Jean-luc.dekeyser@lifl.fr
Version 2013

MODÈLE DE COMMUNICATION

Réseaux d'interconnexion

- **Unité primordiale dans les machines parallèles**
- **Le réseau permet de relier les processeurs à la mémoire ou les processeurs entre eux.**

Dynamicité

- **Les réseaux sont soit:**
  - **Statiques: les liaisons entre les composants sont établies une fois pour toute.**
  - **Dynamiques: Les liaisons peuvent changer dans le temps. On utilise souvent dans ce cas un algorithme de routage.**

Synchronisation

- **Le fonctionnement de ce réseau est**
  - **Synchrone: Une horloge globale rythme alors les échanges d'information (SIMD).**
  - **Asynchrone: Chaque noeud possède alors son propre référentiel de temps (MIMD).**

Accessibilité

- **Le type de communication est variable:**
  - **Complet: Tous les points du réseaux peuvent être reliés à chaque top d'horloge. (avec une éventuelle réorganisation des connexions existantes)**
  - **Incomplet: Seuls certains noeuds sont accessibles depuis chaque noeud du réseau. (on doit traverser le réseau plusieurs fois dans certains cas)**

Fonctionnalité

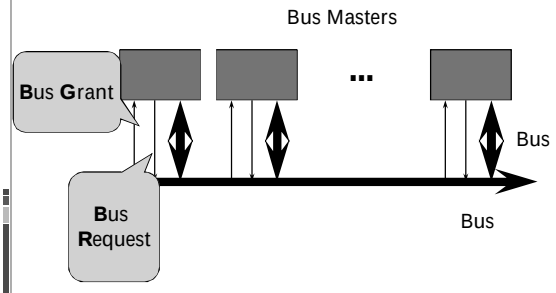
- **Le réseau établi une liaison entre un port d'entré et un port de sortie.**
- **Il permet ainsi un échange d'informations entre ces ports.**

SYSTÈMES MONO-BUS

Connexion par mono-BUS

- **Par extension du modèle de Von Neumann, on peut simplement réaliser un multi-processeur en connectant plusieurs CPU sur un bus.**
- **Cet organe de communication est alors partagé dans le temps pour réaliser les transfert**
  soit mémoire => CPU
  soit CPU => CPU.

Bus Time-shared



Efficacité

- **Il y a augmentation de la vitesse du système si et seulement si la somme des transferts égale le débit du BUS**
- **Exemple:**
  1 CPU utilisant 50% de son temps le BUS
  =>Maxi de 2 CPU sur le même BUS
  (Rôle des mémoires caches)

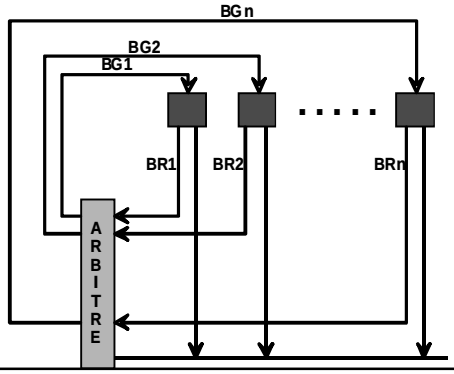
Requêtes multiples

- **Le maître reçoit plusieurs requêtes de différents CPU. Lequel choisir ??**
- **On introduit une notion de priorité - statique ou dynamique - qui permet d'arbitrer entre les demandeurs.**

Arbitrage

- **Méthodes d'arbitrage sont**
  - **Parallèles: Les BR et BG entrent et sortent en même temps de l'arbitre**
  - **Séries: Un signal est véhiculé d'un maître à l'autre jusqu'à obtention du maître le plus prioritaire demandeur du bus (daisy chaining)**
  - **Centralisées: Un système unique de résolution.**
  - **Décentralisées: réparti sur les CPU**

Priorité parallèle Centralisée



13

Priorité Parallèle Centralisée(2)

