TP Threads Java: Le dîner des philosophes

Le dîner des philosophes est un problème classique des systèmes distribués sur le partage de ressources communes entre plusieurs processus. Ce problème a été énoncé par Edsger Dijkstra.

Le problème se décrit ainsi (voir figure 1) :
- Cinq philosophes se trouvent autour d'une table ronde (il peut y en avoir plus).
- Chacun des philosophes a devant lui un plat de spaghetti.
- Chacun a une fourchette à sa gauche.

FIGURE 1 – Cinq philosophes à table devant leur plat de spaghetti.

Un philosophe n'a que trois états possibles :
- Penser pendant un temps indéterminé.
- Etre affamé.
- Etre en train de manger.

Règles :
- Pour manger, un philosophe a besoin de 2 fourchettes (celle à sa gauche et celle à sa droite).
- Chaque philosophe pense un certain temps (car il est philosophe) puis devient affamé (car il est un humain).
- Quand il est affamé, il faut qu'il mange un certain temps comme tout le monde.

Modélisation du problème. Créez un programme modélisant le problème utilisant des Threads pour simuler les philosophes et des objets partagés pour les fourchettes. Utilisez des techniques de synchronisation pour assurer qu'une même fourchette ne puisse être utilisée par deux philosophes en même temps.

— Faites en sorte que les philosophes pensent et mangent pendant des durées aléatoires entre 0 et 256 millisecondes.
— Affichez une trace écrite de ce qui se passe sur le terminal.

Dans un premier temps faites en sorte que chaque philosophe affamé prenne sa fourchette de gauche dès qu'elle est libre et ensuite celle de droite dès qu'elle est libre.

Ceci n'est pas une solution car il est possible que tous les philosophes soient affamés indéfiniment et donc de meurent (car ils sont humains). Il suffit que tous les philosophes prennent la fourchette à leur gauche en même temps et se retrouvent à attendre l'autre indéfiniment (*dead-lock*).

$1^{ière}$ **solution : Inviter des philosophes ambidextres.** Une solution consiste à changer l'ordre de sélection des fourchettes pour certains philosophes. Par exemple les philosophes pairs prennent d'abord la fourchette de droite puis celle de gauche. Testez cette solution.

Cette solution garantit qu'il ne puisse y avoir de *dead-lock*, dès qu'il y a un philosophe qui prend d'abord le couvert de droite et un qui prend d'abord le couvert de gauche. Il ne peut pas y avoir de famine (*starvation*) non plus.

Malheureusement, les philosophes sont sur une table ronde et n'ont donc pas de numéro d'ordre de position. Les philosophes connaissent juste leur nom (qui est unique) et celui de leurs voisins. [1]

$2^{ième}$ **solution : Fournir une ardoise.** Une autre solution consiste à synchroniser l'ensemble des philosophes. Pour cela on peut leur mettre un unique objet, tel une ardoise, sur lequel ils peuvent partager leur état actuel et ainsi décider (en fonction de l'état de leurs voisins) si ils peuvent récupérer les fourchettes et manger. Il faut aussi pour cela être capable de connaître l'état de ses voisins et de les réveiller si ils sont affamés après avoir reposé les couverts. Testez cette solution.

Cette solution empêche tous les philosophes de mourrir de famine mais il se peut que certains soit affamés indéfiniment car à chaque fois que l'un de ses voisins termine de manger, il se peut que son autre voisin soit en train de manger (*starvation*). Cette situation peut être évitée en ajoutant une file d'attente sur l'ardoise (un philosophe attend qu'aucun de ses voisins affamés ne soit prioritaire pour manger). Améliorer la solution précedente avec un tel mécanisme de priorité.

Cette solution demande une synchronisation globale de tous les philosophes. Non seulement cela leur demande beaucoup d'effort de synchronisation, mais en plus cela rend la solution très fragile en présence de malveillance de la part des philosophes. Par exemple si un philosophe ne tolère pas les idées d'un autre et veut l'empêcher de manger, il est possible qu'il y arrive même si ils ne sont pas voisins.

1. En réalité cette solution reste réalisable en choisissant en premier le couvert de gauche si on a un "plus petit" nom (selon l'ordre lexicographique par exemple) que ses voisins et en choisissant le couvert de droite si on a un "plus grand" nom que ses voisins. Elle peut par contre s'avérer peu efficace et injuste envers les différents philosophes.

$3^{ième}$ **solution : Communiquer avec ses voisins.** Une autre solution existe ne demandant aucun *verrou* ni ressource partagée commune à tous les philosophes. Il suffit de rajouter 4 états possible pour chaque couvert aux philosophes et de la communication entre voisins. Chaque philosophe considère un couvert comme à lui ou non et comme propre ou sale. Le principe de l'algorithme est alors le suivant :

- En arrivant à table, si un philosophe a un nom plus "petit" (selon l'ordre lexicographique par exemple) que son voisin, il se donne la propriété de la fourchette qu'ils ont en commun et la considère comme sale.
- Lorsqu'un philosophe veut manger, si il est propriétaire des couverts il mange, sinon il prévient ses voisins de son intention et attend que l'on lui en donne la propriété pour manger.
- Lorsqu'un philosophe observe que son voisin veut la fourchette en commun, si il n'est pas en train de manger et qu'elle est sale, il la lui donne. Sinon il attend de marquer le couvert comme sale pour la lui transmettre.
- Lorsqu'un philosophe fini de manger, il marque ses couverts comme sales.