Algoritmi paraleli
Curs 5

Vlad Olaru

vlad.olaru@fmi.unibuc.ro

Calculatoare si Tehnologia Informatiei

Universitatea din Bucuresti

Topologii de retea interprocesor

- reprezentate ca o retea de procesoare $G = (N,A)$
- presupuneri:
  - 1. comunicare simultana pe toate arcele adiacente unui nod (procesor) in ambele directii (pt graf neorientat)
  - 2. absenta intarzierilor in coada, intarzieri egale pt pachete de dimensiune egala pe toate arcele
  - 3. intarzierea unui pachet pe un arc este de 1 unitate de timp (in absenta altor precizari)
  - 4. algoritmii sunt initiati simultan pe toate nodurile (procesoarele)
- unele topologii sunt mai potrivite decat altele pt. task-uri de comunicare standard

Criterii de evaluare topologii (1)

- *(1) diametru* – distanta maxima dintre oricare perechi de noduri (nr. min de arce)
  - ex: topologie cu diametru r => timp de transmisie pachet $O(r)$
- *(2) conectivitate* – de arce sau de noduri, limitata superior de *min degree(n)*
  - nr min de arce/noduri care trebuie eliminate pt. ca graful sa devina neconex
  - util pt. toleranta la defecte sau cresterea gradului de paralelism al transmisiei
    - comunicatia se pastreaza in prezenta defectelor de noduri sau arce
    - reducere timp de comunicare cu factor k
      - ex: topologie cu conectivitate k => min k cai de comunicatie individuale intre doua noduri/procesoare, fara arce comune
      - mesaj lung impartit in pachete trimise in paralel pe k cai

      Obs: pachetele pot ajunge neordonat la destinatie => mesajul trebuie reasamblat in ordine

Criterii de evaluare topologii (2)

- *(3) flexibilitate*
  - (1) capacitatea de a rula un algoritm dezvoltat pe o topologie $G_1=(N_1,A_1)$ pe o alta topologie $G_2=(N_2, A_2)$
  - pp existenta unei functii care asociaza fiecarui nod din G_1 un nod G_2 din a.i. arcele din G_1 corespund unor arce din G_2

$$f : N_1 \rightarrow N_2 \text{ a.i. } f(i) \neq f(j) \; daca \; i \neq j \; si \; \big(f(i), f(j)\big) \in A_2 \; daca \; (i,j) \in A_1$$

  - se spune ca G_1 se *mapeaza* pe G_2
  - (2) permite divizarea unui task intre procesoarele unei topologii date a.i. sa se minimizeze comunicatia
    - pp taskul divizat in subtaskuri, fiecare asociat cu un procesor din graf
    - pp anumite subtaskuri *(i,j)* interactioneaza, i.e. executia i sau j pp cunostinta unor valori calculate in timpul executiei j sau i

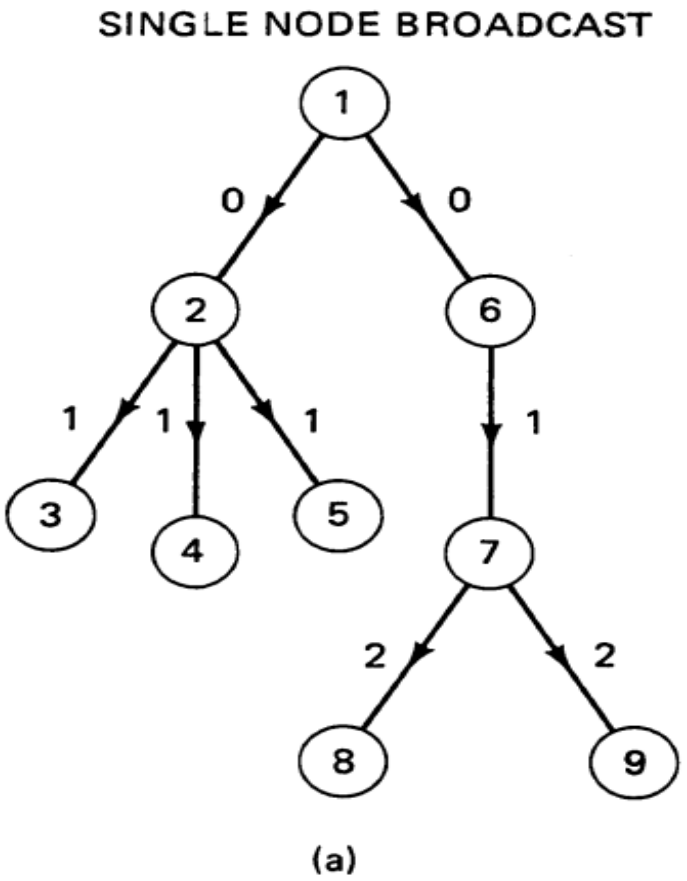
    $\Rightarrow$ subtaskurile se asigneaza unor procesoare care au link direct de comunicare

    $\Rightarrow$ graful de interactiuni intre subtask-uri G_1 se *mapeaza* pe graful de conexiuni G_2
    - ex: mapare graf de dependente pe graful de interconexiune procesoare in executia unei iteratii din metode de relaxare

$$x_i(t+1) = f_i(x_1(t), \ldots, x_n(t)) \quad i = 1, \ldots, n$$

Criterii de evaluare topologii (3)

- *(4) intarzierea la comunicare* – importanta pt task-uri de tip broadcast, scatter/gather, etc pt. operatii de produs scalar, inmultire vector-matrici, etc
- exemple de operatii:
- **(1) difuzare uninod (single node broadcast) / multinod (multinode broadcast)**
  - trimiterea unui pachet de la un procesor la toate celelalte, respectiv executia simultana din toate nodurile a unui broadcast single node
  - ex. broadcast multinod: interatiile din metodele de relaxare studiate anterior
    - daca x_i e asociata unui procesor si f_i depinde de toate variabilele => la sf iteratiei fiecare procesor trebuie sa trimita valoarea variabilei sale catre toate celelalte procesoare
  - solutie single node broadcast: *spanning tree (arbore de acoperire)* cu radacina in nodul sursa



