Examen ACCOV

Tous documents autorisés

Jeudi 20 Juin 2002, durée 2h30

On s'intéresse au problème de prise de rendez-vous bi-points dans un contexte d'environnement réparti composé de plusieurs sites concurrents. On suppose que le support de communication inter-sites est fiable et que les sites ne tombent pas en panne. On ne s'intéressera pas ici au mécanisme utilisé pour mettre fin au rendez-vous.

1 Première version

Dans un premier temps, un site X qui veut prendre rendez-vous envoie une requête à un serveur. Celui-ci attend qu'un autre site Y demande également un rendez-vous puis autorise le rendez-vous entre X et Y. Dans cette version les sites ne peuvent choisir avec qui ils obtiennent le rendez-vous. Une première modélisation, dans laquelle on abstrait le comportement du serveur, donne le réseau suivant :

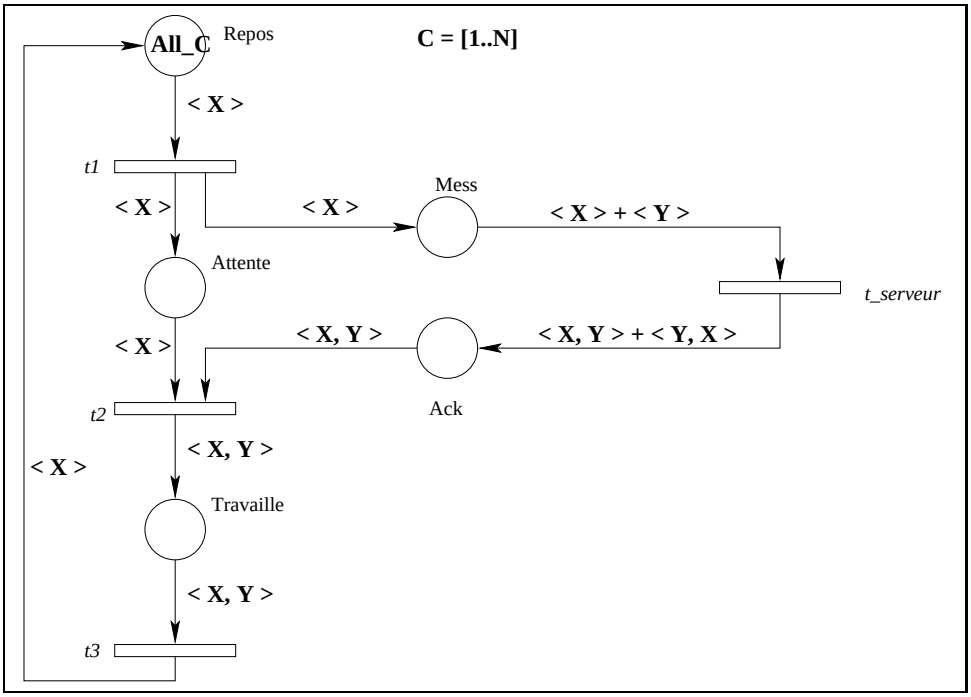

Figure 1: Le modèle initial

Question 1 (1 point)

Donner une courte interprétation des transitions $t1$, $t2$, $t3$ et $t\_serveur$. Donner également un exemple (en terme de séquence de franchissements) de rendez-vous réussi (par exemple entre le site 1 et le site 2).

Avant d'analyser ce modèle, on le réduit une première fois et on obtient le réseau suivant (figure 2) :

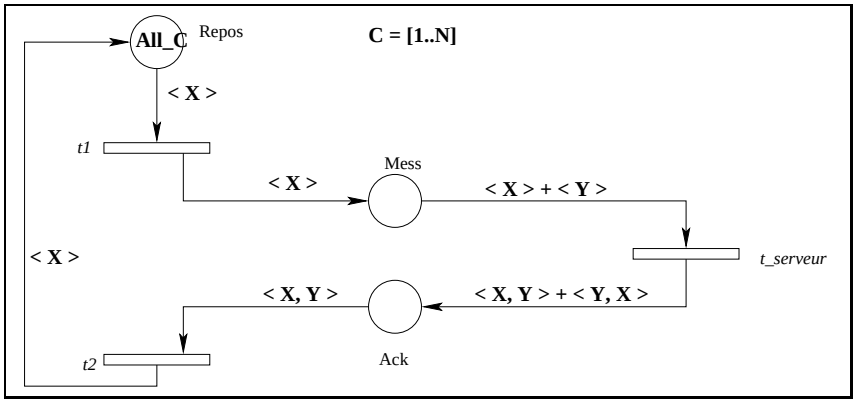

Figure 2: Le précédent modèle un peu réduit

Question 2 (1 point)

Si l'on suppose que l'on a d'abord supprimé la place *Attente* qui était une place implicite, dire quelle autre réduction a été appliquée sur le modèle de la figure 1 pour obtenir celui de la figure 2.

