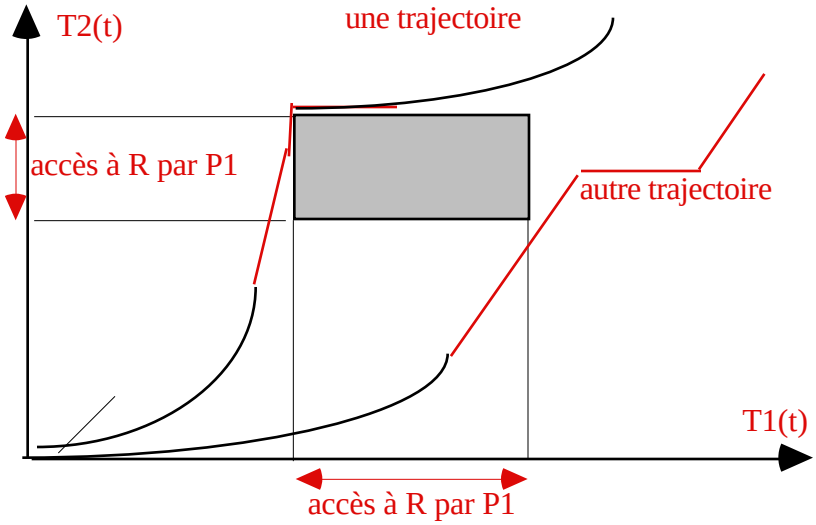
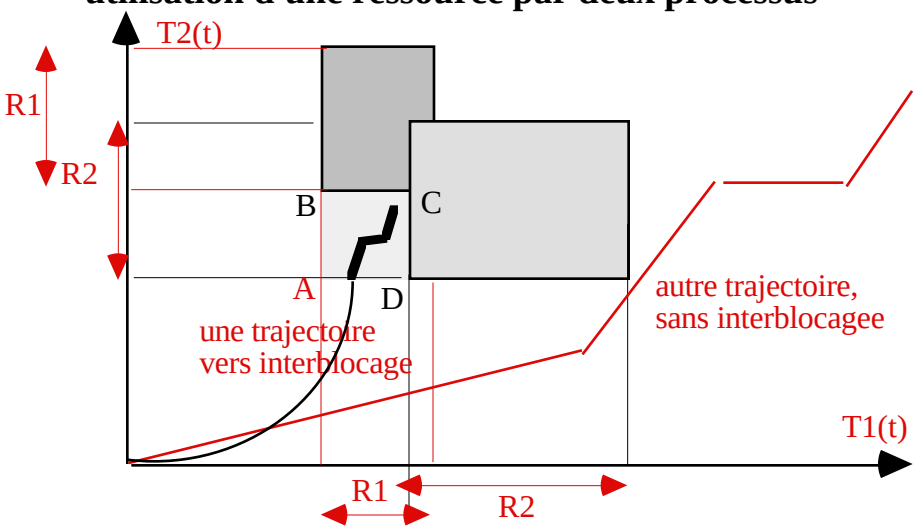
DOSSIER INTERBLOCAGE

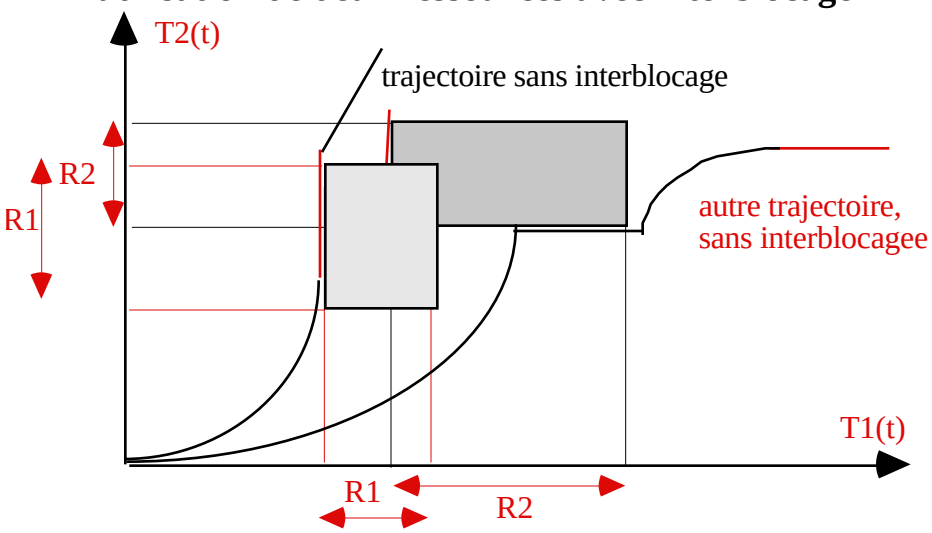
Description graphique simple



utilisation d'une ressource par deux processus



utilisation de deux ressources avec interblocage



utilisation ordonnée de deux ressources (sans interblocage)