Intarzierea la broadcast

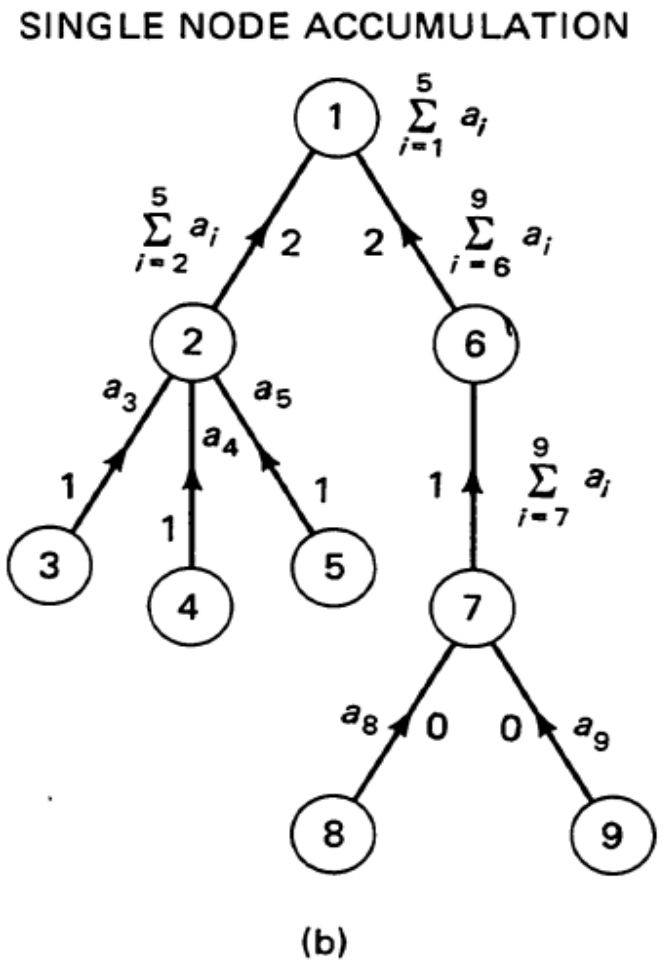
- **difuzare uninod (single node broadcast) / multinod (multinode broadcast)**
  - solutie single node broadcast: *spanning tree (arbore de acoperire)* cu radacina in nodul sursa
    - alegerea potrivita a arborelui de acoperire poate permite executia difuzarii in $O(r)$, unde r = diametrul retelei
  - optimizare broadcast mesaje lungi
    - mesajul impartit in pachete mici, trimise individual pe arcele din arborele de acoperire

    $\Rightarrow$ efect similar pipelining (cf. slide-uri curs trecut)
  - reduce timpul de broadcast de la $O(r)$ la $O((r + m - 1)/m)$, pt m pachete per mesaj, care necesita fiecare $1/m$ unitati de timp de transmisie pe link
  - solutie difuzare multinod: folosim cate un arbore de acoperire pt fiecare nod sursa
    - dificultate: arborii de acoperire pot partaja arce, la sosirea simultana a mai multor pachete la acelasi nod trebuie modelata si intarzierea de comunicare pe link

    => nevoia de analiza specifica, pt fiecare tip de retea de interconectare

Acumulare uni- si multinod (1)

- **single node and multinode accumulation**
  - problema duala difuzarii, in sensul in care foloseste tot arbori de acoperire ca solutie
  - single node accumulation
    - fiecare nod trimite un pachet catre acelasi nod
    - pp. pachetele se pot “combina” in transmisia pe un arc, intarzierea unui pachet “combinat” fiind aceeasi cu aceea a unui pachet obisnuit
      - ex tipic: calculul prefix, acumularea sumelor partiale ale unui produs scalar



Acumulare uni- si multinod (2)

- **single node and multinode accumulation**
  - multinode accumulation
    - acumulare uninod simultana la mai multe noduri
  - se poate demonstra ca problemele de acumulare se rezolva in acelasi timp cu cele de difuzare
    - ex: single node accumulation poate fi vazut ca un algoritm de single node broadcast rulat invers; reciproca e si ea adevarata

Alte operatii

- **schimb total (total exchange)**
  - trimite un pachet de la orice nod catre orice nod
  - diferenta fata de multinode broadcast: se trimit pachete diferite, nu acelasi pachet !
- **single node scatter**
  - trimite un pachet separat de la fiecare nod la toate celelalte noduri
- **single node gather**
  - colecteaza la un nod un pachet trimis de toate celelalte noduri

Observatii

- in multinode broadcast, fiecare nod primeste un pachet diferit de la toate celelalte noduri => rezolva single node scatter
- total exchange
  - versiune multinod a doua probleme de single node scatter si single node gather
  - de asemenea, generalizare a multinode broadcast in care pachetele trimise de fiecare nod celorlalte este diferit
- concluzie: problemele de comunicare discutate formeaza o ierarhie in termeni de dificultate
  - un algoritm care rezolva o problema in ierarhie rezolva urmatoarea problema din ierarhie fara cost de timp additional
  - total exchange rezolva multinode broadcast (accumulation)
  - multinode broadcast (accumulation) rezolva single node gather (scatter)
  - single node scatter (gather) rezolva single node broadcast (accumulation)

Ierarhia complexitatii operatiilor de comunicare

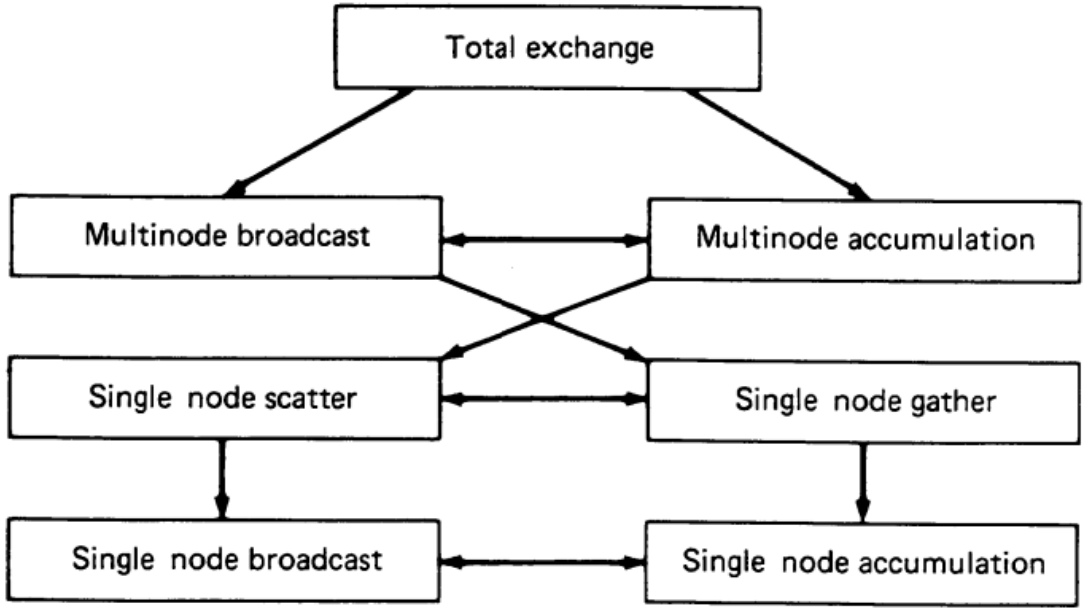
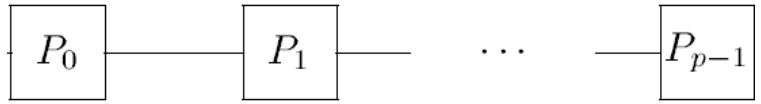


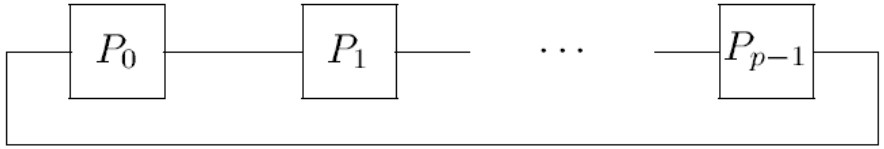
Figure 1.3.5 Hierarchy of basic communication problems in interconnection networks. A directed arc from problem A to problem B indicates that an algorithm that solves A can also solve B, and that the optimal time for solving A is not more than the optimal time for solving B. A horizontal bidirectional arc indicates a duality relation.