- **Chaque CPU peut générer un BR vers l'arbitre.**
- **Lorsque l'un est choisi par l'arbitre, un unique BG est envoyé vers l'élu.**
- **BB permet de libérer le bus en fin de transfert**
- **Un processeur peut utiliser le bus quand il a reçu un BG et que le bus est libre. Il maintient le signal BR.**

14

Gestion de l'arbitrage

- **Une demande plus prioritaire est prise en compte par l'arbitre: par désactivation du BG courant et activation du nouveau BG.**
- **Le maître courant du bus observe la désactivation de son BG, il peut donc libérer le bus en fin de communication.**
- **On évite ainsi l'interruption due à une requête d'un CPU plus prioritaire.**

15

Priorité parallèle décentralisée

- **1 arbitre dans chaque CPU**
- **Les requêtes BR sont envoyées à tous les CPU**
- **L'arbitre local envoie 1 Grant à son CPU si celui-ci est élu.**

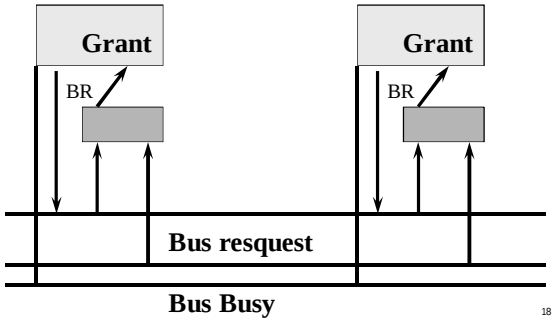
16

Parallèle décentralisée(suite)

- **Plus fiable (panne d'un arbitre panne d'un seul cpu)**
- **Nécessite un Time-out pour libérer le bus en cas de panne**
- **Bus moins large, pas de BG**
- **Réalisation avec des portes logiques**

17

Arbitrage parallèle décentralisé



18

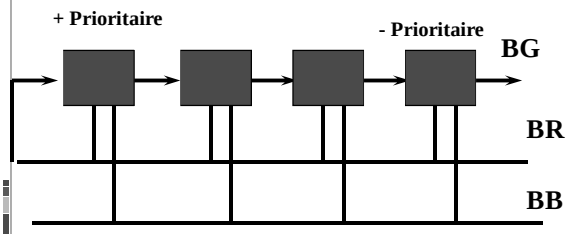
Priorité Série

- **Passage d'un signal d'un cpu vers un autre séquentiellement**

  " Daisy chain"

19

Sur le signal Grant



20

Sur le signal Grant

- **Le plus répandu**
- **1 ligne de requête partagée**
- **1 ligne de "Grant" du premier CPU vers les suivants**
- **1 ligne Busy**

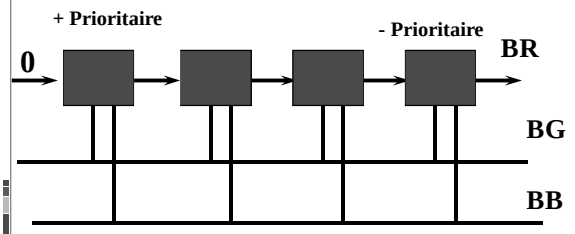
21

Fonctionnement

- **Lorsqu'au moins un BR est activé, le BG est validé par le contrôleur ou le CPU le plus prioritaire.**
- **Le BG est transmis jusqu'au premier CPU ayant émis un BR.**
- **Celui-ci prend alors le bus après le BB inactif.**

22

Sur le signal Request



23

Sur le signal Request

- **Un CPU demande le bus par émission d'un BR vers la droite.**
- **Lorsqu'un CPU reçoit un BR à gauche, il le transmet à droite s'il n'est pas maître du BUS, il libère le bus par un BG sinon (car moins prioritaire que le demandeur).**
- **Le CPU qui émet un BR et qui n'en reçoit pas à gauche devient alors maître du bus. (cf 8086)**