Question 3 (2 points)

Démontrer qu'il n'y a pas d'interblocage dans ce dernier modèle dès que $N \geq 2$; vous pourrez démontrer que

1. si $t1$ et $t\_serveur$ ne sont pas franchissables alors $t2$ l'est nécessairement;
2. si $t2$ et $t\_serveur$ ne sont pas franchissables alors $t1$ l'est nécessairement;
3. si $t1$ et $t2$ ne sont pas franchissables alors $t\_serveur$ l'est nécessairement.

Conclure alors sur les problèmes d'interblocage pour le modèle initial.

On souhaite implémenter cette première version. Pour cela on détaille le comportement du serveur (voir figure 3) :

1. le serveur forme un couple dès qu'il le peut;
2. sa variable interne `Premier` lui permet de savoir s'il s'agit d'une première demande d'un couple; dans ce cas, il sauvegarde l'identité de ce premier site dans la variable `Id_En_Attente`, et met en attente ce site sur une entrée interne `Attente_Interne` (transition $te1$).

Dans le cas contraire (appel d'un site Y alors qu'il y a déjà un autre site X en attente interne), il libère le site Y en lui signifiant qu'il sera en rendez-vous avec X et sauvegarde l'identité de Y (transition $te2$);

3. lorsqu'un second site Y a appelé le serveur, celui-ci libère le site en attente X et lui signifie qu'il est en rendez-vous avec Y (transition $ts1$).

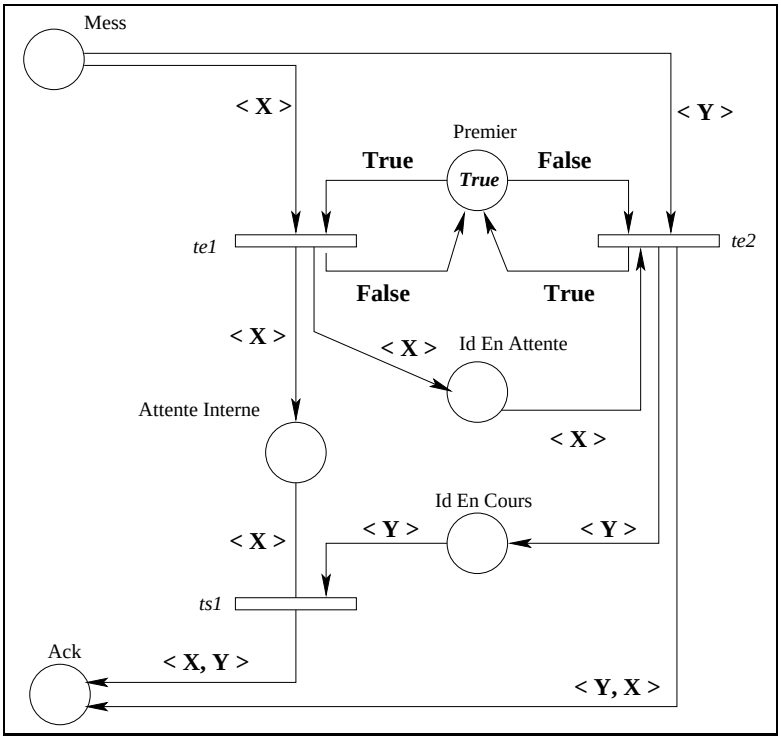

Figure 3: Détails sur le serveur

Question 4 (2 points)

Montrer par un raisonnement similaire à celui de la question 3 que ce comportement du serveur n'introduit pas d'interblocage.

Question 5 (1+2 points)

Ce dernier modèle ne souffre pas du problème d'interblocage. Il pose néanmoins un problème au niveau de la cohérence des rendez-vous dans le cas où le serveur tarde à relâcher le site en attente. Montrer par exemple que l'on peut atteindre un état où le site 1 pense être en rendez-vous avec le site 2 alors que le site 2 pense être en rendez-vous avec le site 4.

Dire pourquoi l'implantation du serveur avec un objet protégé et une gestion de l'attente interne par une entrée interne gardée permet d'éviter ce problème.