UNE ALLOCATION FIABLE PEUT ÊTRE INÉQUITABLE

Exemple 1. : n = 6 ; m = 18 ; C = (4 4 4 4 4 4)

à t0 : A = (3 3 3 3 3 2) ; Dist = (1 1 1 1 1 2) ; X = 1 ;

à t1 : P6 demande deux ressources ; req(6) = 2 ; l'état résultant n'est pas fiable ; P6 est bloqué ;

à t2 : P1 demande une ressource ; il est servi ; X = 0 ;

à t3 : P2, P3, P4, P5 demandent tous 1 ressource et sont bloqués ;
req(2) = req(3) = req(4) = req(5) = 1

à t4 : P1 libère 1 ressource ; X = 1 ; P2 est servi; X = 0 à nouveau.
A = (3 4 3 3 3 2) ; Dist = (1 0 1 1 1 2) ; X = 0.

à t5 (resp. t6 et puis t7) : P1 libère une ressource ; X = 1 ;
P3 (resp. P4 et puis P5) est servi; X = 0 .

Noter que P1 a libéré toute son annonce et A(1) = 0 à t7.

à t8 (resp. t10, puis t12) : P1 demande 1 ressource ; il est bloqué.

à t9 (resp. t11 et puis t13) : P2 (resp. P3 et puis P4) libère une ressource et P1 est servi.

à t14 : P5 libère une ressource. C'est le même état qu'en t1. D'où famine pour P6

Exemple 2. n = 5 ; m = 6 ; C = (6 5 2 2 2)

à t0 : A = (1 1 1 1 1) ; Dist = (5 4 1 1 1) ; X = 1 ;

à t1 : P2 demande une ressource ; l'état résultant n'est pas fiable ; P2 est bloqué.

à t2 : P3, P4, P5 demandent une ressource ; P3 seul est servi ; X = 0. P4 et P5 sont bloqués.

à t3 : P3 rend tout ; X = 2 ; P4 et P5 sont servis ; X = 0.

à t4 : P3 démarre 1 transaction, demande 1 ressource, est bloqué.

à t5 : P4 rend tout. X = 2 ; P2 ne peut pas être servi ; P3 est servi ; X = 1.

à t6 : P4 démarre une nouvelle transaction et demande une ressource ; il est servi.

à t7 : P5 rend tout. X = 2 ; P2 ne peut toujours pas être servi.

à t8 : P5 demande une ressource et est servi.
L'état d'allocation est le même qu'en t1.

A = (1 1 1 1 1) ; Dist = (5 4 1 1 1) ; X = 1 ;

P2 est toujours bloqué. Famine

Si P1 avait aussi fait une requête, il aurait été aussi en famine.

LE SERVICE À L'ANCIENNETÉ RÉINTRODUIT L'INTERBLOCAGE

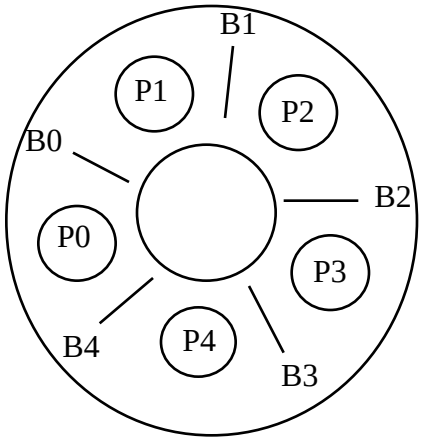
Exemple 3. : n = 3 ; m = 4 ; C = (4 2 1)

à t0, l'état d'allocation est : A = (2 1 1) ; Dist = (2 1 0) ; X = 0.
à t1, P1 demande une ressource ; P1 est bloqué.
à t2, P2 demande aussi une ressource ; P2 est mis en queue derrière P1.
à t3, P3 libère toute son annonce, i.e. une ressource. L'allocateur regarde si P1 peut être servi. L'état A (3 1 0) n'est pas fiable puisqu'il conduit à interblocage quand P1 and P3 demandent chacun une ressource supplémentaire ; la requête de P1 est refusée. Et comme le service est FIFO, la requête de P2 n'est pas examinée alors qu'elle ne conduit pas à interblocage.
à t4, P3 demande une nouvelle ressource ; il est bloqué et mis en queue derrière P2
Le système se trouve alors en interblocage avec X = 1.
Notons bien qu'en servant P2 (dans un service non FIFO) on mettrait le système dans un état fiable, sans interblocage.

Exemple 4. : n = 3 ; m = 6 ; C = (4 4 4) ; annonces identiques

à t0 : A = (0 2 4) ; Dist = (4 2 0) ; X = 0;
à t1, P1 demande une ressource supplémentaire ; P1 est bloqué.
à t2, P2 demande aussi une ressource ; P2 est mis en queue derrière P1.
à t3, P3 libère deux ressources, X = 2.
P1 n'est pas servi puisque l'état A = (1 2 2) n'est pas fiable.
à t4, P3 demande une ressource ; il est mis en queue derrière P2.
Le système est en interblocage avec X = 2.
Noter que servir P2 or P3 (service non FIFO) aurait mené à un état fiable.

