8INF856
Programmation sur architectures parallèles
Devoir 1

(À remettre avant jeudi le 2 octobre 2014 à 13h00)

- *Vous pouvez travailler en équipe d'au plus trois étudiants.*
- *Vous serez évalué pour la rectitude et la qualité de vos réponses.*

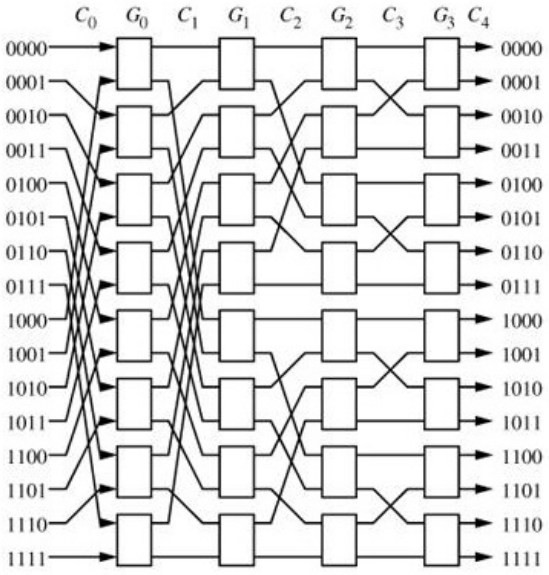

Figure 1: Un réseau Buterfly contenant 8 processeurs.

1. Nous avons vu en classe que les réseaux d'intercommunication à topologie indirecte sont construits à partir de commutateurs (switch) de deux

entrées et deux sorties. On dit d'un réseau qu'il est *complet* s'il permet de réaliser en parallèle toutes les communications qui sont des *permutations*. C'est-à-dire que les processeurs peuvent toujours s'envoyer (et recevoir) simultanément des messages tant qu'il n'y a pas deux d'entre eux qui ont le même destinataire à la condition que chacun d'eux ne soit le destinataire que d'un seul autre

Par exemple, considérez le réseau Butterfly de la figure 1 (source: Wikipedia).

(a) Montrez que ce réseau n'est pas complet.

(b) Montrez comment construire un réseau complet pour $n = 2^k$ processeurs ($k > 1$). Justifiez votre réponse.

2. Certains multi-ordinateurs permettent à des processus qui s'exécutent de migrer d'un noeud à l'autre. Est-il suffisant d'arrêter le processus, de figer son image mémoire et de l'expédier vers un noeud différents? Citez deux problèmes non négligeables qui doivent être résolus pour que cela fonctionne.

3. Les barrières sont des outils de synchronisation utiles mais on peut les remplacer par des variables conditionnelles. Expliquez comment.

4. Dans cet exercice pratique, vous devez écrire un petit programme multithread en C qui s'exécutera sur *dim-linuxmpi1*. Votre programme contiendra une fonction *fib(n)* qui retourne le n-ième nombre de Fibonacci.

   Le thread principal de votre programme devra d'abord lire sur l'entrée standard les entiers positifs n et k. Il devra ensuite créer un tableau TAB de n entiers de type *int* générés de façon aléatoire. Le programme devra afficher les 5 premiers et les 5 derniers nombres du tableau. Vous devrez ensuite créer k threads qui se partageront le travail suivant: il faut remplacer TAB[i] par fib(TAB[i]) pour tous les indices i entre 0 et $n-1$. Finalement, le thread principal doit afficher les 5 premiers et les 5 derniers étéments du tableau.

   Utilisez la fonction clock() pour déterminer le temps d'éxécution entre les deux affichages et exécutez votre programme sur différentes valeurs de n et k (petites, moyennes et grandes). Donner un tableau de vos résultats et faites en l'analyse.