Paradigme du jeton circulant

Introduction

L'objectif de ce TP est d'implémenter un algorithme distribué basé sur le paradigme du jeton circulant.

I. Création topologie

La première étape consiste à mettre en place une topologie en anneau car c'est sur cette topologie que se base le paradigme du jeton circulant.

On a donc créé une fonction *« create_ring »* qui peut créer cette topologie en anneau en fonction du nombre de nœud demandé en paramètre. Comme pour le TP, on utilisera la fonction *« MPI_Graph_create »* en ayant complété les tableaux des index et des arcs. On créera ici une topologie unidirectionnelle et on choisira par défaut un nombre de 10 nœuds. La topologie étant unidirectionnelle, cela implique que chaque nœud connait le nœud suivant mais pas le nœud précédent.

II. Algorithme de MISRA

A. Présentation

L'algorithme consiste à faire visiter les processus par un jeton circulant entre ceux-ci. On aura ici 2 jetons *PING* et *PONG* qui circuleront sur l'anneau, le deuxième étant là comme une sorte de jeton faisant office d'acquittement. Avec ces jetons, on pourra faire en sorte qu'un nœud soit le seul à pouvoir accéder à une ressource à un moment donné. Le jeton pouvant être perdu (dans le cas d'une topologie sans fil si un nœud est déconnecté ou bien s'en va), on simulera alors une perte de jeton à un moment aléatoire. Un nouveau jeton sera alors créé pour le remplacer.

Le mécanisme permettant de détecter la perte d'un jeton est possible grâce à l'algorithme de MISRA. En effet, si le deuxième jeton arrive sur un site alors que le premier n'est pas passé, on pourra directement conclure que le premier a été perdu, c'est à ce moment-là qu'on pourra alors générer un nouveau jeton.

Le code est disponible à l'adresse suivante ou dans le fichier *« TP2_RE51_profondeur_Vincent_MERAT.c »*. Le résultat de l'exécution de ce code est disponible à l'adresse suivante dans lequel les résultats ont été triés dans l'ordre chronologique.

B. Explications

Pour expliquer le fonctionnement de l'algorithme, on va décrire son processus. Le maître commence par envoyer les 2 jetons l'un à la suite de l'autre. Lorsqu'un nœud reçoit un jeton, il le passe à son voisin et ainsi de suite. De cette façon, un même jeton ne passe jamais 2 fois de suite dans un même nœud, l'autre jeton étant passé entre temps. Cependant, si l'un des jetons est perdu, il faut s'en rendre compte pour le générer à nouveau.

Comment se rendre compte qu'un jeton ait été perdu ?

Cela fonctionne avec le principe suivant : $nombre\ passage\ PING - nombre\ passage\ PONG = 0$. C'est un moyen simple mais efficace de s'assurer du bon fonctionnement du système.

Pour *simuler* un réseau sans fil, on a créé une fonction qui définit un nœud ainsi qu'un numéro de tour sur lequel le jeton sera perdu pour vérifier le bon fonctionnement logique de l'algorithme.

Conclusion

Pour résumer le principe de l'algorithme : on a un anneau unidirectionnel (dans lequel les messages circulent dans un sens). On a ici 2 jetons circulant donc dans le même sens. Lorsqu'un jeton arrive sur un nœud, il s'assure que l'autre jeton soit le dernier à être passé, si ce n'est pas le cas, c'est que celui-ci a alors été perdu.

Cet algorithme peut en effet être étendu pour manipuler plus de 2 jetons. En effet, il fonctionne à partir de 2 mais peut être extensible du moment que chaque jeton sache vérifier quel jeton est passé avant lui. Si l'on prend un exemple avec 5 jetons :

- Le jeton 2 vérifiera que le jeton 1 est bien passé
- Le jeton 3 vérifiera que le jeton 2 est bien passé
- Le jeton 4 vérifiera que le jeton 3 est bien passé
- Le jeton 5 vérifiera que le jeton 4 est bien passé
- Pour finir la boucle, le jeton 1 vérifiera que le jeton 5 est bien passé

Cet algorithme correspond enfin mieux à une topologie fixe comme dans une *infrastructure filaire*, en effet, avec une solution sans fil, les *devices* pourraient être déconnectés, ce qui engendrerait systématiquement une perte du jeton ainsi qu'un chamboulement dans l'ordre et la topologie de l'anneau.