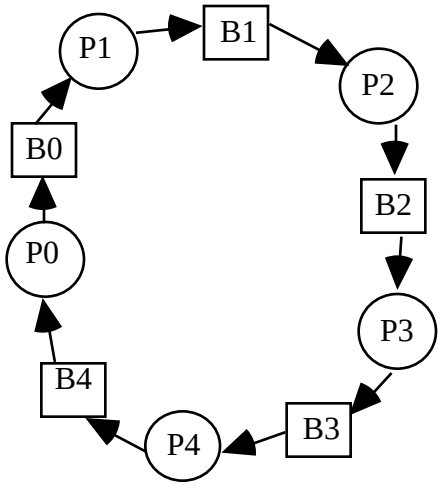
LE REPAS DES PHILOSOPHES



HYPOTHÈSES : 1 Chaque assiette a une place fixe
2. Chaque baguette a une place fixe
3. accès à chaque baguette en exclusion mutuelle

PROTOCOLE : 1. Pour manger, un philosophe doit utiliser deux baguettes, la sienne et celle de son voisin de droite
2. Tout philosophe ne mange que pendant un temps fini , puis restitue les deux baguettes

PROPRIÉTÉS : 1 pas d'interblocage
2 pas de coalition (famine)



graphe d'allocation avec état d'interblocage

LE REPAS DES PHILOSOPHES

SOLUTIONS

1. Allocation globale des deux baguettes (mais famine)

2. Méthode des classes ordonnées

Baguette de plus petit numéro d'abord

3. Prévention dynamique

On garde toujours disponible une baguette pour qu'un philosophe au moins puisse toujours recevoir sa deuxième baguette

PROTOCOLE 0. : un philosophe doit manger assis et prendre d'abord une chaise. Or il n'y a que quatre chaises

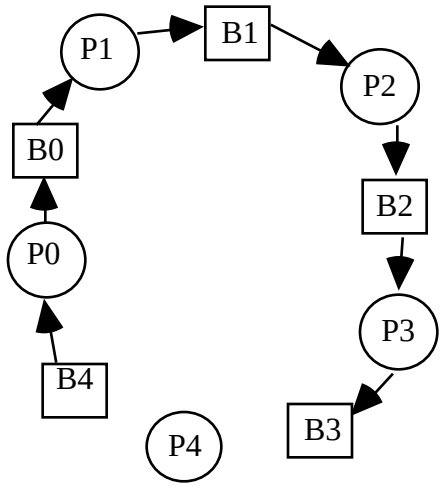
4 Détection d'interblocage. Guérison brutale

On sacrifie un philosophe, le philosophe 0 par exemple et on lui reprend la baguette qu'il a obtenue

5. Prévention pessimiste par estampillage des transactions (repas)

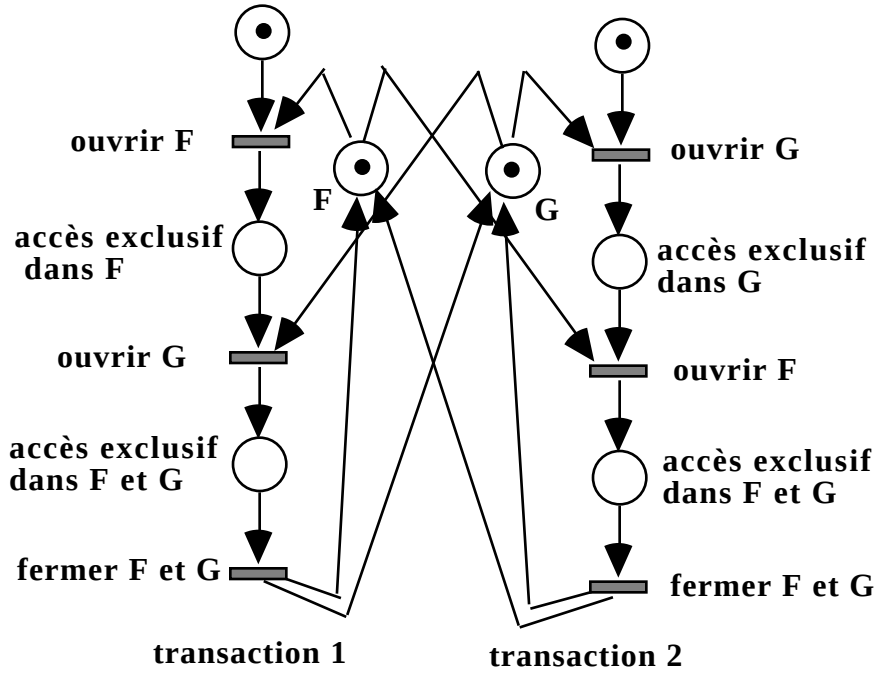
On date les transactions. Quand un philosophe demande une baguette déjà allouée à un autre philosophe, il n'attend que si la date de sa transaction est plus ancienne que celle de l'utilisateur actuel. Sinon, il n'attend pas, rend l'autre baguette et recommence sa transaction.

