Examen de Systèmes Répartis

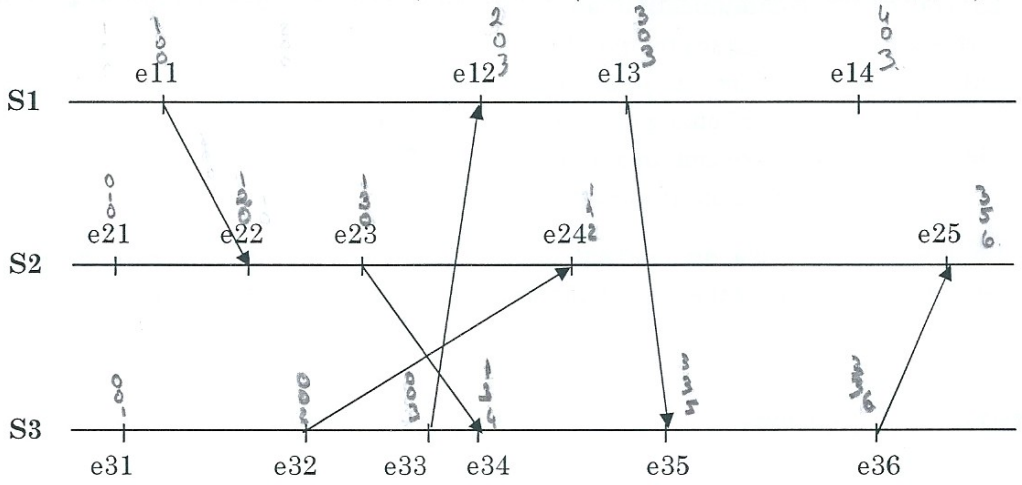
E. Mesnard

10 février 2012

Documents **autorisés : feuille A4 manuscrite Recto/Verso.**
Durée : **1 heure**

Exercice 1 (3 points) — Horloges Vectorielles

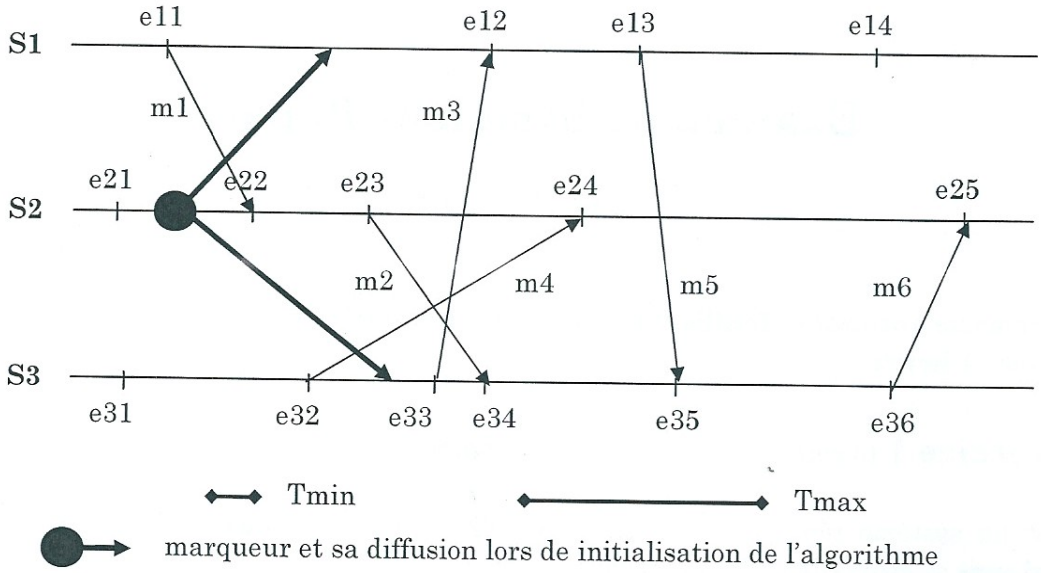
Soit un système réparti à trois sites (S1, S2 et S3) fonctionnant à l'aide d'horloges **logiques vectorielles** (HV – Lamport).
Donner les dates des événements **e14**, **e25** et **e36** (voir trace d'évolution ci-dessous) :



Exercice 2 (7 points) — Etat global Cohérent

Soit un système réparti « fortement communicant » à trois sites (S1, S2 et S3) dans lequel les communications se font dans un réseau connexe fiable, où les canaux de communication entre chaque paire de sites sont unidirectionnels et FIFO.
Ce système réparti est constitué de sites disposant chacun d'une horloge **physique** (on note Hi l'horloge physique du site i). Du fait du grand nombre de messages échangés entres les sites, on peut considérer que la synchronisation des horloges physiques entre les sites est parfaite, donc, qu'il n'y a pas de dérive des horloges entre les sites.
D'autre part, on suppose que la couche de communication bas niveau est totalement bornée : les temps de transmission de tout message (Tmsg) respectent les bornes (minimale et maximale) connues par avance : Tmin < Tmsg < Tmax

Au point indiqué dans la figure ci-après, à l'instant **t,** le site S2 (par l'émission d'un marqueur) est l'initiateur de l'algorithme de **Chandy et Lamport**. Ce site a pour objectif de définir un **état global cohérent**.



