Travaux Dirigés et Pratiques n°1

Algorithmes parallèles et répartis

—Élèves-Ingénieurs de 5-ème année—

Interblocage

L'objectif de cette feuille est de faire le point sur les problèmes d'interblocage (*deadlock*). Ce problème fondamental a donné lieu à la mise en place de plusieurs solutions générales. Nous voyons tout d'abord quelques méthodes préventives plutôt que curatives. Puis, nous abordons la notion d'état sûr et voyons l'algorithme dit du banquier.

Un ensemble de processus est en interblocage (*deadlock*) lorsque chaque processus de l'ensemble attend un évènement qui ne peut être causé que par un autre processus de l'ensemble.

▸ **Exercice 1.** **(notion d'interblocage)** *On considère un système possédant un ensemble de ressources regroupées par type : imprimantes, mémoires de masse, ..., toutes les ressources d'un type donné étant considérées comme équivalentes. En mode normal d'exécution, un processus ne peut utiliser qu'une ressource à la fois. L'obtention, l'utilisation puis la libération de cette ressource est guidée par la séquence suivante : demande-attente sur un type de ressource (ou sur un ensemble de ressources) ; utilisation de la ressource obtenue ; libération de la ressource. On considère par ailleurs les définitions suivantes :*

① *exclusion mutuelle : se produit lorsqu'un type de ressource au moins ne peut être obtenu que par un seul processus à la fois ;*

② *hold and wait : se produit lorsqu'il existe un processus qui détient au moins une ressource et qui en attend une autre détenue par un autre processus ;*

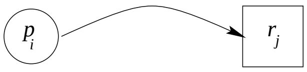
③ *pas de préemption : cette situation correspond à une ressource qui ne peut être libérée que par le processus à qui elle appartient, une fois sa tâche terminée ;*

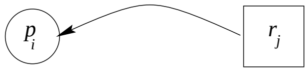
④ *attente circulaire : se produit lorsqu'il existe une suite $\{p_0, \ldots, p_n\}$ de processus en attente tels que, p_0 attend une ressource détenue par p_1, p_1 attend une ressource détenue par p_2, ..., et finalement p_n attend une ressource détenue par p_0.*

1. *Ces conditions sont-elles indépendantes ?*
2. *Montrer que s'il existe un interblocage sur l'utilisation des ressources alors toutes ces conditions doivent être satisfaites.*
3. *En déduire plusieurs façons de se prévenir d'un interblocage. En donner les applications et les limitations.*

4. *Ces conditions sont-elles nécessaires et suffisantes ? Donner éventuellement un exemple où toutes ces conditions sont remplies et où il n'y a pas d'interblocage.*

On représente un processus par un cercle et une ressource par un carré. On formalise alors la notion de demande de la ressource r_j par le processus p_i par le graphique de gauche ci-dessous, le graphique de droite signifiant au contraire que la ressource r_j est détenue par le processus p_i :





On définit par ailleurs les vecteurs d'information suivants :

① **Dispo**$[j] = k$: lorsque que k ressources de type j sont disponibles;

② **Max**$[i][j] = k$: lorsque le processus p_i peut demander au plus k instances de r_j ;

③ **Alloc**$[i][j] = k$: lorsque le processus p_i détient k instances de la ressource r_j ;

④ **Besoin**$[i][j] =$ **Max**$[i][j] -$ **Alloc**$[i][j]$.

Un état du système est caractérisé par le nombre de ressources disponibles, le nombre de ressources détenues par chaque processus et le nombre maximal de ressources nécessaires à chaque processus.

▸**Exercice 2.** (**notion d'état sûr**) *Un état est qualifié de sûr dès lors qu'il existe un ordre total sur l'ensemble des processus $p_{t_0} < p_{t_1} < \cdots < p_{t_n}$ tel que toute demande de ressource par un processus p_{t_i} peut être satisfaite en utilisant les ressources disponibles ou celles actuellement détenues par un processus p_{t_j} tel que $t_j < t_i$. Un état est qualifié de peu sûr s'il ne vérifie pas les conditions d'un état sûr.*

1. *Donner un exemple utilisant le formalisme précédent montrant un ensemble de processus dans un état peu sûr.*

2. *On considère un ensemble de processus dans un état peu sûr, mais pas dans un état d'interblocage. Est-il possible de toujours terminer l'exécution de l'ensemble de ces processus ? Le cas échéant, on donnera un exemple montrant que partant d'un état peu sûr on peut arriver à un état d'interblocage.*

3. *Montrer que tout état sûr ne peut jamais correspondre à une situation d'interblocage.*

4. *En déduire une idée permettant à un système de ne jamais se trouver en situation d'interblocage.*

▸**Exercice 3.** (**algorithme du banquier**) *On considère le vecteur* $\mathbf{R}[i]$ *de requêtes du processus p_i de sorte que si p_i demande k instances de la ressource r_j, alors* $\mathbf{R}[i][j] = k$. *Cet algorithme s'organise autour de trois étapes qui sont les suivantes :*

① *Si* $\mathbf{R}[i] \leq \mathbf{Besoin}[i]$ *Faire* ② *Sinon erreur("demande trop importante") FinSi ;*

② *Si* $\mathbf{R}[i] \leq \mathbf{Dispo}[i]$ *Faire* ③ *Sinon attendre FinSi ;*

③ *Si le nouvel état est sûr Faire* $\mathbf{Alloc}[i] = \mathbf{Alloc}[i] + \mathbf{R}[i]$, $\mathbf{Dispo}[i] = \mathbf{Dispo}[i] - \mathbf{R}[i]$, $\mathbf{Besoin}[i] = \mathbf{Besoin}[i] - \mathbf{R}[i]$ *FinSi.*

On considère l'état d'un système de 4 ressources partagées par 5 processus de la manière suivante au temps t :

	Alloc	**Max**	**Dispo**
p_1	[0012]	[0012]	[1520]
p_2	[1000]	[1750]	[1520]
p_3	[1354]	[2356]	[1520]
p_4	[0632]	[0652]	[1520]
p_5	[0014]	[0656]	[1520]

1. *Que contient au temps t le vecteur* **Besoin** *?*
2. *Le système est-il dans un état sûr ?*
3. *La demande* $\mathbf{R}[2] = [0420]$ *peut-elle être satisfaite immédiatement ?*
4. *Dans l'algorithme précédent, il faut savoir tester si un état du système est sûr. Donner un algorithme réalisant ce test.*
5. *Donner la complexité de l'algorithme du banquier.*

▸ **Exercice 4.** (**que choisir ?**) *Plutôt que de se maintenir dans un état sûr afin de ne pas tomber dans un état d'interblocage, on peut choisir de détecter périodiquement si le système se trouve dans un état d'interblocage.*

1. *Proposer un algorithme de détection d'une situation d'interblocage.*
2. *Quelles sont les limitations liées à la détection d'interblocage par rapport aux autres méthodes évitant les situations d'interblocage.*

▸ **Exercice 5.** (**cas particulier de l'algorithme du banquier**)

1. *Réduire l'algorithme du banquier au cas d'une seule instance de chaque ressource.*
2. *Quel est la complexité de cet algorithme ?*
3. *Montrer par un exemple que l'algorithme du banquier sur plusieurs ressources ne peut être implémenté par l'application de l'algorithme à chacune des ressources.*