Supposons que e0 < e1 < e2 < e3 < e4

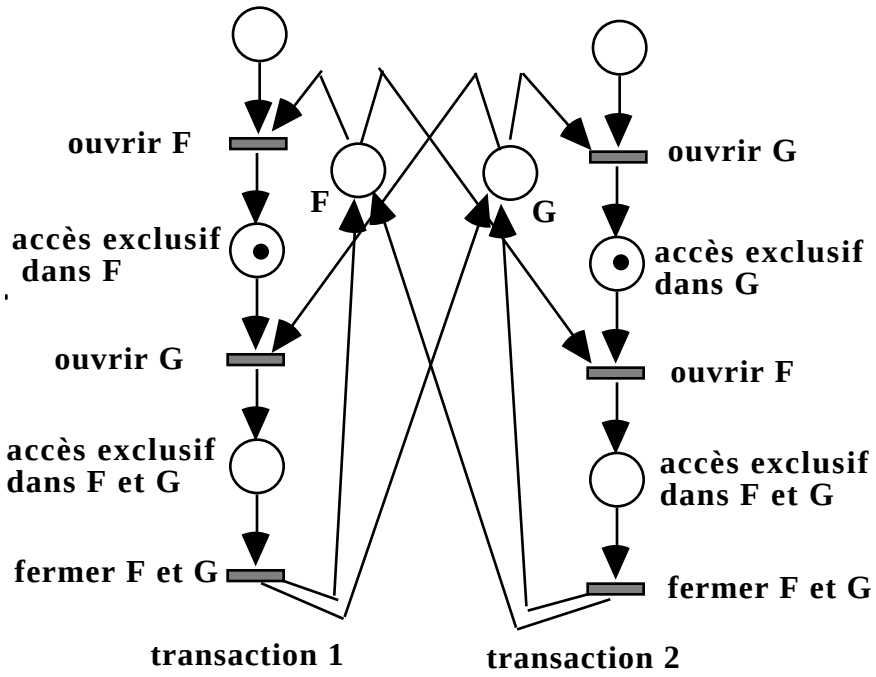


P4, qui fait la transaction la plus récente, n'attend pas B4 attribuée à P0 et rend B3 que P3 va pouvoir utiliser

modélisation par réseaux de Petri : interblocage = non vivacité
TRANSACTIONS ET FICHIERS