Topologii specifice

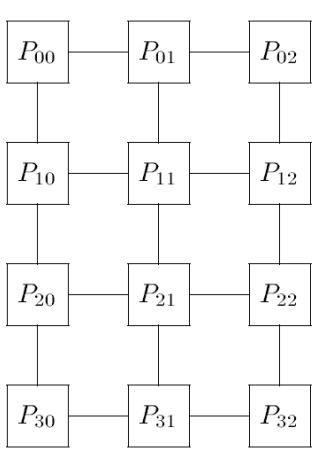
- Lant liniar

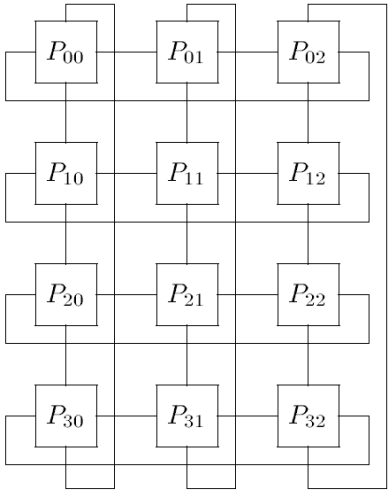


- Inel



- Grid/Tor

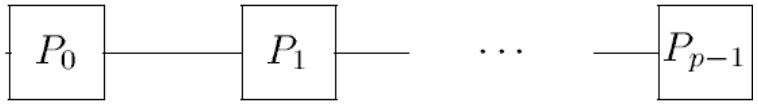




Graf complet

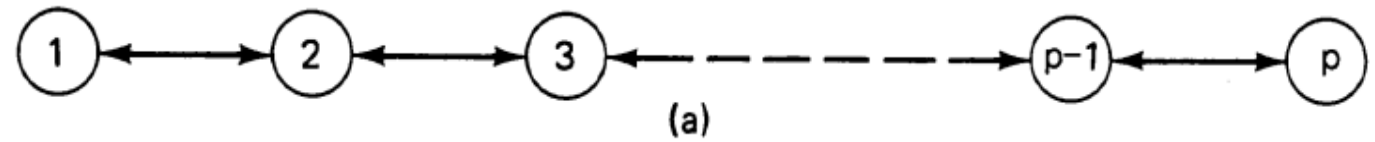
- implementare: magistrala (bus) partajata sau crossbar switch
- flexibilitate maxima
- nr mare de procesoare => crossbar switch complex si costisitor, respectiv intarzieri mari pe bus (in principal datorita asteptarilor in cozi)
- utilizate frecvent in conectarea unui nr redus de procesoare in clustere care apoi sunt interconectate ierarhic

Lant liniar (1)

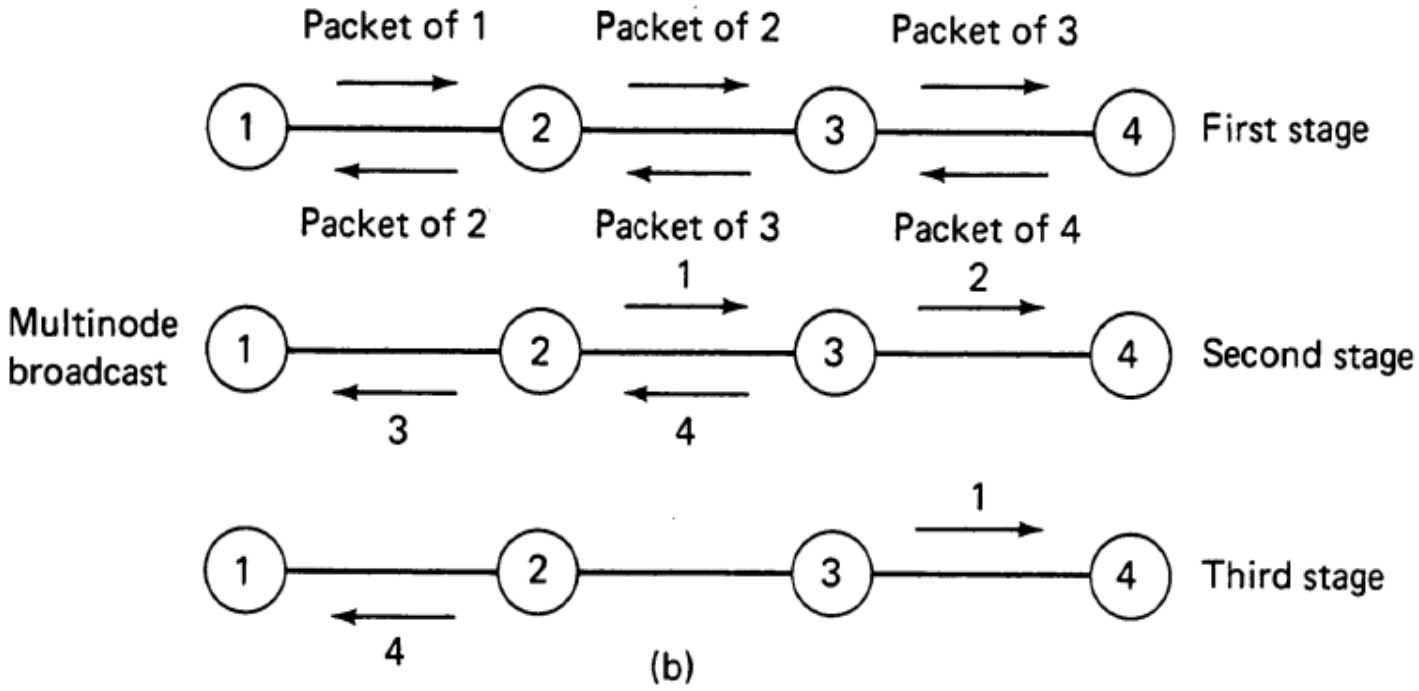


- cele mai defavorabile valori pt diametru si conectivitate
- se poate mapa in mai toate topologiile despre care vom vorbi => penalizarea comunicarii nu se poate imbunatati mai mult decat la celelalte topologii
- timpul de broadcast depinde de nodul initiator, worst case $p - 1$ (diametrul)
  - obs: transmisia pachetului costa o unitate de timp
- timpul de multinode broadcast se poate optimiza tot la $p - 1$ prin folosirea in paralel a tuturor link-urilor (conexiuni full duplex):
  - algoritm in etape
  - in prima etapa, fiecare nod trimite in stanga si dreapta sa valoarea proprie
  - in fiecare etapa k, nodul i primeste pachetele nodurilor $i - k$ *(pt* $i > k$*)* si $i + k$ *(pt* $i + k <= p$*)*
  - multicastul se termina dupa $p - 1$ etape