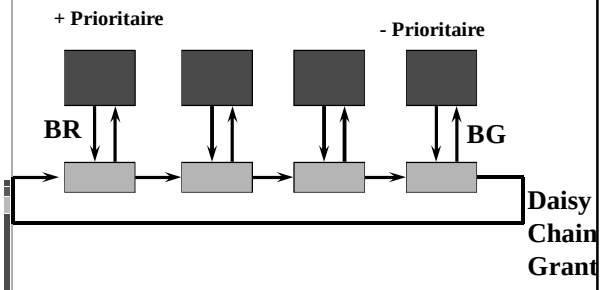
24

Priorité série décentralisée

- Sur signal Grant
- Anneau de CPU
- Le signal Grant circule sur l'anneau jusqu'à la prise du bus par 1 CPU
- Priorité circulante
  - Réseau à jeton
- Panne d'un CPU => Panne du réseau

25

Priorité série décentralisée



26

Arbitrage par scrutation:

- Un contrôleur recherche un CPU demandeur du Bus.
- Centralisé ou décentralisé

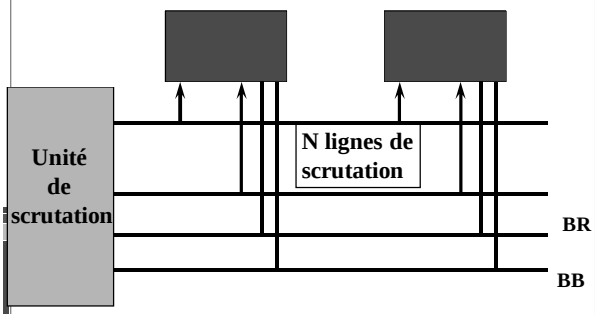
27

Centralisé

- Le contrôleur scrute tous les CPUs successivement (ordre = priorité)
- Le CPU demandeur émet un BR.
- Le bus est alors attribué à ce cpu.
  - Le nombre de lignes de scrutation peut être limité par numérotation des cpu.
  - Priorité statique: un compteur dans le contrôleur, ou dynamique: pseudo-random etc.

28

Scrutation centralisée



29

Décentralisé

- 1 contrôleur/CPU (1 décodeur d'adresse et 1 générateur d'adresse)
- 1 adresse est produite, elle est reconnue par le décodeur
- Si le cpu correspondant est demandeur il prend le bus

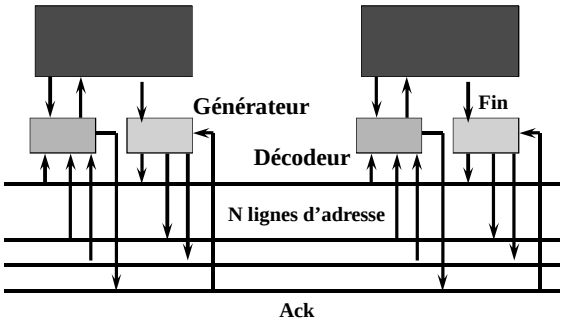
30

(suite)

- **Le décodeur transmet le BR si celui-ci ne décode pas le numéro voulu.**
- **A la fin le générateur d'adresse produit les numéros des cpu suivants si un BR arrive.**

31

Scrutation décentralisée



32

CROSSBAR

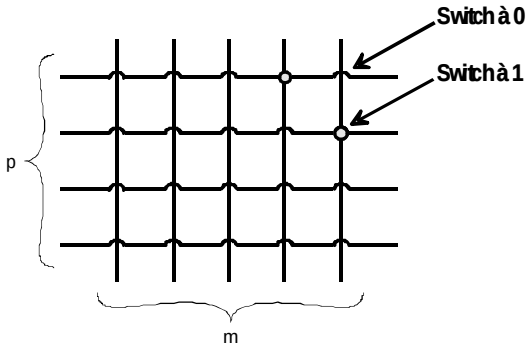
33

Fonctionnement

- **Tableau de connecteurs**
- **Toutes les permutations sont possibles à un instant donné**
- **Un seul processeur avec un seul module mémoire**
- **Nombre de connecteurs = p*m**