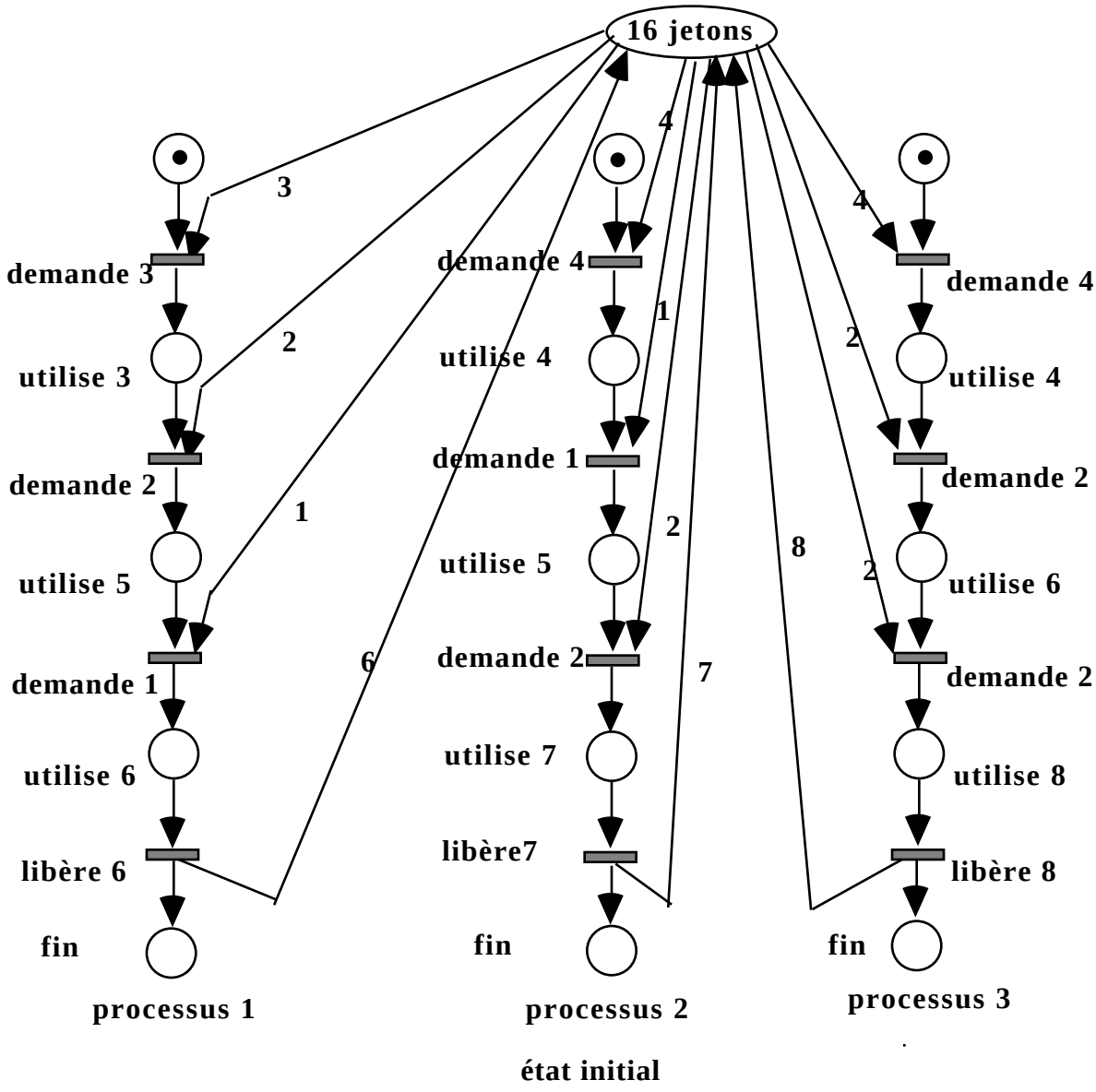
état initial



réseau non vivant : interblocage

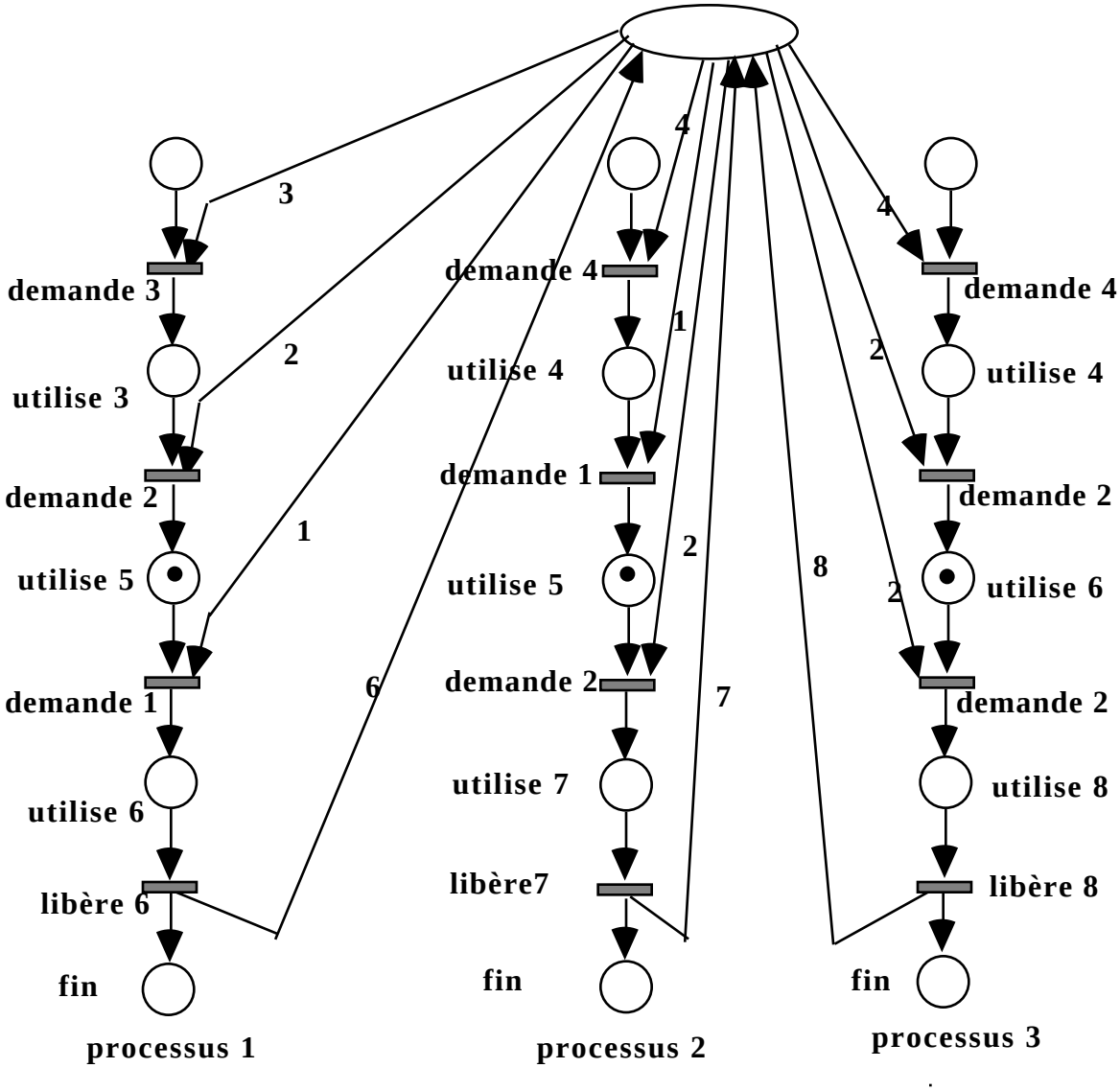
ALLOCATION DYNAMIQUE DE MÉMOIRE

modélisation par réseaux de Petri : interblocage = non vivacité
état initial



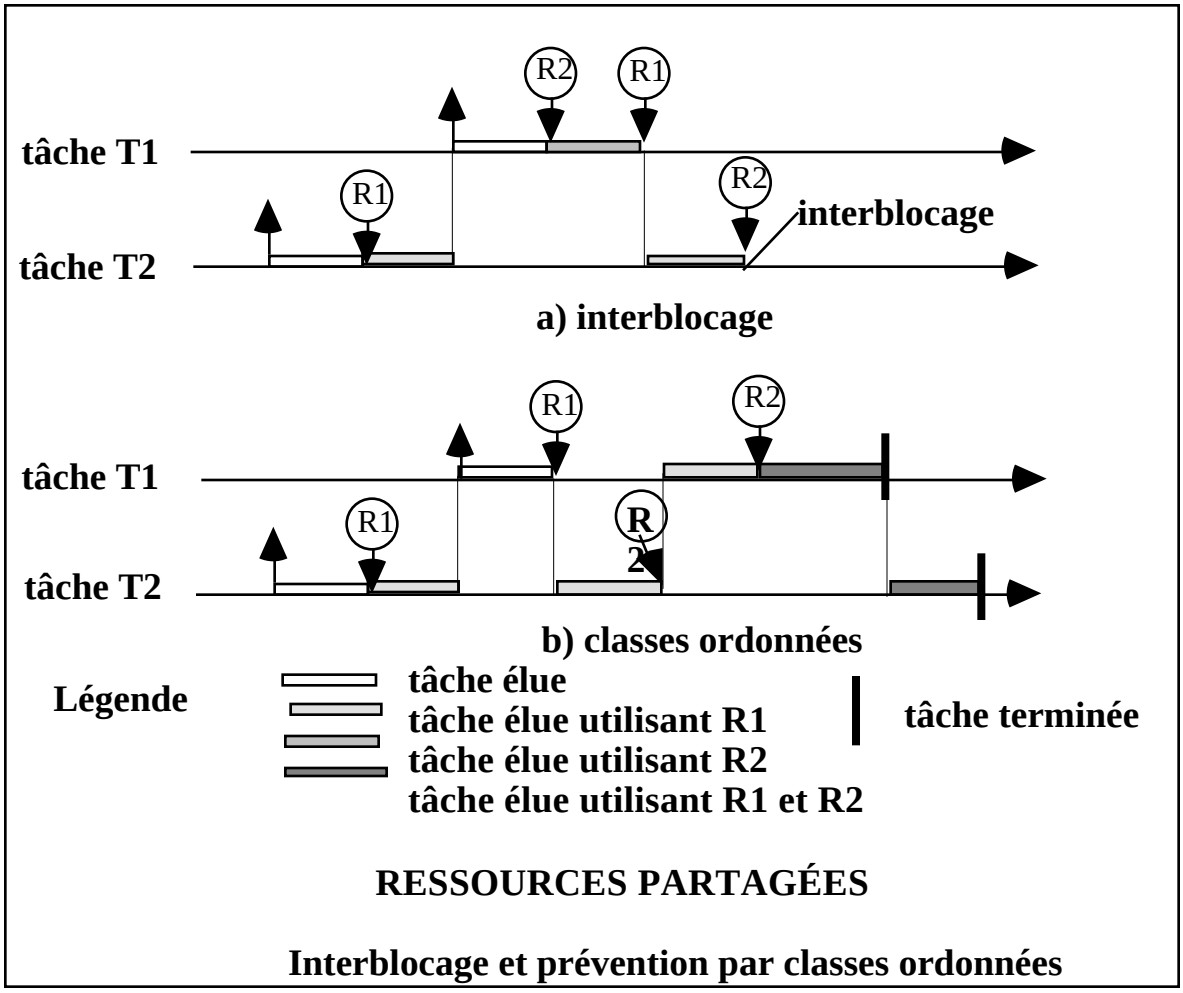
état initial

modélisation par réseaux de Petri : interblocage = non vivacité



marquage non vivant : interblocage

DIAGRAMME DE GANTT



RESSOURCES PARTAGÉES

Interblocage et prévention par classes ordonnées

EXERCICES POUR LE CHAPITRE 06

Exercice 1 (mai 1991). Un système est en interblocage quand:

a-les processus les moins prioritaires ne sont jamais activés,

b-les cases de mémoire attribuées précédemment aux processus prêts sont reprises par l'algorithme de remplacement pour charger des pages du processus élu,

c-des processus bloqués s'attendent circulairement,

d-les interruptions sont masquées et des processus sont en attente active.