Multinode broadcast lant liniar



(a)



(b)

Lant liniar (2)

- timpul optimal de single node scatter
  - intre timpul optimal necesar pt single node broadcast si cel de multinode broadcast => $p - 1$ in cel mai defavorabil caz
  - se poate arata ca timpul optimal (si pt single node broadcast) al unui algoritm care incepe operatia in nodul k este de $\max(k-1, p-k)$
- timpul optimal de schimb total $\Theta(p^2)$
  - link-ul $(k, k+1)$ separa lantul in doua subseturi de k si $p - k$ noduri
  - fiecare nod dintr-un subset trebuie sa trimita un pachet catre nodurile din celalalt subset

  $\Rightarrow$ $(k, k+1)$ transporta $k(p-k)$ pachete in fiecare directie

  $\Rightarrow$ daca alegem cel mai defavorabil link, avem cel putin $max_k[k(p-k)]$ pasi (unitati de timp)

  $\Rightarrow$ prin derivare obtinem valoarea optima a lui $k = \frac{p}{2}$

  $\Rightarrow$ ceiling $((p^2-1)/4)$ unitati de timp
  - alternativ, trebuie sa se rezolve secvential p probleme single node scatter pt fiecare procesor
    - fiecare problema necesita cel mult $p-1$ pasi => $\Theta(p^2)$

Inel

- topologie toleranta la defecte
- daca o conexiune se defecteaza, ramane o cale de la orice procesor la alt procesor
- totusi, diametrul poate atinge valoarea *ceiling((p – 1) / 2)*
- orice operatie pe un inel se poate rezolva intre timpul corespunzator pt acea operatie pe un lant liniar cu acelasi nr de noduri si jumatatea acelui timp
- ex: broadcast, MST echilibrat

Tree (arbore)

- p procesoare, orice comunicatie intre doua noduri foloseste cel putin $p - 1$ link-uri
- conectivitate redusa
- defectarea unui link produce doua subseturi de noduri deconectate
- diametrul worst case $p - 1$ (lista, lant liniar)
- topologia star are diametru minim, dar tot traficul trece prin nodul central => bottleneck
- timpul optim de single node broadcast (accumulation) si single node scatter (gather) nu depaseste $p - 1$
- timpul optim de multinode broadcast este $p - 1$
- schimbul total este $O(p^2)$

Mesh/grid/tor

- topologie potrivita pentru probleme legate de geometria spatiului fizic
- grid d-dimensional cu n_i puncte de-a lungul dimensiunii i

  $(x_1, \dots, x_d)$, x_i ia valori de la 1 la n_i
- arc $((x_1, \dots, x_d), (x'_1, \dots, x'_d))$ a.i. $\exists i \mid |x_i - x'_i| = 1$ si x_j != x'_j pt. j != i
- diametru $\sum_{i=1}^{d}(n_i - 1)$
- diametru mai mic ca la inel, dar semnificativ mai mare decat la arbore binar echilibrat cu acelasi nr de procesoare
- torul e o varianta cu diametru mai mic

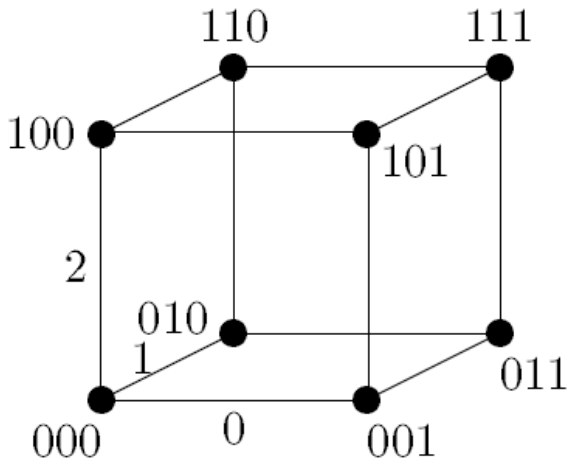
Mesh/grid/tor (2)

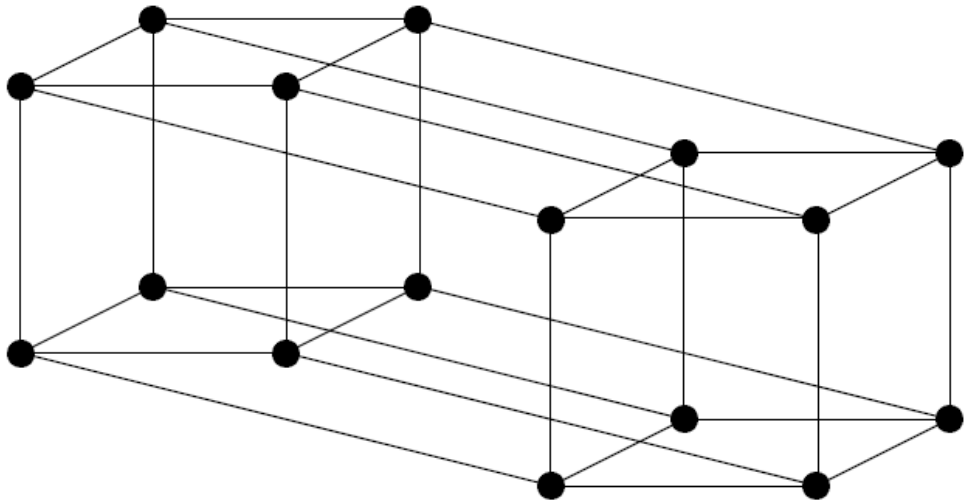
- fie un grid d-dimensional simetric (nr egal de procesoare $p^{1/d}$ pe fiecare dimensiune) cu p procesoare
- diametru $d\ (p^{1/d} - 1)$ => daca pp d fix => timpul optimal de single node broadcast este $\Theta(p^{1/d})$
- maparea lant liniar pe mesh e triviala => timpii optimali de scatter/gather sau multinode broadcast nu depasesc timpii corespunzatori de la lantul liniar $O(p)$
- *dar,* orice scatter ia $\Omega(p)$ pasi pt ca nodul transmite $p - 1$ pachete catre cei max $2d$ vecini (d fix !) => scatter/gather optimal este $\Theta(p)$
- schimb total $\Theta(p^{(d+1)/d})$
  - pp p par si un hiperplan $d-1$ dimensional care imparte gridul in 2 jumatati egale
  - fiecare jumatate are $p/2$ procesoare si primeste $(p/2)^2$ pachete de la procesoarele din cealalta jumatate
  - pachetele trebuie sa traverseze $p^{(d-1)/d}$ linkuri intre cele doua jumatati