1) La couche de communication au fonctionnement « borné » permet d'en déduire certaines caractéristiques temporelles :
   a) Indiquer la date au plus tard à laquelle les sites S1 et S3 recevront le marqueur initial émis par le site S2 à l'instant t.
   b) Donner la durée maximale de la phase d'enregistrement (des messages dans les canaux de communication) sur chacun des sites. Justifier.

2) Compléter le schéma en mettant en évidence les échanges de marqueurs et les enregistrements des canaux de communication. Donner l'état enregistré.

Exercice 3 (10 points) — Exclusion mutuelle par jeton

Certains algorithmes d'exclusion mutuelle dédiés aux systèmes répartis sont basés sur la notion de jeton : le site détenteur du jeton peut entrer en Section Critique – SC (donc, accéder à la ressource) s'il le souhaite ; par contre, il se doit de transmettre le jeton aux sites qui en auraient éventuellement fait la demande.
L'objectif de cet exercice est de comparer deux algorithmes de ce type : Suzuki/Kasami et Raymond. Les comparaisons seront réalisées sur un système répartis constitué de 4 sites (S1 à S4, notés également Si ou Pi sur les résumés des algorithmes).

Algorithme de Suzuki/Kasami

Rappels sur le principe : Les sites sont en réseau connexe (tous interconnectés). Un site demandeur diffuse sa requête (REQ) à tous les autres sites. Celui qui a le jeton répond.
L'algorithme s'appuie sur les structures de données suivantes :

- sur chaque site Si : **RNi** (RNi[j] est le numéro d'ordre de la dernière demande reçue sur Si, de la part du site Sj).
- sur le jeton : **Q** (file contenant les demandes des sites pour entrer en Section Critique, et ordonnée selon les instants d'arrivée de ces demandes), et **LN** (LN[i] est le numéro d'ordre de la dernière entrée effective en SC du site Si).

Suzuki / Kasami en résumé :

Demande d'entrée en SC (sur p_i)

```
if  not jeton_présent
      RN[i] = RN[i] + 1
      diffuser(REQ, i, RN[i])
      attendre (jeton_présent)
sc_en_cours = true
entrer en section critique
```

Sortie de SC

```
sc_en_cours = false
LN[i] = RN[i]
for all j | p_j ∉ queue
      if RN[j] == LN[j] + 1
         entrer_queue (j)
if not queue_vide
      k = sortir_queue
      envoyer (k, JETON)
```

Réaction à l'arrivée d'un message sur p_i

· réception de (REQ, j, n):

```
RN[j] = max (RN[j], n)
if  jeton_présent and not sc_en_cours
      if RN[j] == LN[j]+1 (ce qui entre autre)
         envoyer (j, JETON)
         jeton_présent = false - donne jeton
```

· réception de JETON :

```
jeton_présent = true
sc_en_cours = true
entrer en section critique·
```

Algorithme de Raymond

Principe : Contrairement à l'algorithme de Suzuki/Kasami, cet algorithme s'appuie sur une organisation particulière des sites. Les sites sont organisés en **arbre orienté**, reconfigurable **dynamiquement**. La racine est à tout moment le site qui possède le jeton, et les arcs sont toujours orientés en direction de la racine. D'autre part, l'algorithme s'appuie sur les structures de données suivantes :

- Le jeton ne porte pas d'information (juste un booléen de présence).

Par contre, chaque site gère :

- un pointeur **ptr**, construisant l'arc orienté, indiquant le voisin à qui transmettre.
- une queue **Q**, enregistrant les requêtes qui transitent par le site.

Exclusion mutuelle répartie (Raymond)

Demande d'entrée en SC (sur p_i)

```
if  not jeton_présent
      if queue_vide
         envoyer(ptr, REQ)
      entrer_queue (i)
      attendre (jeton-présent)
sc_en_cours = true
entrer en section critique
```

Sortie de SC

```
sc_en_cours = false
if not queue_vide
      ptr = sortir_queue
      envoyer (ptr, JETON)
      jeton_présent = false
      if not queue_vide
         envoyer(ptr, REQ)
```

Réaction à l'arrivée d'un message sur p_i (depuis p_j)

réception de REQ:

```
if  jeton_présent
      if sc_en_cours
         entrer_queue (j)
      else
         ptr = j
         envoyer (ptr, JETON)
         jeton_présent = false
else
      if  queue_vide
         envoyer(ptr, REQ)
      entrer_queue (j)
```

réception de JETON :

```
ptr = sortir_queue
if  ptr = i
      jeton_présent = true
else
      envoyer (ptr, JETON)
      if not queue_vide
         envoyer(ptr, REQ)
```