Exercice 2 (juin 1991)

On dispose d'un certain nombre de ressources critiques nécessaires au fonctionnement de programmes telles que imprimante graphique, imprimante couleur, table traçante, modem,...

Chaque programme, lorsqu'il s'exécute, demande l'allocation des ressources qui lui sont nécessaires. Lorsqu'il a fini de s'exécuter, il libère ces ressources. Cela donne en particulier le cas suivant:

```
task body P1 is
begin
demander(table_traçante);
demander(modem);
...exécution...
libérer(modem);
libérer(table_traçante);
end P1;
```

```
task body P2 is
begin
demander(modem);
demander(imprimante);
...exécution...
libérer(imprimante);
libérer(modem);
end P2;
```

```
task body P3 is
begin
demander(imprimante);
demander(table_traçante);
...exécution...
libérer(table_traçante);
libérer(imprimante);
end P3;
```

Question 1. Cette solution n'est pas satisfaisante dans tous les cas; expliquer pourquoi.
Question 2. Proposer une solution ne posant plus de problème.

Exercice 3 (juin 1993)**.** Quatre processus P1, P2, P3 et P4 se partagent 50 granules sur un disque. Chacun garde les granules acquis et attend l'arrivée de sa nouvelle demande avant de continuer. Il n'y a pas de réquisition.
P1 demande successivement 1 granule, puis 4, puis 3, puis 6, puis 7, puis 1, puis rend les 22 granules acquis.
P2 demande successivement 4 granules, puis 3, puis 9, puis 5, puis 5, puis rend les 26 granules acquis.
P3 demande successivement 2 granules, puis 3, puis 4, puis 7, puis 2, puis 2, puis 1, puis rend les 21 granules acquis.
P4 demande successivement 7 granules, puis 2, puis 1, puis 2, puis 2, puis 9, puis rend les 23 granules acquis.
Question 1) :Montrer que l'état suivant correspond à un interblocage du système :

R(t1) = 6
A(t1) = (14, 7, 9, 14)
D(t1) = (21, 16, 16, 23)

Donner un autre exemple d'état d'interblocage du système.
Question 2) : On prend la valeur de 26 granules comme annonce maximale pour chacun des processus du système. En garantissant au processus le mieux pourvu en granules la possibilité d'atteindre cette valeur maximale, on peut simplifier la politique de prévention dynamique de l'interblocage. Rappeler cette politique et expliquer pourquoi cette simplification est valable.

En particulier, quand il applique cette politique(simplifiée), que doit répondre l'allocateur s'il est confronté dans l'état t3 aux demandes D(t3) :

R(t3) = 18
A(t3) = (14, 7, 3, 8)
D(t3) = (14, 16, 9, 16)

C - A(t3) = (12, 19, 23, 18) ?

L'allocateur peut-il satisfaire n'importe lequel des processus demandeurs (autre que P1) en étant assuré de ne jamais laisser le système aller vers un état d'interblocage? Expliquer votre réponse pour chaque processus.

Exercice 4 (septembre 90)**.** Soit un système qui gère un total de 48K de mémoire allouable, qui sert les requêtes des processus sans réquisition et qui bloque tout processus demandeur tant que sa requête ne peut être satisfaite. Trois processus PA, PB, Pc ont déclaré au préalable que leur demande maximale serait de 25K, 15K et 41K respectivement. A un moment de l'exécution, les 3 processus sont actifs tous les 3 et ont acquis 3K, 9K et 24K respectivement. Laquelle (ou lesquelles) des requêtes suivantes de mémoire supplémentaire peut elle être servie avec l'assurance qu'il n'en résultera jamais d'interblocage, une fois l'allocation faite:

a- PA demande 9K,
b- PC demande 7K,
c- PB demande 6K,
d- PA demande 6K.

Exercice 5 (février 1997)

Question 1) 5 processus se partagent 11 ressources d'une classe de ressources banalisées qu'ils demandent une à une (par exemple des granules de mémoire secondaire). Chaque processus demande au total au plus 3 ressources. Montrer qu'il ne peut y avoir interblocage. (piste : il y aurait interblocage si chaque processus avait reçu x ressources et en demandait encore plus et s'il n'y a plus assez de ressources pour permettre à un processus au moins d'obtenir trois ressources).

Question 2) On généralise cet exemple : N processus se partagent M ressources d'une classe de ressources banalisées qu'ils demandent une à une. La demande totale T de chaque processus ne peut pas dépasser M ressources et la somme des demandes totales de tous les processus N * T est plus petite que M + N. Montrer qu'il ne peut pas y avoir interblocage.