  $\Rightarrow \Omega(p^{(d+1)/d})$ unitati de timp
  - se poate arata ca schimbul total este si $O(p^{(d+1)/d})$

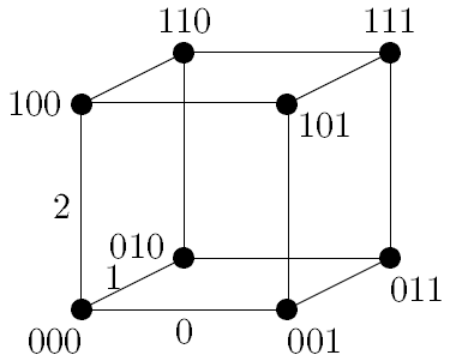
Hipercub

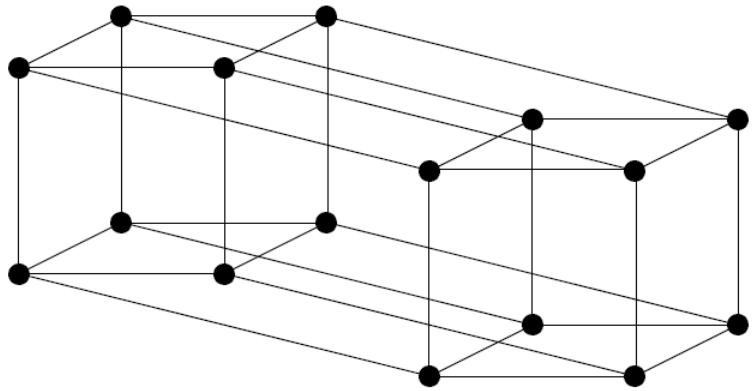
- intr-un spatiu d-dimensional, multimea punctelor cu coordonate 0/1 pot fi vazute ca varfurile unui cub d-dimensional
- daca asociem aceste puncte intr-un graf cu procesoarele si conectam printr-un arc orice doua asemenea puncte care difera printr-un singur bit=> *hipercub* sau *d-cub*
- formal, un d-cub este un grid d-dimensional cu doua procesoare in fiecare dimensiune (n_i = 2 pt. orice i)
- fiecare nod/id procesor al unui d-cub are o reprezentare binara pe d biti





Constructie hipercub





- notam H_d hipercubul de dimensiune d, cu $p = 2^d$ procesoare
- fiecare procesor are d vecini
- P_i vecin cu P_j daca reprezentarile binare ale i si j difera printr-un singur bit (distanta Hamming 1)
- def. recursiva: H_d se obtine prin doua copii H_{d-1} in care se leaga arcele aflate in aceleasi pozitii (aceleasi reprezentari binare) in cele doua H_{d-1}
  - adresele nodurilor din H_d: se adauga un bit in pozitia cea mai semnificativa
    - 0 pentru prima copie H_{d-1}
    - 1 pentru cea doua copie H_{d-1}

Proprietati hipercub

- orice cale dintre doua noduri are un nr de arce cel putin egal cu distanta Hamming dintre reprezentarile binare ale celor doua noduri
- mai mult, exista o cale care are exact lungimea data de distanta Hamming
  - se schimba din 0->1 si invers bitii care difera in reprezentarea binara
  - ex: calea (1101)-> (0110) se construieste astfel: (0101), (0111),(0110)
    sau (1111),(1110),(0110)

=> diametru = d (sau $log\ p$, pt. ca $p = 2^d$)

Mapare lant liniar pe hipercub

- hipercubul este o arhitectura flexibila
- ex: mapare lant liniar cu 2^d procesoare pe hipercub
- problema revine la constructia unei secvente de 2^d nr binare cu d biti a.i. doua numere succesive in secventa difera printr-un singur bit
- solutia: coduri Gray, in particular reflected Gray codes (RGC)
- constructia RGC similara constructiei recursive a hipercuburilor
- RGC au proprietatea ca primul si ultimul nr din secventa difera tot printr-un singur bit => mapare inel pe hipercub

Reflected Gray codes

- data o secventa RGC de dimensiune $d - 1$

$$\{b_1, b_2, \ldots, b_p\}$$

unde $p = 2^{d-1}$ si $b_1, b_2, \ldots, b_p$ siruri binare

- secventa RGC de dimensiune d este

$$\{0b_1, 0b_2, \ldots, 0b_p, 1b_p, \ldots, 1b_2, 1b_1\}$$

- aceasta secventa mapeaza un inel de 2^d noduri pe un d-cub
- in general, se poate mapa orice inel de p procesoare, p par, pe un hipercub de dimensiune $2^{ceiling(\log p)}$
- orice ciclu intr-un hipercub are nr par de noduri => inel cu nr impar de noduri nu poate fi mapat pe hipercub
  - *indicatie*: se numara schimbarile de biti din reprezentarile nodurilor traversate in ciclu

