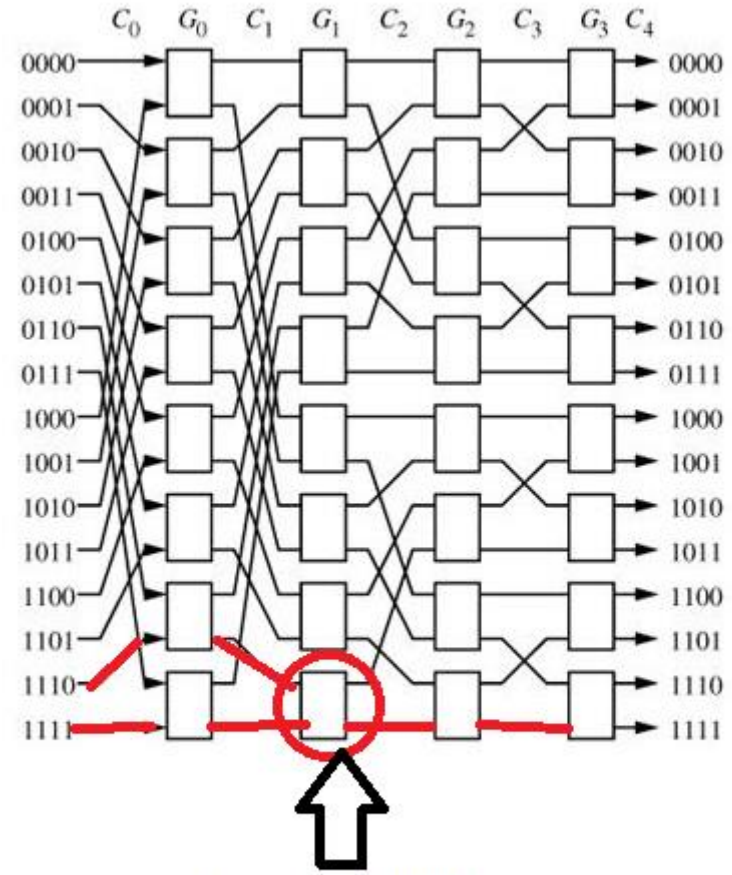
8INF856
Programmation sur architectures parallèles
Devoir 1

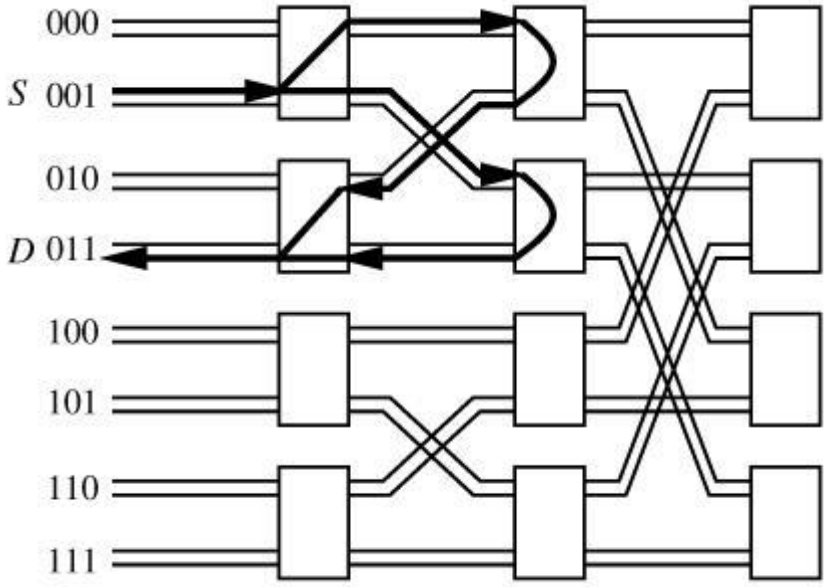
Adrien Cambillau
Corentin Raoult

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



Comme on peut le voir sur l'image ci-dessus, si le processeur 1111 envoie un message au processeur 1110 et que 1110 envoie un message à 1111, ces messages arriveront au même moment sur le deuxième switch, provoquant une collision.

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



Il suffit de rendre le réseau bidirectionnel, cela permet aux messages de revenir en arrière, créant de cette manière des chemins alternatifs permettant d'éviter des conflits.