34

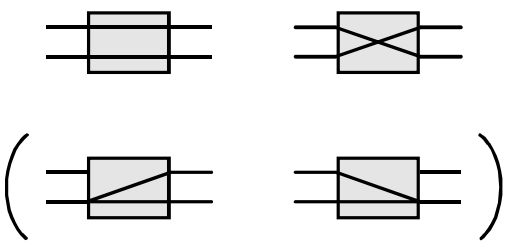
CrossBar



35

Commutateur de base

- **Construction à partir d'une cellule de base C22**
- **2 entrées/2 sorties**



36

RÉSEAUX DYNAMIQUES

37

Non bloquant

- **Une nouvelle connexion entre une entrée libre et une sortie libre est toujours possible**
- **Indépendant des connexions courantes.**

40

Fonctionnement

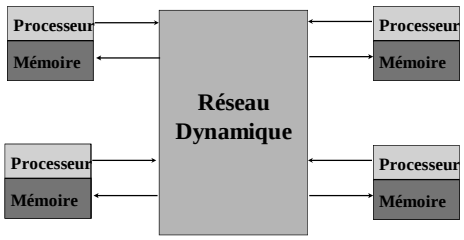
- **Interconnexion sans bus pour des architectures MIMD et SIMD**
- **Les connections s'établissent au fur et à mesure des communications**

38

Ré-arrangeable

- **Simplification du hard (Complexité)**
- **Une nouvelle connexion est toujours possible, elle peut demander une modification des chemins existants pour être établie**

41



39

Bloquant

- **Encore plus simple**
- **Certaines connexions peuvent ne pas être établies en fonction des connexions en cours**

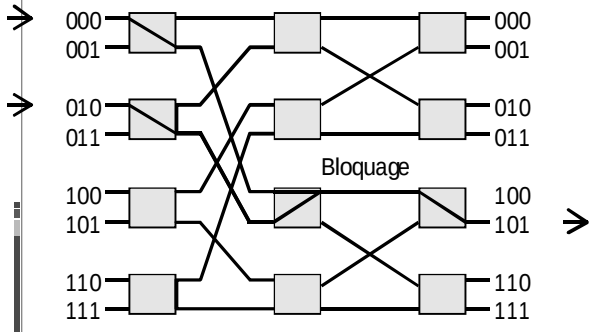
42

Degré du réseau

- **Simple étage**
  - **L'information ne traverse qu'un seul étage de connecteurs**
  - **Ex: Cross-bar 1 n*m connecteurs O(n2)**
- **Multi étages**
  - **L'information traverse plusieurs connecteurs**
  - **Construction à partir de cross-bar ou à partir d'une cellule de base**
  - **Ex: Clos (cours de DEA)**

43

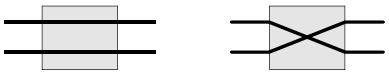
Baseline 8x8



46

Construction de réseaux

- **A partir de la cellule de base**



44

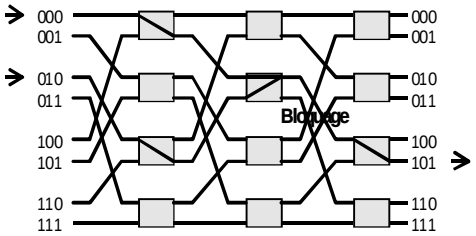
Algorithme de routage

- **Contrôlé par le numéro de destination**
- **=> 0 sortie en haut**
- **=> 1 sortie en bas**
- **Adapté à une transmission par paquets**

47

Réseaux bloquants

- **Conflit sur une liaison est irréductible.**
- **Exemple: réseau Oméga**
- **Il y a conflit lorsque 0 et 2 connecte 4 et 5**



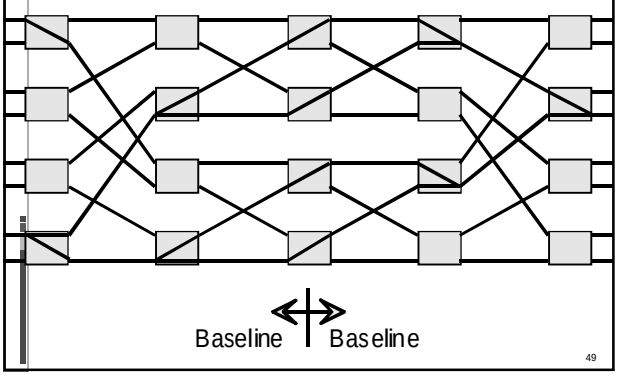
45

Ré-arrangeable:

- **Réseau de Bénes construit à partir d'un baseline et de son symétrique**
- **4 chemins différents pour un couple (E,S)**
  - **4 choix possibles => ré arrangement**

48

Bénes



49

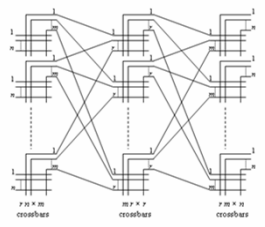
Fonctionnement

- **Les chemins sont fixés entre 2 éléments => lien uni ou bidirectionnel**
- **On achemine des données entre les processeurs**
- **Communications sont par message**
  - **Entête: numéro du destinataire**
  - **Corps: données**

52

Non bloquant:

- **Clos nxm non bloquant si ($m \geq 2n - 1$)**
  - **Sinon réarrangeable**



50

Terminologies des graphes

- **Le degré D est défini comme étant le nombre de liens à partir d'un noeud. On introduit D_{max} et D_{min}**
  - **Si $D_{max} = D_{min}$ le graphe est régulier**
- **La distance d est définit comme étant le plus court chemin entre deux noeuds.**
- **Le diamètre D du graphe est la plus grande des distances**

D = max d

53

RÉSEAUX STATIQUES

51

Caractéristiques

- **Diamètre: plus il est grand plus la distance entre 2 noeuds extrêmes est grande.**
- **Connectivité: Nombre de chemins entre 2 noeuds. Résistance aux pannes**
- **Faisabilité: Facilement réalisable - degré fixe - symétrie**
- **Sous -topologies: inclure des topologies simples**

54

Caractéristiques (suite)

- **Les autres caractéristiques des réseaux statiques sont :**
  - **Routage facile**
  - **Diffusion et autres macro-communications**
  - **Extensibilité**

55

Réseau linéaire

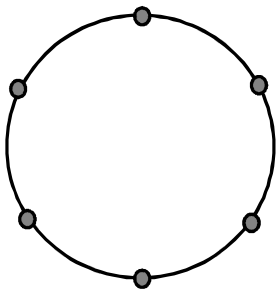
- **D = N-1**
- **D max= 2**

- **Numérotation des processeurs de 0 à N-1**

56

Anneau

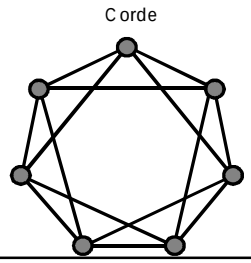
- **D = N/2 (très long)**
- **D = 2**
- **Facile à programmer**



57

Réseau en anneau avec corde

- **Limite la longueur du chemin**
- **Connections étendues à plusieurs voisins**



58

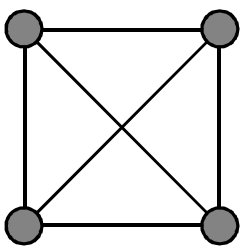
Réseaux complètement connectés

- **Tous les noeuds sont connectés 2 à 2**
- **Avec n noeuds :**
  - **On a n-1 liens par noeud D = n-1**
  - **n(n-1)/2 chemins différents**
- **L'accès est direct D = 1**

59

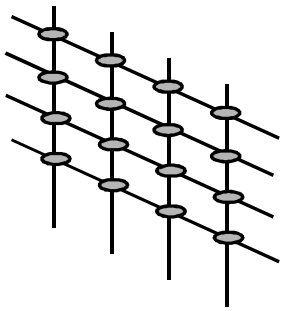
ex: n=4

- **=> pas de conflit**
- **=> utilisable pour n petit**



60

Réseaux 2 dimensions



61

Grille

- **Pas de lien de rebouclage**
- **Proche de l'algorithme**
- **4 voisins N E W S**
- **$D_{max} = 4$**
- **$D = \sqrt{2}(N-1)$**

