

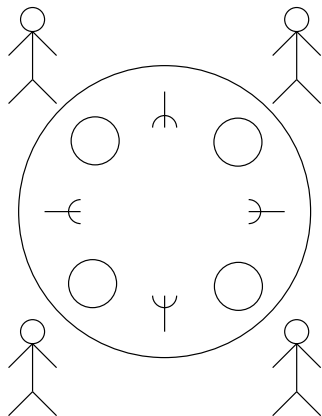
# Sur des stratégies locales de contrôle de famine

**Raymond Devillers**

Université Libre de Bruxelles, Belgium

MEFOSYLOMA — 2 Septembre 2009

# Problème des philosophes (Dijkstra)



**do** penser

protocole d'entrée (PE)  
manger (section critique)  
protocole de sortie (PS)

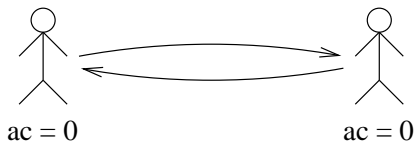
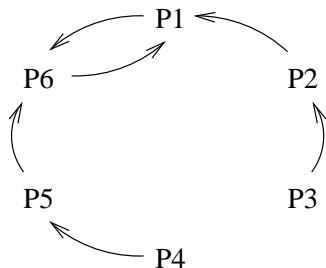
**od**

sources de famine :

1. gestion des files d'attente
2. interblocages
3. coalition ou hasard malencontreux

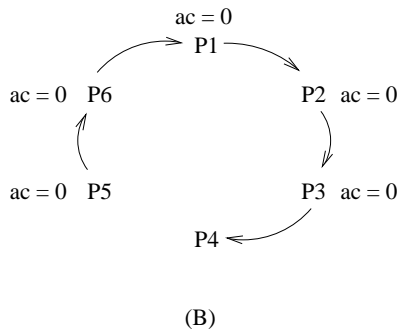
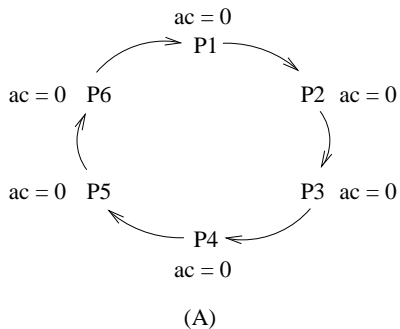
solution de Dijkstra :

1. compteur non négatif
2. initialisé (assez haut) lorsqu'on a faim
3. décrémenté lorsqu'on se fait dépasser



1. hypothèse : expédience
2. conséquence : on peut ne contrôler qu'un voisin, choisi statiquement ou dynamiquement
3. condition de sûreté : pas de cycle de contrôle effectif (compteurs nuls)
4. choix de la valeur initiale du compteur : pas de cycle aller-retour nul

5. et pour les cycles d'Euler ? Ils ne peuvent arriver !



hélas, il y a des problèmes !

1. il faut protéger les compteurs !

Par un sémaphore ? Non, car ce serait global !

Par des sémaphores (un par philosophe) ? OK mais comment y accéder ? Ce sont de nouvelles fourchettes !

Il existe une solution : la méthode hiérarchique de Havender.

Mais alors pourquoi ne pas l'avoir utilisée tout de suite ?

Cela revient à avoir à la fois des gauchers et des droitiers ; c'est simple mais rigide ! D'où l'intérêt de combiner.

2. comment implanter l'expédience ?

Quand un philosophe a faim : pas trop de problème.

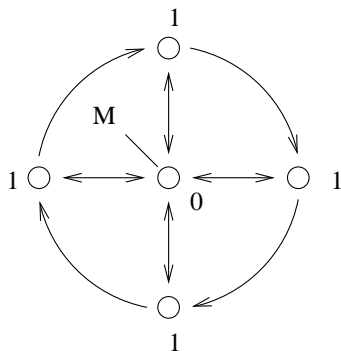
Quand un philosophe arrête de manger : il faut réveiller ses voisins si nécessaire ; mais ce n'est pas strictement local, ou c'est semi-actif, et c'est pas facile avec des réseaux de Petri !

### 3. Ça se généralise mal !

si on a  $n$  voisins, on peut en contrôler  $n - 1$

mais un choix de compteur nul peut produire des cycles non locaux

et surtout ça ne marche qu'au début



ça explique aussi pourquoi il était si important de ne contrôler qu'un voisin philosophe !

## ma solution (avec Lauer)

1. au lieu d'associer un compteur à chaque processus, on va en associer un par voisin !
2. on ne demande plus l'expédience, juste qu'un processus en attente avec des voisins penseurs ne va pas rester comme ça indéfiniment ;
3. l'absence de cycle de contrôle effectif redevient N&S ;
4. quand on a faim, on peut choisir des compteurs nuls si tous les compteurs qui nous sont associés chez nos voisins sont non nuls ;
5. il faut toujours protéger les compteurs par des sémaphores et Havender ;
6. c'est plus lourd, car il y a beaucoup plus de compteurs, mais c'est aussi plus souple !

P.J. Courtois and J. Georges : *On Starvation Prevention*. RAIRO Informatique 11 (2) : 127-141 (1977)

Raymond R. Devillers, Peter E. Lauer : *A General Mechanism for Avoiding Starvation with Distributed Control*. Inf. Process. Lett. 7(3) : 156-158 (1978)

E.W Dijkstra : *A Class of Allocation Strategies Inducing bounded Delays Only*. Proc. SJCC : 933-936 (1972)