrer par l'absurde que les relations causales spécifiées garantissent le maintien de l'invariant.
   Réponse : Pour violer la propriét d'exclusion mutuelle, il faut qu'il existe un chevauchement entre 2 intervalles $[E_i(p) : S_i(p)]$ et $[E_j(q) : S_j(q)]$ c'est-à-dire un ordonnancement causal des événements d'entrée et de sortie :
   $$E_i(p) \prec E_j(q) \prec S_i(p) \prec S_j(q) \quad avec \quad p \neq q$$
   Or, si $E_i(p) \prec E_j(q)$ alors l'ordonnancement total des événements d'entrée implique que $i < j$. Mais alors la relation causale $E_j(q) \prec S_i(p)$ est impossible d'après les propriétés Sequence et Term qui impliquent que si $i < j$ alors : $S_i(p) \prec E_{i+1}(r) \ldots \prec E_j(q)$.
   Il ne peut donc exister de tels entrelacements des intervalles et la propriété d'exclusion est donc bien garantie.

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**).
   Réponse :


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).
   Réponse : Cette situation apparaît sur le chronogramme de la question précédente : La requête R_4 émise par C ne peut pas être délivrée à C immédiatement car la délivrance des requêtes R_2 et R_3 à C n'a pas encore eu lieu. C'est donc l'ordonnancement total de délivrance des messages de requêtes qui peut conduire à retarder la délivrance d'une requête du propre processus diffuseur.

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

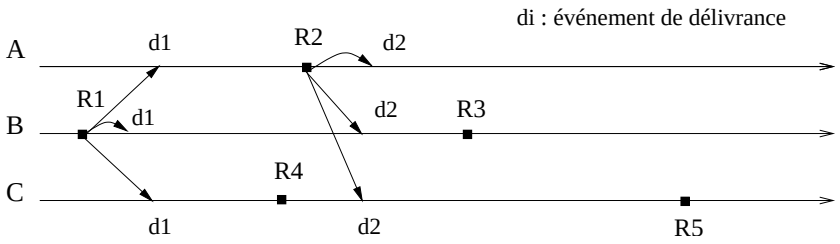


FIG. 1 – Diffusion des requêtes

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.

   Réponse : Lorsque l'événement de délivrance de la première requête se produit dans un processus p, deux cas sont possibles :
   - cette première requête délivrée est celle émise par le processus lui-même ($R_1(p)$ est délivrée au processus p) : alors le processus p sait qu'il possède la première requête et il peut donc créer le jeton et entrer en exclusion mutuelle.
   - cette première requête délivrée est celle émise par un processus différent, c'est-à-dire $R_1(q)$ avec $q \neq p$. Le processus p constate donc qu'il ne possède pas la première requête et il ne doit donc pas créer de jeton. De plus, s'il avait lui-même déjà émis une requête, il doit attendre que le jeton lui parvienne.

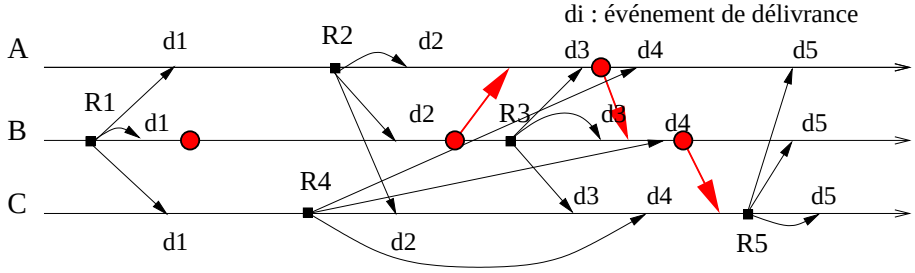
   En conclusion, le moment de l'exécution permettant de décider est la délivrance de la première requête et le prédicat testé par un processus p est sur la délivrance de $R_1(q)$: $q = p$.

7. Après quel événement un processus possesseur du jeton connaît-il l'identité du processus auquel il devra transmettre son jeton ?

   Réponse : Un processus p possesseur du jeton à la suite de sa requête $R_i(p)$ connaît l'identité du processus auquel il doit transmettre le jeton lorsqu'il reçoit la délivrance de la requête $R_{i+1}(q)$ puisqu'elle contient l'identité du processus émetteur de cette requête "suivante".

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).

   Réponse : Les événements d'entrée et de sortie ne sont pas placés pour ne pas alourdir la figure. Seuls les tranferts du jeton sont présents.



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.

   Réponse : Un processus peut s'arréter sans perturber le système global à condition qu'il soit dans un état vérifiant les deux propiétés suivantes :

il est hors exclusion, il ne possède pas le jeton et il n'a pas émis de requête

   Sous de telles conditions, les autres processus ne percoivent pas l'arrêt du processus défaillant et ils peuvent donc continuer à utiliser le service d'exclusion mutuelle.

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.

    Réponse : Si un processus p émet une requête $R_i(p)$, lorsque cette requête lui est délivrée, toutes les requêtes précédentes de 1 à $i-1$ lui sont parvenus. Il sait donc qui a pu posséder le jeton et dans quel ordre. Par exemple, la requête $R_{i-1}(q)$ lui a été forcément délivrée (on suppose que $q \neq p$, seul cas qui nous intéresse ici) et il connaît donc de quel processus viendra le jeton.

    Un processus ayant émis une requête peut armer un délai de garde. Si ce délai expire avant d'avoir reçu le jeton, le processus peut entreprendre une enquête auprès des autres processus. S'il est possible de tester l'arrêt d'un processus (par envoi d'un *ping* par exemple), le processus peut décider de tester l'état des processus dans l'ordre inverse des requêtes qui lui ont été délivrées. Tant qu'il trouve des processus arrêtés, il remonte la "chaîne" des requêtes délivrées jusqu'à un processus qui lui réponde.

    Ce processus peut alors donner son état : s'il n'est pas en attente du jeton et s'il ne le possède pas lui-même, c'est que le jeton s'est perdu et le processus enquêteur peut créer un nouveau jeton. Sinon, si le processus répondeur possède le jeton, il pourra le transmettre directement au processus enquêteur lors de sa sortie d'exclusion. Cette stratégie suppose qu'il est possible de savoir si un processus est arrêté : cette hypothèse n'est pas vérifiée dans les systèmes totalement asynchrones (voir problème du consensus). Elle peut cependant être admise dès que la durée de transmission d'un message est bornée.