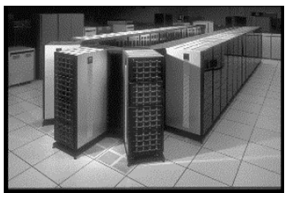
64

Grille torique

- **4 liens par noeud (ou 8 sur la MasPar, liens supplémentaires sur diagonale)**
- **2n liens sur le tore**
- **D = 4**
- **D = √N**
- **Numérotation P(i,j)**

62

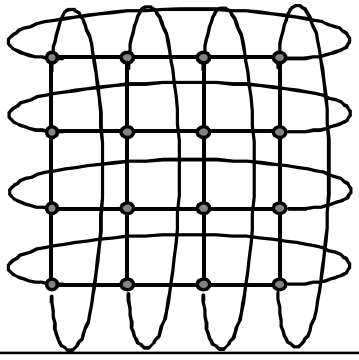
Real World 2D mesh



- 1824 node Paragon: 16 x 114 array

65

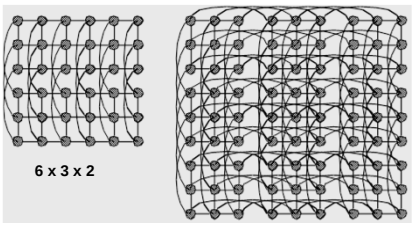
Grille torique



(tore)

63

Extensions en 2 dimensions

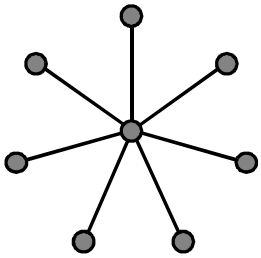


- dimensions logiques multiples par des liens plus longs

66

Réseau étoile

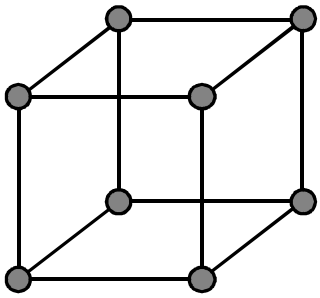
- **n-1 liens**
- **Bottleneck sur le noeud du centre**
- **D max = N-1**
- **D min = 1**
- **D = 2**



67

Cube

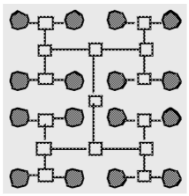
- **D = 3**
- **D= 3**

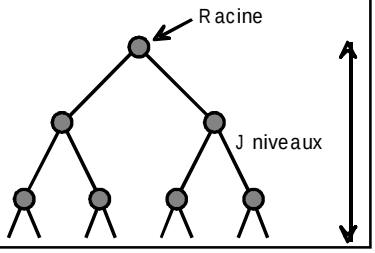


70

Réseau arborescent

- **Exemple Binaire**
- **D = 3**
- **D = 2 $Log_2(N)$**



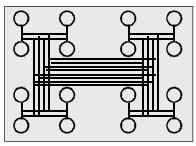


Tore de dimension 3

- **Même construction que les grilles toriques de 2 dimensions**
- **Produit de 3 anneaux**
- **Extension à plusieurs dimensions**

71

Fat-Trees



- Liens : plus épais plus on monte
- BW fonction de N

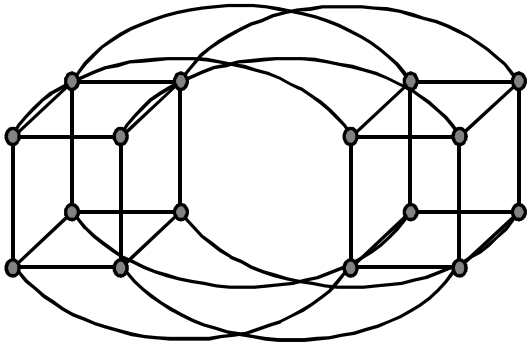
69

Hypercube

- **Construction par récurrence d'un hypercube de degré N à partir de deux hypercubes de degré N-1.**
- **Il suffit de relier deux à deux les sommets correspondants des deux hypercubes initiaux.**

72

Hypercube 4



73

Hypercube (suite)

