Chapitre 3 : Réseaux d'interconnexion

Définition

- Ordinateur parallèle = processeurs + mémoires + **réseau d'interconnexion**
- Ce réseau permet la communication entre les unités de traitement.
- $\Longrightarrow$ besoin d'interconnexion rapide et fiable.

Topologie

- Pas possible d'interconnecter tout les processeurs (prix)
- **Topologie** = structure des liens entre les processeurs
- Processeurs = adjacents si reliés par une connexion directe
- Processeurs = distants si nécessaire de passer par des procs intermédiaires
- Le ràseau peut être vu comme un graph ou :

  — noeud = processeur ou mémoire

  — arc = connexion (bidirectionelle ou non)

- Topologie statique = connexions fixes
- Topologie dynamique = possibilité de reconfigurer les connexions

Performances d'un réseau

Diamètre D = longueure du chemin le plus court entre les 2 noeuds les plus éloignés

Connectivité Nombre de noeuds adjacents à chaque noeuds (= **Degré**)

Latence Temps nécessaire à un message pour transiter entre les 2 noeuds les plus éloignés. Dépends du diamètre et de la technologie utilisé (+ préparation du message et correction d'erreurs)

Bandwidth Débit d'information qu'un noed peut transmettre dans le réseau (généralement MByte/s)

Largeur bisectionnelle Nombre d'interconnexions entre deux moitiés identiques d'un réseau (fig 3.1 p 58). Si il existe plusieurs façon de diviser le réseau on prends la plus défavorable.

Scalabilité Possibilité d'augmenter la taille du réseau en ajoutant des processeurs

Prix Nombre decomposants en fonction du nombre de noeuds

Fonctionnalité Capacité de combiner des messages allant vers une même destination et gestion des conflits de routage

Capacité Nombre de messages mximum simultanés dans le réseau

Le routage

- **Routage** = algorithme de transite d'un message le plus efficacement possible entre les noeuds dans un réseau donné
- Routage minimal = chemins les plus courts entre la source et la destination
- Routeur = circuit chargé du routage
- L'algorithme de routage doit éviter :
  - Deadlock = message bloqué de façon indéfinie
  - Livelock = message tournant en boucle sans arriver à destination
- Single-port (*processor bound*) = communication n'utilisant qu'un seul canal à la fois
- Multi-port (*link bound*) = communication utilisant plusieurs canaux simultanément
- Routage local = les noeuds acheminnent les message uniquement en se basant sur une information locale

  - — Routage local déterministe = information locale = adresse de destination du message
  - — Routage local adaptif = information sur le reste du réseau (chemins de taille similaire, pannes)

- Routage synchrone = messages envoyés simultanéments et sousmis à un cycle de routage complet (temps de transfert = celui du message le plus lent)
- Routage asynchrone = messages envoyés et reçu en fonctions des besoins du programme

Primitives de communication

- Point à point (one to one)
- Permutation
- Broadcast (one to all)
- Réduction (many to one)
- Echange total (all to all)
- Distribution (one to all personalisé) = envoi de message différents à tout les noeuds (par ex Scatterv en MPI)
- Gather = inverse de Scatter (all to one personalisé)
- Echange total personalisé = tout les processeurs envoient un message différents à tout les autres processeurs
- Scans = calcule de la donnée finale en fonction des données précédents (noeuds 1 à $i-1$)

Technique de dommunication

- **Store and forward** = envoi du message du noeud $i-1$ au noeud i (stockage dans i), ensuite envoi de i à $i+1$
  - — Temps de transfert du message proportionel à la bandwidth, la longueur du

message et le nombre de noeuds + temps de préparation du message

— $t_{sf} = t_s + L(n-1)\frac{1}{W}$

— Pas rapide (n'exploite pas la puissance du réseau)

- **Cut through** = Les messages peuvent continuer sans être reçu complètement par un noeud
- **Wormhole** est un exemple de Cut through

  — Découpe du message en morceaux de taille fixe (flit)

  — Seul le premier flit contient l'adresse

  — Tout les flits suivants suivent le premier (comme des wagons avec une locomotive)

  — $t_{wh} = t_s + (n-1+L-1)\frac{1}{W}$

  — Beaucoup plus efficace que Store and forward

- Canaux virtuels = mutiplexage en temps du canal physique pour éviter les deadlocks

Réseaux statiques

Réseaux avec connexions physiques fixes entre les noeuds $\implies$ impossible de modifier la topologie.