Question 3) Si 5 processus se partagent 6 ressources d'une classe de ressources banalisées qu'ils demandent une à une (par exemple des segments de mémoire principale) et si chaque processus demande au total au plus 2 ressources, il ne peut y avoir interblocage selon le résultat du 2 ci-dessus. Supposons maintenant que 5 processus se partagent 6 ressources d'une classe A de ressources banalisées qu'ils demandent une à une sans dépasser le total Ta = 2 et aussi 11 ressources d'une classe B de ressources banalisées qu'ils demandent une à une sans dépasser le total Tb = 3. Montrer que contrairement au cas où il n'y a qu'une classe de ressources, la présence de deux classes de ressources peut conduire à interblocage si chaque processus peut demander les ressources une à une dans une classe ou l'autre, sans dépasser les totaux Ta de A et Tb de B.

Question 4) Montrer qu'avec 7 ressources de classe A et 14 ressources de classe B, il n'y a pas d'interblocage; (pour vous mettre sur la piste de la solution : il en sera de même, et il n'y a pas interblocage, avec 6 ressources de classe A et 15 ressources de classe B; ou avec 8 ressources de classe A et 13 ressources de classe B; ou ...; ou avec 10 ressources de classe A et 11 ressources de classe B).

RÉPONSES AUX EXERCICES

Réponse à l'exercice 1 : c

Réponses à l'exercice 2

1) attente circulaire possible

2) P3 doit demander la table traçante avant l'imprimante

Réponses à l'exercice 3

1) il y a interblocage car R(t1) vaut 6 ce qui est inférieur à chacune des demandes des processus
autres exemples d'interblocage (parmi d'autres)

A( ) = (14, 21, 2, 12) D( ) = (21, 26, 5, 14) R( ) = 1
A( ) = (14, 16, 5, 14) D( ) = (21, 21, 9, 23) R( ) = 1
A( ) = (0, 21, 20, 9) D( ) = (1, 26, 21, 10) R( ) = 0
A( ) = (8, 16, 9, 14) D( ) = (14, 21, 16, 23) R( ) = 3
A( ) = (5, 21, 9, 14) D( ) = (8, 26, 16, 23) R( ) = 1

2) La solution consiste à appliquer l'algorithme du banquier qui se simplifie ici : il suffit de toujours garder à t, après allocation fiable ou refus, de quoi servir la demande maximale du processus qui a obtenu à t le plus de ressources (c'est aussi celui qui a la plus petite valeur pour C - Ai).

Donc pour déterminer les états fiables à partir de t3, on essaie de servir 1 processus et on applique la règle.

R(t3) = 18
A(t3) = (14, 7, 3, 8)
D(t3) = (14, 16, 9, 16)

C - A(t3) = (12, 19, 23, 18) ?

Servir P2 donnerait, (en lui allouant 9 granules)

A(t4) = (14, 16, 3, 8)
D(t4) = (14, 16, 9, 16)

C - A(t4) = (12, 10, 23, 18) R(t4) = 9 Aucun processus ne peut atteindre 26. On ne doit pas servir P2

Servir P3 donnerait, (en lui allouant 6 granules)

A(t4) = (14, 7, 9, 8)
D(t4) = (14, 16, 9, 16)

C - A(t4) = (12, 19, 17, 18) R(t4) = 12 P1 peut atteindre 26. On peut servir P3

Servir P4 donnerait , (en lui allouant 8 granules)

A(t4) = (14, 7, 3, 16)
D(t4) = (14, 16, 9, 16)

C - A(t4) = (12, 19, 23, 10) R(t4) = 10 P4 peut atteindre 26. On peut servir P4

Réponse à l'exercice 4 : A5.c

Réponses à l'exercice 5

1) au pire, chaque processus a obtenu déjà 2 ressources, ce qui a mobilisé 10 ressources. Le premier qui demandera sa troisième ressource pourra terminer et rendre une ressource au moins; cette ressource pourra permettre à un autre processus de terminer, ...

2) au pire, les N processus ont chacun T - 1 ressources; il doit en rester une pour que l'un après l'autre tous les processus puissent terminer, donc M = N(T - 1) + 1 ou encore
M + N = N * T + 1 ou encore M + N > N * T

3) soit le tableau donnant les allocations avec Ta = 2 et Tb = 3

processus	P1	P2	P3	P4	P5	reserve
ressource A	1	1	1	1	1	1
ressource B	2	2	2	2	2	1

Si P1 reçoit la dernière ressource A et P2 la dernière ressource B, on est en interblocage dès que tous les processus demandent une nouvelle ressource qu'elle soit de la classe A ou de la classe B.

4) Pour éviter l'interblocage, il faut que, dans le pire cas, la dernière ressource puisse permettre aux processus de finir l'un après l'autre. Il vient :

processus	P1	P2	P3	P4	P5	reserve
ressource A	2	2	1	1	1	0
ressource B	2	2	3	3	3	1