- **Pour un hypercube de degré n, on a $N = 2^n$ noeuds**
- **D = n = log N**
- **D = n = log N**
- **En tout on a $2^n * n/2$ liens**
- **Réseau régulier, excellents diamètre et connectivité**

74

Routage sans table

- **Pour émettre un message entre deux processeurs:**
  - **On peut facilement trouver un chemin valide en connaissant seulement le numéro du destinataire ainsi que les numéros des noeuds traversés.**
- **Pas de vue globale du réseau**

75

Informations locales

- **Chaque Noeud doit donc:**
  - **Connaître son numéro**
  - **Connaître le numéro du destinataire**
  - **Les arcs sont numérotés en fonction du poids du bit qui varie entre 2 voisins**

76

Méthode du XOR

- **Par un XOR on obtient les arcs sur lesquels on peut envoyer le message**
  - **ex: de 111 vers 001 on obtient**

    111
    XOR 001
    110 => choix des arcs 1
    210 ou 2
  - **on choisit 1 on arrive sur 101**

    101
    XOR 001
    100 => choix de l'arc 2
  - **on arrive sur 001**

77

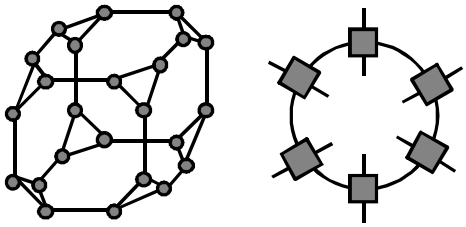
Propriétés

- **La distance entre 2 noeuds est égale à la distance de Hamming entre les 2 numéros des noeuds**
- **2 noeuds diamétralement opposés ont une distance de Hamming de n ($N = 2^n$), tous les bits sont différents**

78

Hypercube à cycles connectés

- **Chaque sommet est constitué d'un anneau de noeuds**
- **Nombre de noeuds de l'anneau = N**



79

MÉTHODES DE TRANSMISSION

82

Caractéristiques des liens

- Temps de cycle Tch: pour positionner un bit sur un fil (Bandwith = 1/Tch)
- Largeur w: nombre de bits transférés en même temps
- Type de transmission : unidirectionnel et bidirectionnel.

80

Les modes de communication

- **Protocole d'acheminement des messages**
- **Découpage des messages?**
- **Blocage en cours de route?**
- **Routage des messages?**

83

Machines réelles

Machine	Topology	Cycle Time (ns)	Channel Width (bits)	Routing Delay (cycles)	Flit (data bits)
nCUBE/2	Hypercube	25	1	40	32
TMC CM-5	Fat-Tree	25	4	10	4
IBM SP-2	Banyan	25	8	5	16
Intel Paragon	2D Mesh	11.5	16	2	16
Meiko CS-2	Fat-Tree	20	8	7	8
CRAY T3D	3D Torus	6.67	16	2	16
DASH	Torus	30	16	2	16
J-Machine	3D Mesh	31	8	2	8
Monsoon	Butterfly	20	16	2	16
SGI Origin	Hypercube	2.5	20	16	160
Myricom	Arbitrary	6.25	16	50	16

81

Store and forward

- **Message est stoppé sur chaque noeud, déposé en mémoire locale puis réexpédié vers le processeur suivant**
- **Routage logiciel**
- **Simplicité du circuit d'interconnexion**
- **Temps de traversée proportionnel à la distance**
- **Interruption du processeur**

84

Temps de transmission

- Pour un message de l bits avec un header de H on obtient:

$$T = Tch[d \times (l + H)/w]$$

- d : distance entre les deux noeuds
- (Méthode store and forward)

85

Routage câblé

- **Routage matériel**
- **Plus d'arrêt du message sur les noeuds**
- **Plusieurs modes d'acheminement**
  - **Circuit-switched**
  - **Worm-hole**
  - **Virtual cut through**
  - **Packet-switched**