Anneaux et chaînes

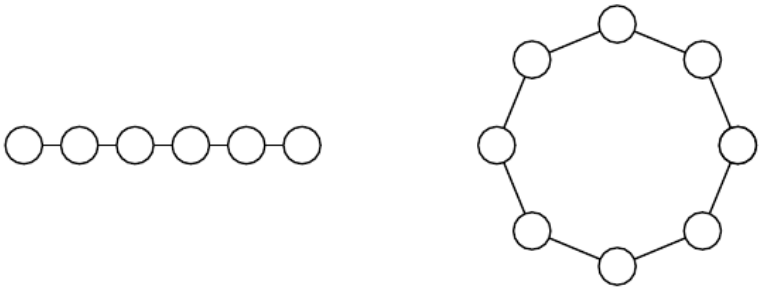

Figure 1 – Topologie Linéaire et en Anneau

- Adapté au traitement de problèmes unidimensionnels
- Diamètre : $D = N/2$
- largeur bisectionnelle = 2
- Prix = $\sim N$

Grilles de processeurs (mesh)

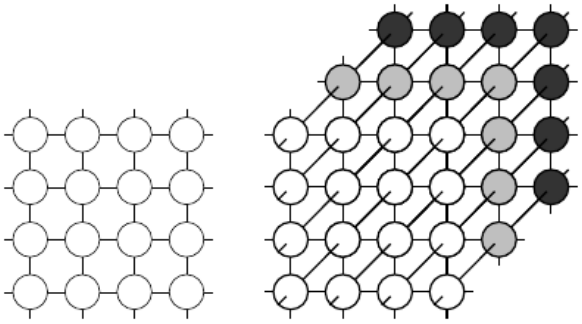

Figure 2 – Topologie en grille (2D et 3D)

- Mesh 2D plus répandus (ILLIAC, IV, DAP, Paragon)
- Mesh 3D : T3E, Parsytec
- Diamètre : $D = dN^{1/d}$ avec d la dimension

- Si les bords sont reliés $D = \frac{1}{2}dN^{1/d}$
- Le prix est proportionel à N
- Le routage s'effectue le long des axes de la grille
- Risque de congestion avec un routage X-Y
- Broadcast par ligne et ensuite colones par exemple $O(\sqrt{N})$ (fig 3.8)

hypercube

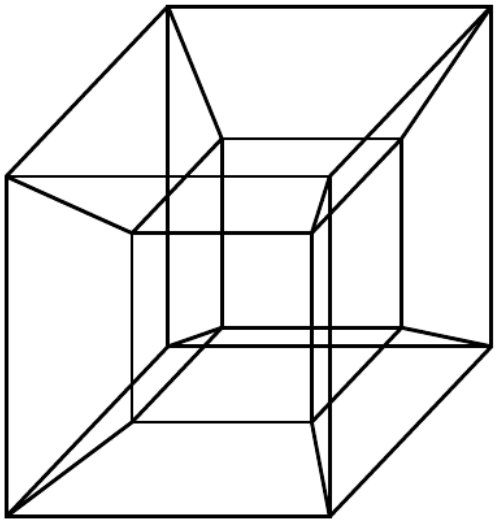

Figure 3 – Topologie hypercube de degré 4

- Généralisation du cube dans une dimension supérieur à 3
- Pour k dimensions et des bords de longueur 2 on aura 2^k noeuds
- Chaque noeuds est de degré k (k voisins)
- Diamètre $D = k$
- Le nombre de chemins entre deux noeuds augmente avec la taille de l'hypercude $\implies$ moins de risques de congestion
- Routage dans un réseau hypercubique :

— Numérotation des noeuds en binaire (k bits)

— Les noeuds reliés le long de la dimension l diffèrent que du l ième bit

— Pour aller de entre 2 noeuds (adresse A) on fait $A_1 \oplus A_2$ pour obtenir le vecteur

- Broadcast en hypercube : Connexion multiport (one to many), la racine envoi dans toute les dimensions i puis chaque noeuds envois dans les dimensions inférieurs à i
- Echange total hypercubique : Chaque noeud envois ses données plus toutes celles reçue précédément dans la dimension l le tout $N - 1$ fois
- Pour mapper une grille sur un hypercube on utilise la fonction de Gray