Question 6 (4 points)

Implémenter le corps de l'objet protégé `Serveur` en utilisant le programme proposé en Annexe 1 et les règles de comportement du serveur énoncées plus haut.

2 Seconde version

On souhaite maintenant permettre à un site X de demander explicitement un rendez-vous avec un site Y. Pour cela on distingue deux possibilités de comportements des sites :

- un site X peut demander un rendez-vous à un site Y (transition $t1$); il devra attendre que le site Y accepte des rendez-vous;
- un site Y peut se mettre en attente de demande de rendez-vous avec lui (transition $t4$); dans ce cas, il attend qu'un site quelconque X veuille prendre rendez-vous avec lui.

La modélisation de ce comportement est donné figure 4 (de nouveau on abstrait le comportement du serveur de gestion des rendez-vous).

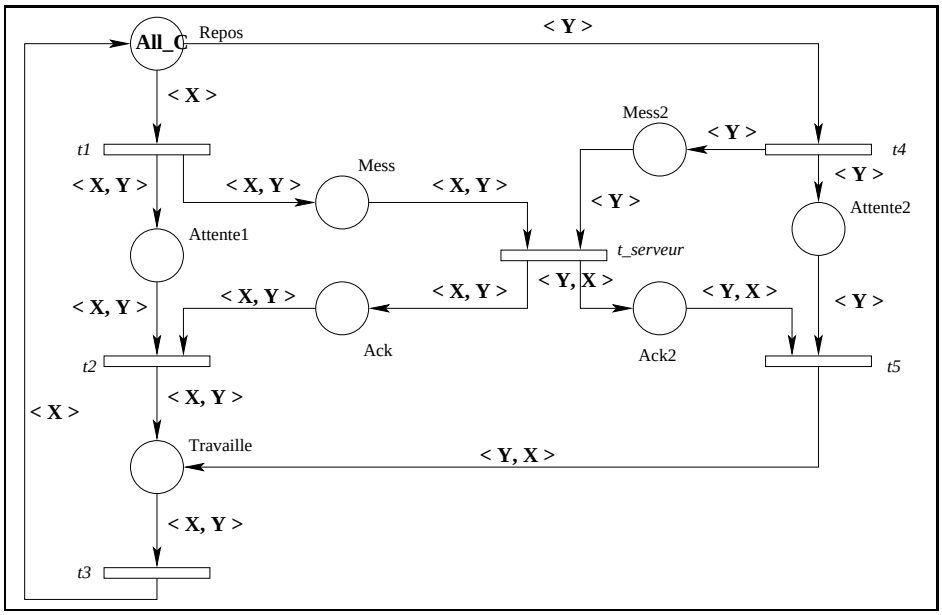

Figure 4: Le second modèle

Question 7 (2 points)

Montrer que maintenant il y a deux types possibles d'interblocage.

Question 8 (1+4 points)

Proposez une stratégie permettant d'éviter ces problèmes d'interblocage et proposez une seconde version de l'objet protégé serveur du programme donné en Annexe 1 implémentant cette stratégie (remarque : vous pourrez considérer que le serveur peut envoyer soit une réponse positive soit une réponse négative lorsqu'il détecte une possibilité d'interblocage).

Annexe 1

```
with Text_Io; use Text_Io;

procedure Prog1 is

   N: constant Natural := 5;
   type Id is mod N;
   --=============================================================
   protected Numero is
      procedure Get(X: out Id);
   private
      Current : Id := Id'First;
   end Numero;

   protected body Numero is
      procedure Get(X: out Id) is
      begin
         X := Current; Current := Current + 1;
      end Get;
   end Numero;
   --=============================================================
   protected Serveur is
      entry Demande(X: in Id; Y: out Id);
   private
      entry Attente_Interne(X: in Id; Y: out Id);
      Premier          : Boolean := True;
      Couple_Possible  : Boolean := False;
      Id_En_Attente    : Id;
      Id_En_Cours      : Id;
   end Serveur;

   protected body Serveur is separate;
   --=============================================================
   procedure Work(X,Y: in Id) is
   begin
      Put_Line(Id'Image(X) & " travaille avec " & Id'Image(Y));
   end Work;
   --=============================================================
   task type Site;

   task body Site is
      Ego: Id;
      Y  : Id;
   begin
      Numero.Get(Ego);
      loop
         Serveur.Demande(Ego, Y);
         Work(Ego, Y);
      end loop;
   end Site;
   --=============================================================
   Les_Sites : array(Id) of Site;
begin
   null;
end Prog1;
```