Cet algorithme peut donc se résumer à :

a) un site ne peut entrer en Section Critique que lorsqu'il possède le jeton. Sinon, il enregistre les requêtes dans une queue Q locale.

b) un site avec une queue non vide enverra la première requête à son pointeur, à moins de l'avoir déjà fait. Il se mettra alors en attente du jeton.

c) quand la racine de l'arbre reçoit une requête, elle transmettra le jeton à son voisin en tête de la queue locale Q, dès qu'elle quittera la Section Critique. Ensuite, elle changera la direction de son arc, et pointera vers ce site.

d) à la réception d'un jeton, un site non demandeur doit :
   - le transmettre à son voisin en tête de la queue Q,
   - retirer cette requête de Q,
   - pointer vers ce voisin,
   - et si la queue n'est pas vide, envoyer une requête au pointeur.

1) Donner les valeurs des RNi, LN et Q, de l'algorithme de **Suzuki/Kasami**, quand le système évolue de la manière suivante :

a) Etat initial : le site S1 diffuse une requête et obtient, de la part du système d'exploitation, le jeton (qui va ensuite circuler normalement entre les sites).

b) Concurrence 2 et 3 : le site S2, **puis** le site S3 demandent à entrer en Section Critique, alors que le S1 n'a toujours pas libéré la ressource.

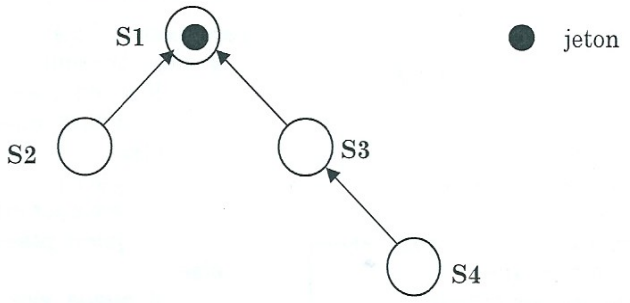
c) Libération : S1 libère la ressource et transmet le jeton.

d) Demande 4 : S4 veut également accéder à la ressource, alors qu'elle est toujours occupée.

e) Fin évolution : aucun autre site ne fait de demandes.

2) La particularité de l'algorithme de **Raymond** est que la transmission du jeton (du possesseur vers le demandeur) provoque la modification de l'arbre dynamiquement : l'arc est alors orienté dans l'autre sens. Indiquer à quelle(s) ligne(s) de l'algorithme cette modification dynamique est réalisée.

3) En prenant comme structure initiale des sites l'organisation suivante :



analyser, avec l'algorithme de **Raymond**, l'évolution décrite à la question 1). Pour cela, préciser les changements d'orientation des arcs de l'arbre orienté au cours du temps. Redessiner les orientations au fur et à mesure du transfert du jeton. Mettre en évidence les messages échangés et indiquer les valeurs des queues locales Qi.

4) Conclure en comparant les résultats obtenus par cet algorithme de Raymond avec ceux obtenus en appliquant l'algorithme de Suzuki/Kasami.

Vendredi 10 février 2012

Examen de Système Réparti (suite)
Troisième Année F1-F2-F5
1 feuille recto/verso manuscrite

Questions : (20 pts)

1. Pour les Web Services, a quoi sert le WSDL ? (1 pt)

2. Quels sont les points communs et les différences entre un web service utilisant la technologie REST et celle utilisant la méthode XML-RPC ? (3 pts)

3. Actuellement, on parle de "haute disponibilité" au niveau des serveurs. (4 pts)
   a. Qu'est-ce que cela signifie ?
   b. Quelles architectures et mécanismes se cachent derrière ce concept ?
   c. Expliquez le fonctionnement de Heartbeat sous linux.
   d. Est-ce que cela permet de s'assurer contre les programmes malveillants ?

4. On parle d'orchestration des services. (5 pts)
   a. Qu'est-ce que cela signifie ? Quel est l'intérêt de cette technique ?
   b. On aimerait tester un orchestrateur, c'est-à-dire vérifier s'il est conforme à sa spécification.
      i. Que veut-dire « être conforme » à sa spécification ? Qu'est-ce que cela sous-entend ?
      ii. Quelle architecture de test est alors obligatoire ? Explicitez votre réponse (si nécessaire avec un schéma).
      iii. A quoi doit-on faire attention pour tester cet orchestrateur afin d'être sûre que les tests soient justes ?

5. Afin de pouvoir gérer les événements dans un système réparti, on peut utiliser différentes sortes d'horloges. (4 pts)
   a. Quelles sont-elles ?
   b. Dans quelle ordre sont-elles apparues et pourquoi ?
   c. NTP utilise l'algorithme de Marzullo, plutôt que celui de Cristian. Pourquoi, quel était le défaut de l'algorithme de Cristian ?

6. Est-ce que NFS (Network File System) est compatible avec Kerberos ? Explicitez votre réponse ainsi que les avantages et inconvénients de cette solution. (3 pts)