Arbres binaires

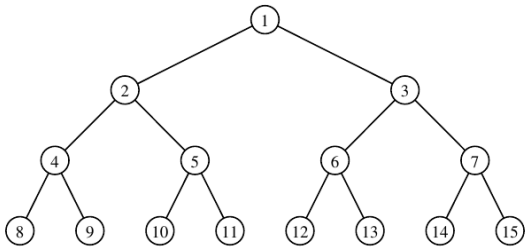

Figure 4 – Topologie Arbre binaire

- Topologie relativement peu utilisée dans sa forme statique
- Divisé en k niveaux (l) avec 2^l noeuds
- Nombre de noeuds = $2^{k+1} - 1$
- Diamètre : $D = 2k$
- degré des noeuds = 3 sauf aux extrémité
- Largeure bisectionnelle = 1 $\implies$ bottleneck
- Scallable proportionellement à N

- Routage : en représentant les numéros de noeuds sous forme binaire on remonte au parent commun (prefix binaire similaire)

Réseaux de permutation et “shuffle exchange”

- Utilisation de fonctions de permutations bijectives plutôt que des représentations géométriques
- 2 noeuds i et j sont reliés si $f(i) = j$
- **Perfect shuffle** : décalage des bits vers la droite ($010 \rightarrow 001$)
- **Inverse shuffle** : décalage des bits vers la gauche ($010 \rightarrow 100$)
- **Butterfly** : échange du premier et dernier bit ($100 \rightarrow 001$)
- **Bit reversal** : inversion de l'ordre des bits ($0100 \rightarrow 0010$)
- **Exchange** : remplacement du bit de poid faible par son complément à 1 ($0111 \rightarrow 0110$)
- Important de combiner deux opération si pas cyclique
- "Shuffle exchange" = combinaison de shuffle et exchange (no shit)

  — Degré = 3

  — Prix = $3N$

  — Scalabilité compliqué car si $N++$ les connexions changent

  — Largeur bisectionnelle = $N/2$

  — Diamètre : $D = 2log_2(N) - 1$

  — Routage : on "déplace" les bits avec l'opération de shuffle ($s()$) et on les corrige avec l'exchange ($e()$). Par exemple, pour aller de 0010 à 1100 on fait $e(0010) \rightarrow s(0011) \rightarrow s(1001) \rightarrow 1100$

Réseaux totalement connectés et réseaux aléatoires

- Réseaux totalement connectés pas envisagebale mais théoriquement intéressant

— Degré = $N-1 \implies \sim N^2$ connexions

— Diamètre : $D = 1$

— Largeur bisectionnelle = $N^2/4$

Réseaux dynamiques

Certains problèmes sont plus ou moins adaptés certains type de topologie. Un réseaux dynamique permet donc un gain de polyvalence pour une machine à but général. Cela implique une pénalité de prix et de complexité.
Dans l'ordre croissant de complexité et prix on a :

- Les connexions par un bus
- Les réseaux de commutateurs multiétages et arbres dynamiques
- Les réseaux crossbar

Connexions par bus

Arbres binaires

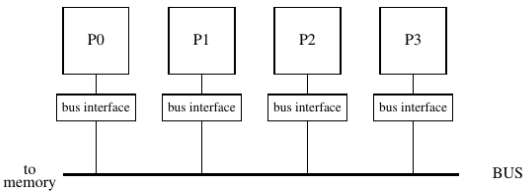

Figure 5 – Connexion par bus

- Proche d'un ordinateur séquentiel
- Utilisé dans plusieurs multiprocesseurs symétriques
- Limité à des systèmes avec peu de processeurs
- En général on a :

  - Un bus de données
  - Un bus d'adressage
  - Un bus d'arbitrage (gestion des conflits)
  - Un bus de synchronisation et d'interruptions

- Il faut concidérer le temps de latence plutot que de diamètre du réseau
- La latence ne dépends presque pas de N
- La Bandwidth est inversemment proportionelle à N
- Bandwidth Limitée par la réquence du bus et la largeur du chemin.
- Arbitrage d'un bus :

  - Priorité fixe (daisy chaining) : l'unité avec la priorité la plus haute l'emporte (possibilité de starvation)
  - Découpage en tranche : interval de temps
  - Priorité dynamique : même principe que le daisy chaining mais avec possibilité de changer la priorité au cours du temps (par ex premier arrivé, premier servi)

Crossbar switch

Un crossbar switch est une grille dont chaque point d'intersection est un commutateur qui permet de relier entre eux deux éléments.