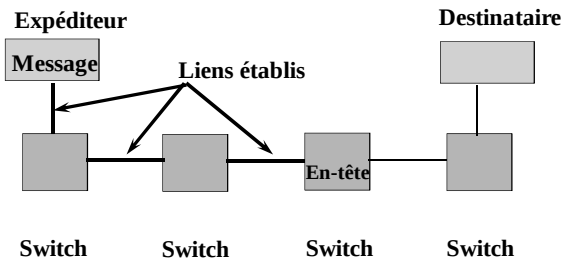
86

Circuit-Switched

- **Emission de l'en-tête du message (numéro du destinataire)**
- **Etablissement des connexions le long du chemin**
- **Dès l'arrivée au destinataire, envoi d'un Ack sur ce chemin**
- **Emission du message sur ce chemin**
- **Libération du chemin**

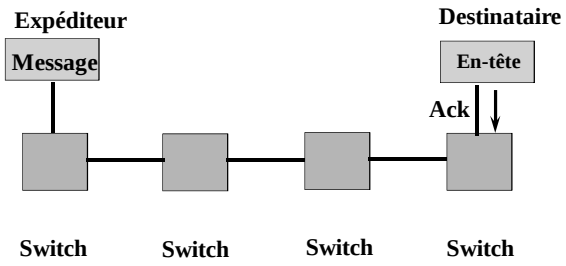
87

Circuit-switched



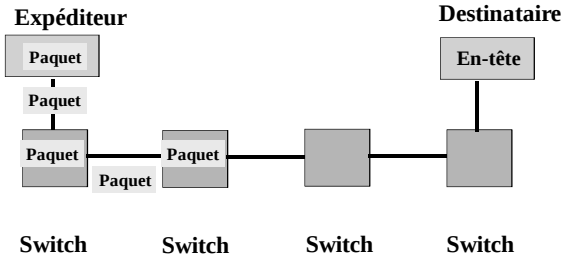
88

Circuit-switched



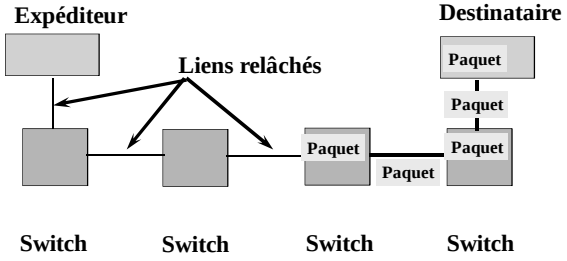
89

Circuit-switched



90

Circuit-switched



91

Worm-hole

- **Envoi de l'en-tête vers les destinataires**
- **Mais le corps du message suit tout de suite l'en-tête**
- **En cas de blocage de l'entête, les paquets sont mémorisés dans les switches du chemin**
- **On utilise le temps d'établissement du circuit pour la communication**

92

Temps de transmission

- On obtient une réduction du temps de transmission:

$$T = Tch [d \times H / w + l / w]$$
$$= Tch [d \times h + l / w]$$

avec $h = H / w$ temps de transfert du header

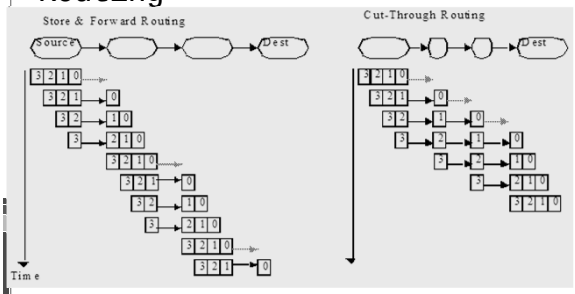
93

Virtual cut Through

- **Fonctionnement semblable au Worm-hole**
- **En cas de blocage les paquets sont acheminer jusqu'au switch sur lequel l'en-tête est bloqué**
- **On libère au plus vite les noeuds du chemin**
- **Pas de réalisation matérielle car il faut des mémoires de switch très grandes**

94

Store&Forward vs Cut-Through Routing



95

Packet-switched

- **Découpe du message en petits paquets**
- **Un en-tête dans chaque paquet**
- **Chemins différents**
- **Pas de réservation de lien**
- **Risques d'interblocage**

96