Ex. mapare inel pe hipercub folosind RGC

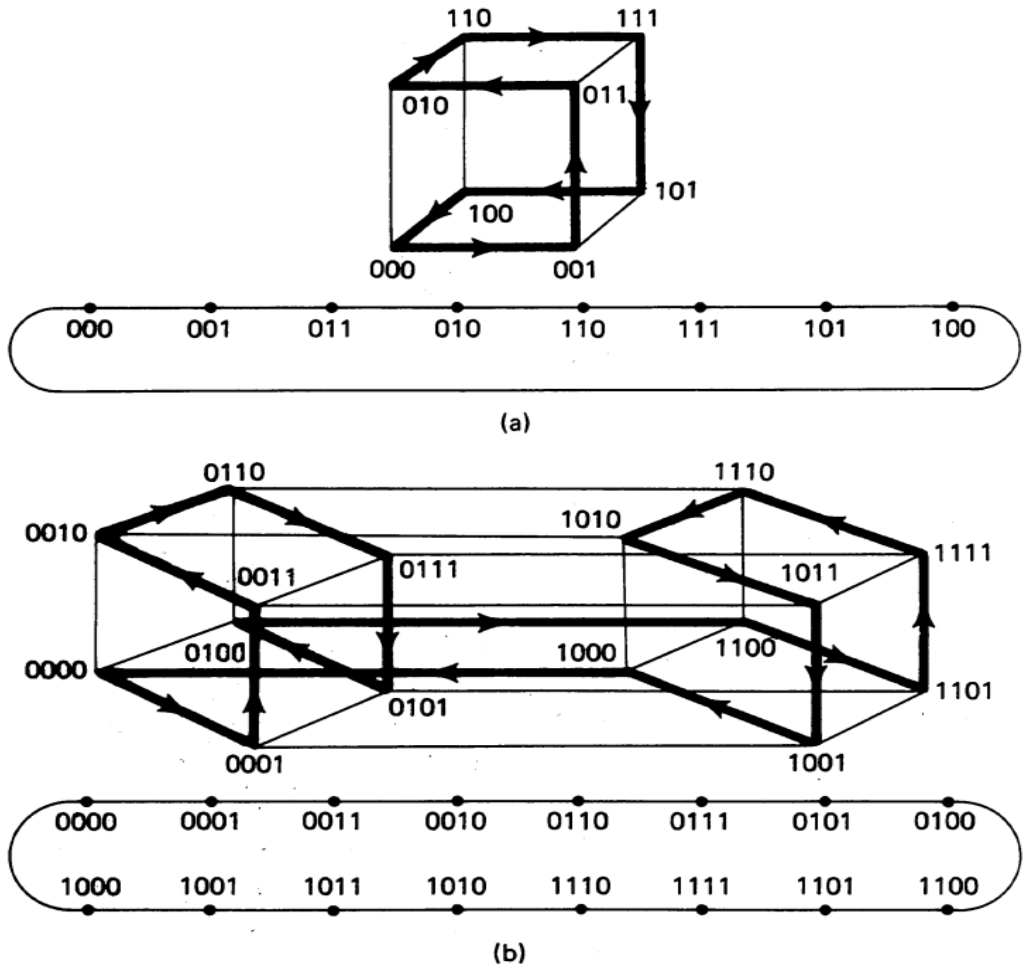


Figure 1.3.11 Reflected Gray code sequences, and the corresponding mappings of rings on (a) a 3–cube and (b) a 4–cube.

Mapare hipercub pe grid bidimensional folosind RGC

- generalizare a constructiei anterioare de RGC-uri
- fie d_a si d_b intregi pozitivi a.i. $d = d_a + d_b$
- fie $\{a_1, a_2, \ldots, a_{p_a}\}$ si $\{b_1, b_2, \ldots, b_{p_b}\}$ secvente d_a-*bit* si d_b-*bit* RGC

unde $p_a = 2^{d_a}$ si $p_b = 2^{d_b}$

Atunci matricea $p_a \; x \; p_b$ de siruri de d biti

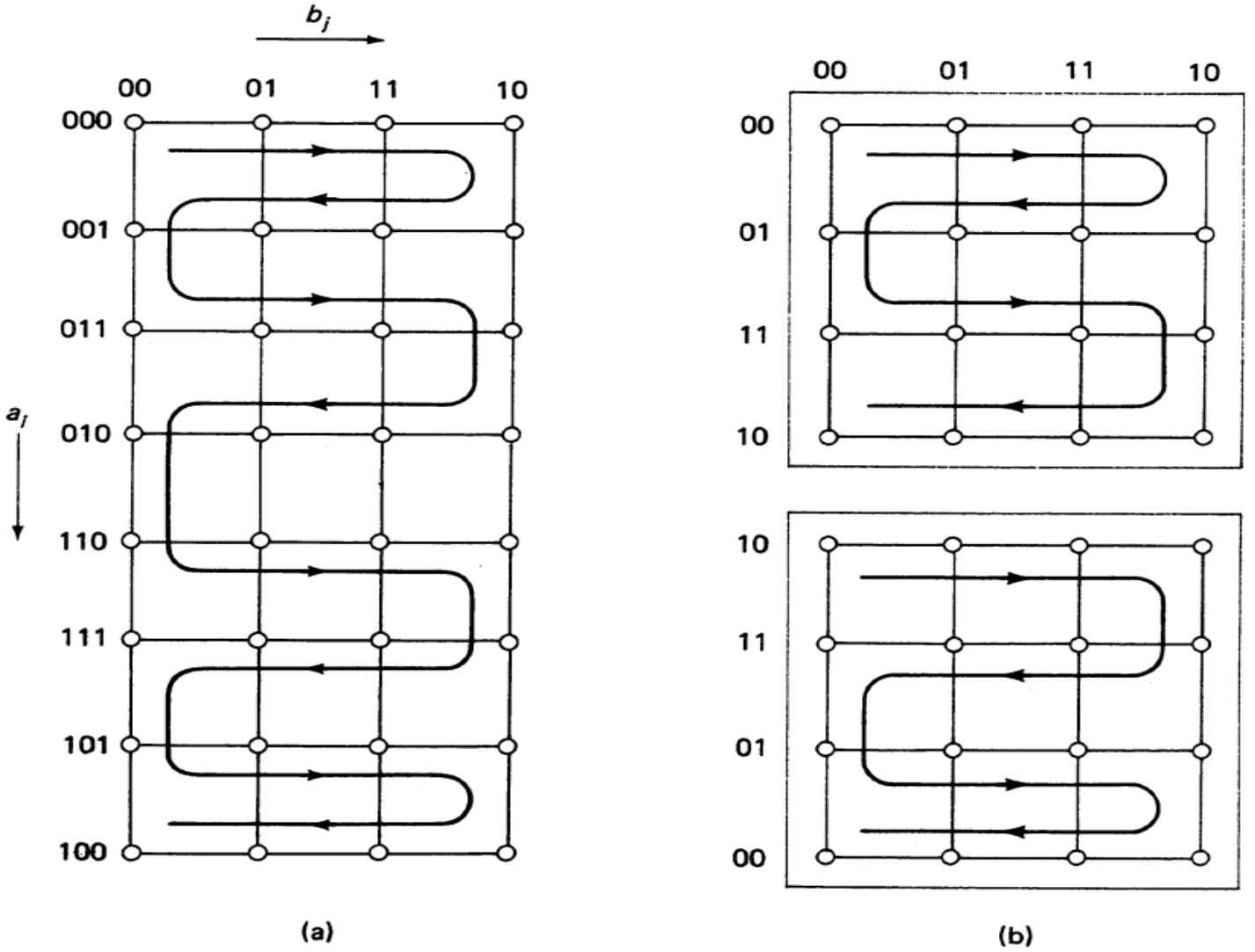
$\{a_i, b_j \mid i=1,2\ldots, p_a, j = 1,2\ldots, p_b\}$

se poate folosi pentru a obtine o secventa *d-bit* RGC traversand liniile matricii alternand stanga-dreapta respectiv dreapta-stanga:

Traversare matrice pt. obtinerea *d-bit* RGC

$$
\begin{bmatrix}
a_1b_1 & \Longrightarrow & a_1b_2 & \Longrightarrow & \dots & \Longrightarrow & a_1b_{p_b} \\
 & & & & & & \Downarrow \\
a_2b_1 & \Longleftarrow & a_2b_2 & \Longleftarrow & \dots & \Longleftarrow & a_2b_{p_b} \\
\Downarrow & & & & & & \\
a_3b_1 & \Longrightarrow & a_3b_2 & \Longrightarrow & \dots & \Longrightarrow & a_3b_{p_b} \\
 & & & & & & \Downarrow \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
 & & & & & & \Downarrow \\
a_{p_a}b_1 & \Longleftarrow & a_{p_a}b_2 & \Longleftarrow & \dots & \Longleftarrow & a_{p_a}b_{p_b}
\end{bmatrix}.
$$

Ex. 5-cub mapat pe grid bidimensional



d-cub (5-cub) mapat pe grid bidimensional

- fig. *a:* nodurile unui d-cub pot fi mapate pe un grid bidimensional cu p_a (8) si p_b(4) noduri pe cele doua dimensiuni
  - elementul *(i,j)* din grid este nodul din d-cub cu id-ul a_i, b_j
  - pe fiecare linie din grid este mapat un d_b–cub cu p_b noduri
  - pe fiecare coloana din grid este mapat un d_a–cub cu p_a noduri
- fig. *b:* constructia inductiva a RGC de dimensiune d
- in mod similar se poate obtine o generalizare a maparii pe grid-uri multidimensionale
  - grid k-dimensional
  - n_i noduri in dimensiunea i, $i = 1, \dots k$
  - $n_i = 2^{d_i}$, $d = d_1 + \dots + d_k$

Comunicarea in hipercub H_d

- fiecare nod are d vecini => exista cel mult d cai independente intre oricare doua noduri
- de fapt, intre orice doua noduri sunt exact d cai independente (care nu au in comun decat capetele caii)
  - daca id-urile capetelor caii difera prin n biti, atunci exista
    - n cai cu n arce intre cele doua noduri
    - $d - n$ cai cu $n + 2$ arce

$\Rightarrow$ *conectivitate* d-cub = d

$\Rightarrow$ *comunicare simultana eficienta pe mai multe cai intre doua noduri ale unui hipercub*

Comunicatie globala

- difuzare
- difuzare generala
- distributie
- schimb complet

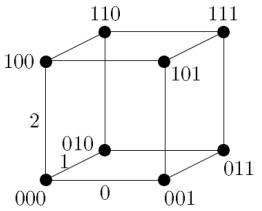
Alte proprietati hipercub H_d

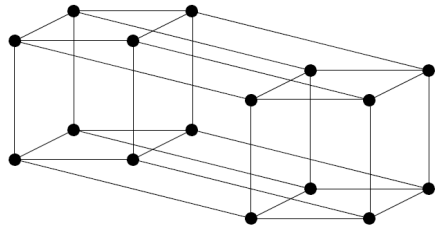
- $\forall$ nod $\boldsymbol{i} \in H_d$ exista un *arbore de acoperire binomial* cu radacina in $\boldsymbol{i}$, in care exista o cale de lungime d de la $\boldsymbol{i}$ la orice nod
- arborele se poate folosi pt single node broadcast de la radacina la orice nod in d pasi

$\Rightarrow$ imbunatatire cu un factor de $\frac{2^d-1}{d}$ fata de un lant liniar cu 2^d procesoare

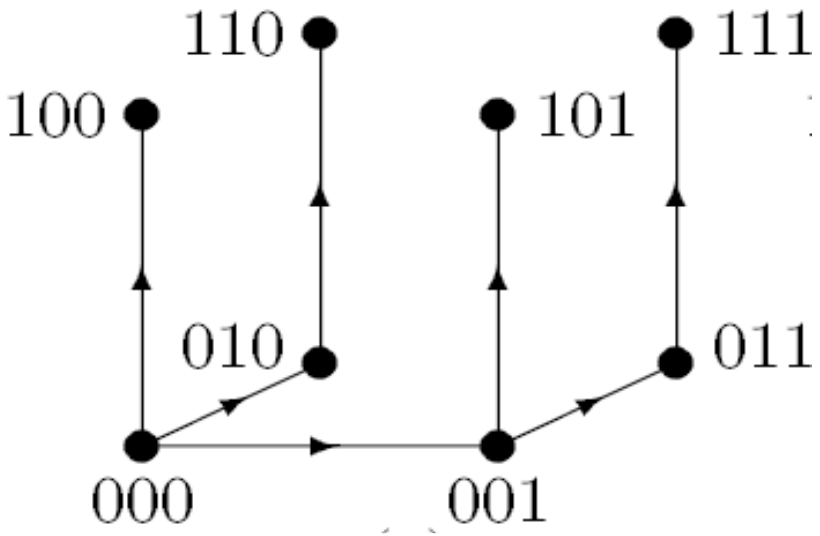
- acelasi algoritm se poate folosi pt. single node accumulation in d pasi

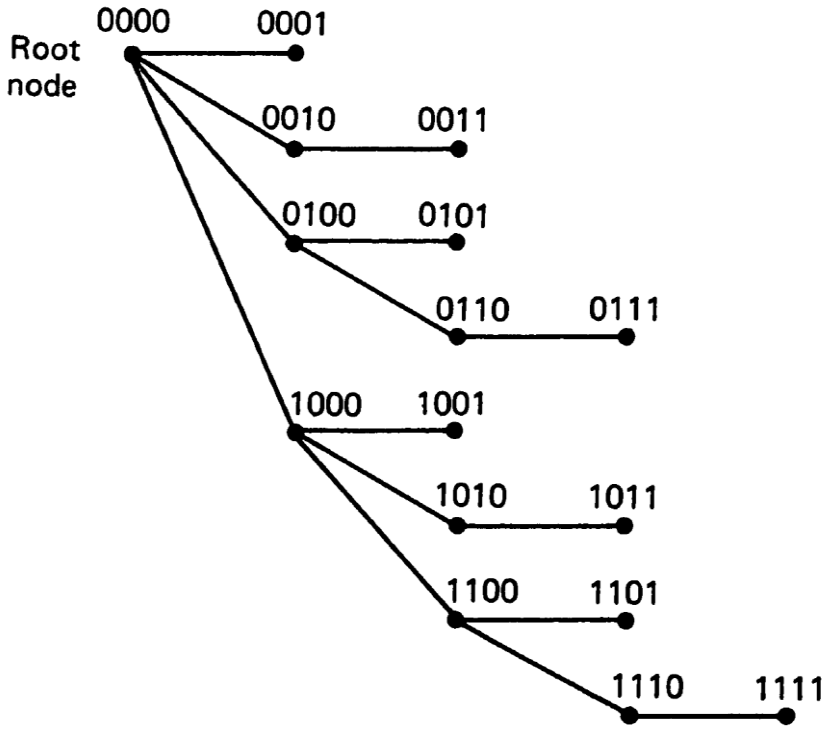
Difuzare hipercub





- constructia porneste din radacina (000…0)
  - constructie generala cu radacina in nodul *i*: aduna *i* mod 2 la valoarea fiecarui nod din arborele cu radacina (000…0)
- *id* copii nod: inverseaza unul dintre bitii zero ai parintelui care urmeaza celui mai din dreapta bit 1