- Possibilité de réaliser une topologie en étoile
- On peut concidérer un crossbar comme un commu $N \times N$ permettant $N!$ états différents de communication one to one
- Possible d'étalire N chemins simultanés tant que les élements de destinations sont différents
- Temps de latence dépends des commu et non de N

- Coup de N^2 pour les commutateurs et N pour les fils

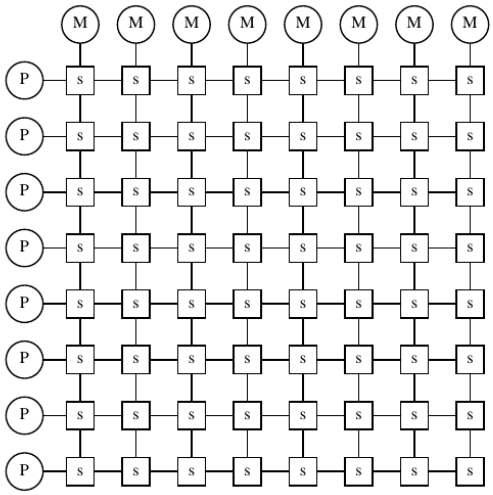

Figure 6 – Schéma d'un crossbar 8×8. Les éléments notés s apparaissant aux points de croisement sont les switches. Cet exemple montre une situation où 8 processeurs sont reliés à 8 mémoires.

Réseaux multiétages

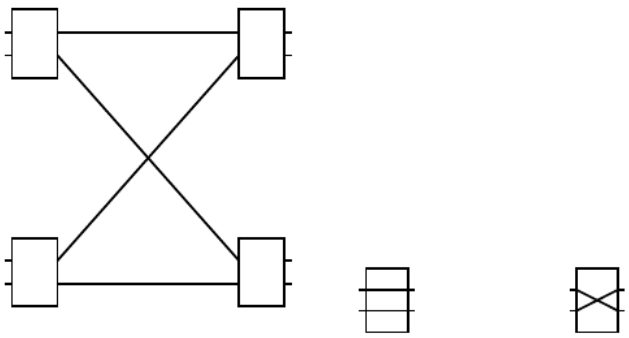

Figure 7 – Réseau multiétage 4×4

- Similaire au crossbar mais avec un essemble de commutateurs 2×2 plutôt que $N \times N$
- une combinaison entrée sortie contraint les autres combinaisons $\Longrightarrow$ toutes les permutations d'entrée/sortie ne sont pas possibles.

Réseaux Oméga

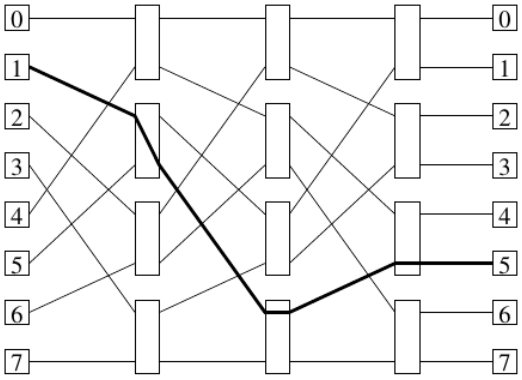

Figure 8 – Réseau multiétage oméga 8 entrées

- Basé sur une topologie "perfect shuffle"
- On a $log_2 N$ colones $N/2$ commutateurs 2×2 avec duplification
- Cout de $N log_2 N$
- possibilité de Broadcast (duplifiage sur toute les couches)
- Routage : Adresse en binaire de $log_2 N$ bits, chaque bite correspond à une couche, si 0 sortie haut si 1 sortie bas
- chemins uniques $\Longrightarrow$ possibilité de blocage

Fat-tree

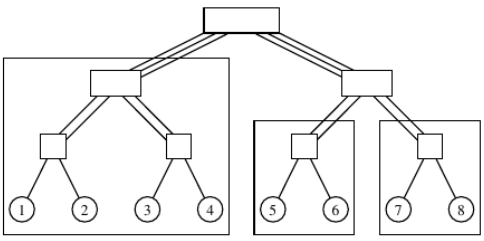

Figure 9 – Réseau Fat Tree

Réseau en arbre ou les processeurs se situent aux extrémités et les noeuds sont des switches

- l'épaisseur augmente plus on se raproche de la racine (plus de transit de message)
- Latence = $2log_2(N) - 1$
- Routage similaire à un arbre binaire static (avec équilibrage des charges sur les sections redondantes)
- Largeur bisectionnelle $log_2 N$
- A remplacé petit à petit l'hypercube dans les machines fortement parallèles
- Bien adapté aux architectures à mémoire distribuée