Une autre solution possible serait un hypercube, étant donné que si il y a n bits différents entre l'adresse de départ et celle d'arrivée, alors il y aura n! plus courts chemins, cela rend l'hypercube moins sensible aux collisions.

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

Un des problèmes est lié à la migration d'un processus d'un nœud à l'autre est que pendant le temps de la migration les autres processus ne peuvent pas utiliser la mémoire qui est figée par le processus migrant. Cela a pour effet de bloquer tous les processus utilisant cette zone de mémoire.

Un autre problème est lié à l'adresse mémoire du processus qui va changer lors de la migration, ce qui empêchera les autres nœuds de retrouver le processus.

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

Les barrières sont des outils de synchronisation permettant de bloquer tous les processus tant qu'ils n'ont pas tous atteint la barrière. Avec une variable conditionnelle on peut faire la même chose avec des threads. En effet plusieurs threads se partage une variable P qu'ils incrémentent à tour de rôle. On peut dire aux threads de se bloquer tant que P n'aura pas atteint une certaine valeur qui est égale au nombre de threads. Ce qui veut dire que tant que tous les threads n'ont pas tous incrémenté P ils seront tous bloqués.

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.

Id : 8inf856-16, mdp : HvfiLx
pour compiler : gcc Fibonacci.c -o Fibonacci -lpthread -lrt
-ltr car on utilise pas clock() mais clock_gettime(), les résultats obtenus avec clock() n'étaient pas cohérents.

On observe que le multithreading n'est utile que lorsqu'on a un grand nombre de traitements à réaliser. De plus avoir un nombre de thread trop important n'a que peu d'influence au-dessus de 8 threads. Cela doit être lié au nombre de cœurs disponibles pour exécuter le programme.

Ci-dessous les résultats obtenus pour différentes valeurs de n et k.

n	k	temps d'exécution
----------taille du TAB = 10----------		
10	1024	0.059020
10	512	0.027283
10	256	0.013671
10	128	0.007159
10	64	0.003864
10	32	0.002054
10	16	0.001479
10	8	0.000489
10	4	0.000169
10	2	0.000053
10	1	0.000047
----------taille du TAB = 100----------		
100	1024	0.055156
100	512	0.026845
100	256	0.013251
100	128	0.006835
100	64	0.003751

100 32 0.001880

100 16 0.001358

100 8 0.000450

100 4 0.000159

100 2 0.000059

100 1 0.000064

----------taille du TAB = 1000----------

1000 1024 0.054086

1000 512 0.026639

1000 256 0.012902

1000 128 0.006644

1000 64 0.003475

1000 32 0.002126

1000 16 0.001298

1000 8 0.000457

1000 4 0.000164

1000 2 0.000184

1000 1 0.000180

----------taille du TAB = 10000----------

10000 1024 0.050700

10000 512 0.023418

10000 256 0.011656

10000 128 0.007124

10000 64 0.004048

10000 32 0.002408

10000 16 0.001460

10000 8 0.000527

10000 4 0.000479

10000 2 0.000751

10000 1 0.001440

----------taille du TAB = 100000----------

100000 1024 0.060004

100000 512 0.026716

100000 256 0.013399

100000 128 0.006797

100000 64 0.004266

100000 32 0.002549

100000 16 0.002765

100000 8 0.001984

100000 4 0.003473

100000 2 0.006719

100000 1 0.013325

----------taille du TAB = 1000000----------

1000000 1024 0.052198

1000000 512 0.036943

1000000 256 0.026739

1000000 128 0.022214

1000000 64 0.018900

1000000 32 0.018193

1000000 16 0.017340

1000000 8 0.016915

1000000 4 0.037286

1000000 2 0.069832

1000000 1 0.134606

----------taille du TAB = 10000000----------

10000000	1024	0.194456
10000000	512	0.175251
10000000	256	0.173761
10000000	128	0.169779
10000000	64	0.168110
10000000	32	0.168096
10000000	16	0.166598
10000000	8	0.171229
10000000	4	0.331495
10000000	2	0.661568
10000000	1	1.321132

----------taille du TAB = 100000000----------

100000000	1024	1.673872
100000000	512	1.663561
100000000	256	1.661759
100000000	128	1.663572
100000000	64	1.658200
100000000	32	1.657812
100000000	16	1.658693
100000000	8	1.660259
100000000	4	3.310714
100000000	2	6.606101
100000000	1	13.210793