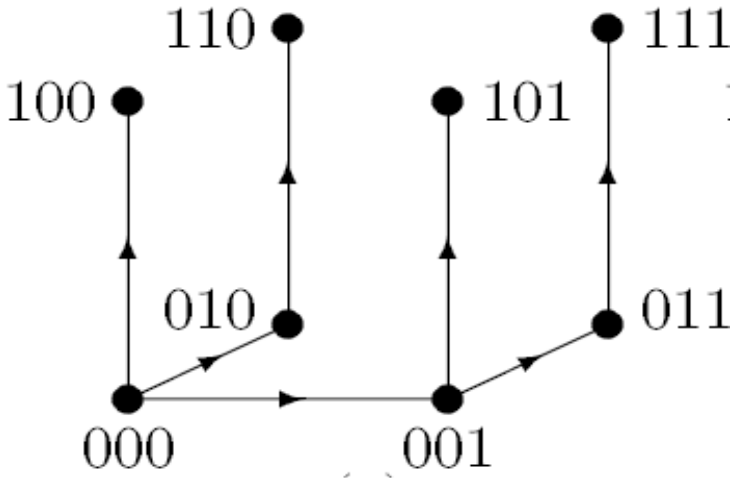


Difuzare hipercub

Notam:

recv(M,s) receptia mesajului *M* de la vecinul s

send(M,s) trimiterea mesajului *M* la vecinul s

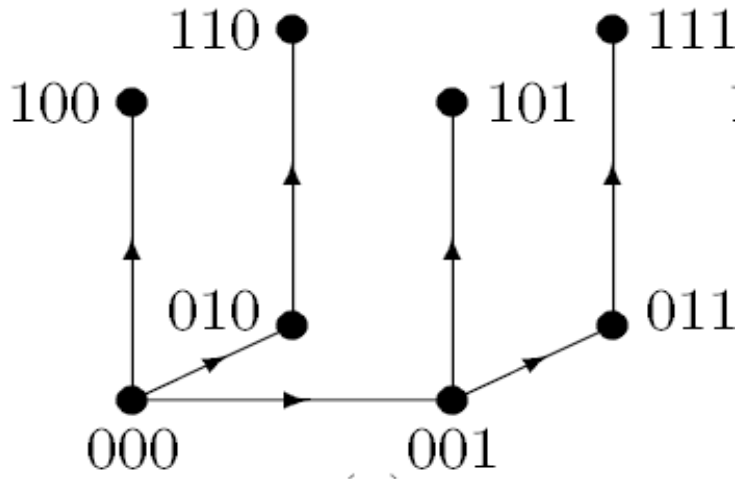


Algoritm difuzare hipercub

1. $l \leftarrow$ poziţia celui mai semnificativ bit de 1 din id (-1 pentru P_0)
2. **dacă** id $\neq 0$ **atunci**
   1. **recv**(M, l) {recepţionează de la tată}
3. **pentru** $i = l + 1 : d - 1$
   1. **send**(M, i) {trimite tuturor fiilor}

Difuzare hipercub

Complexitate $O(d) = O(\log(p))$



Algoritm difuzare hipercub

1. $l \leftarrow$ poziţia celui mai semnificativ bit de 1 din id (-1 pentru P_0)
2. **dacă** id $\neq 0$ **atunci**
    1. **recv**(M, l) {recepţionează de la tată}
3. **pentru** $i = l + 1 : d - 1$
    1. **send**(M, i) {trimite tuturor fiilor}

Difuzare generala hipercub H_d

- fiecare procesor trimite un pachet catre orice alt procesor
- fiecare nod are cel mult *d* vecini => primeste cel mult *d* pachete simultan
- fiecare pachet separat vine de la $2^d - 1$ procesoare
- => difuzarea generala ia cel putin $ceiling((2^d - 1) / d)$ pasi
- exista insa algoritmi optimali care ating aceasta valoare minima pornind de la algoritmul de broadcast (single node)

Algoritm optimal de difuzare generala

- fie un algoritm de broadcast in *m* pasi din nodul 0 reprezentat de o secventa de multimi de arce orientate disjuncte A_1, …, A_m
  - A_i contine arcele care transmit un pachet de la momentul $i - 1$ pana la momentul i
  - daca S_i(respectiv E_i) sunt id-urile de nodurilor de start (respectiv sfarsit) din A_i

    => $S_1 = \{(000\ldots0)\}$ si

    S_i e inclus in $\{(000\ldots0)\} \cup (\bigcup_{k=1}^{i-1} E_k)$

    $\forall\ id \neq 0$ apartine unui E_i

- setul de noduri + secventa A_1, …, A_m alcatuiesc un arbore de acoperire al H_d
- similar, definim un algoritm de broadcast dintr-un nod oarecare *t* ca fiind

  $A_i(t) = \{(t\ xor\ x, t\ xor\ y) \mid (x,y) \text{ arc din } A_i\}$

- obs: (t *xor* x, t *xor* y) e link ⇔ (x,y) e link (i.e., difera in acelasi bit)

Algoritm optimal de difuzare generala

- fie $r_i(x,y)$ = nr de noduri t pt. care (x,y) este arc in $A_i(t)$
- $\forall t$ ca mai sus, $\exists$ un pachet de trimis pe link-ul (x,y) => pt. i oarecare si transmisie simultana pe toate arcele din $A_i(t)$, timpul T_i de transmisie pe aceste link-uri este

$$T_i = \max_{(x,y)} (ri(x,y))$$

pt toate arcele (x,y) | (admitem intarzieri in cozile de asteptare)

$\Rightarrow$ timpul total al difuzarii generale este cel mult $T_1 + T_2 + \ldots + T_m$

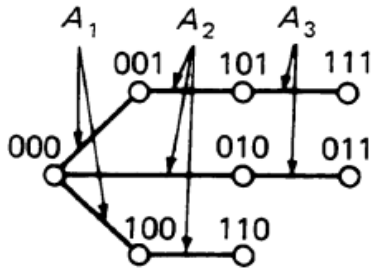
$\Rightarrow$ algoritmul optim trebuie sa selecteze secventa $A_1, \ldots, A_m$ a.i. $T_1 + T_2 + \ldots + T_m$ are valoare mica

si

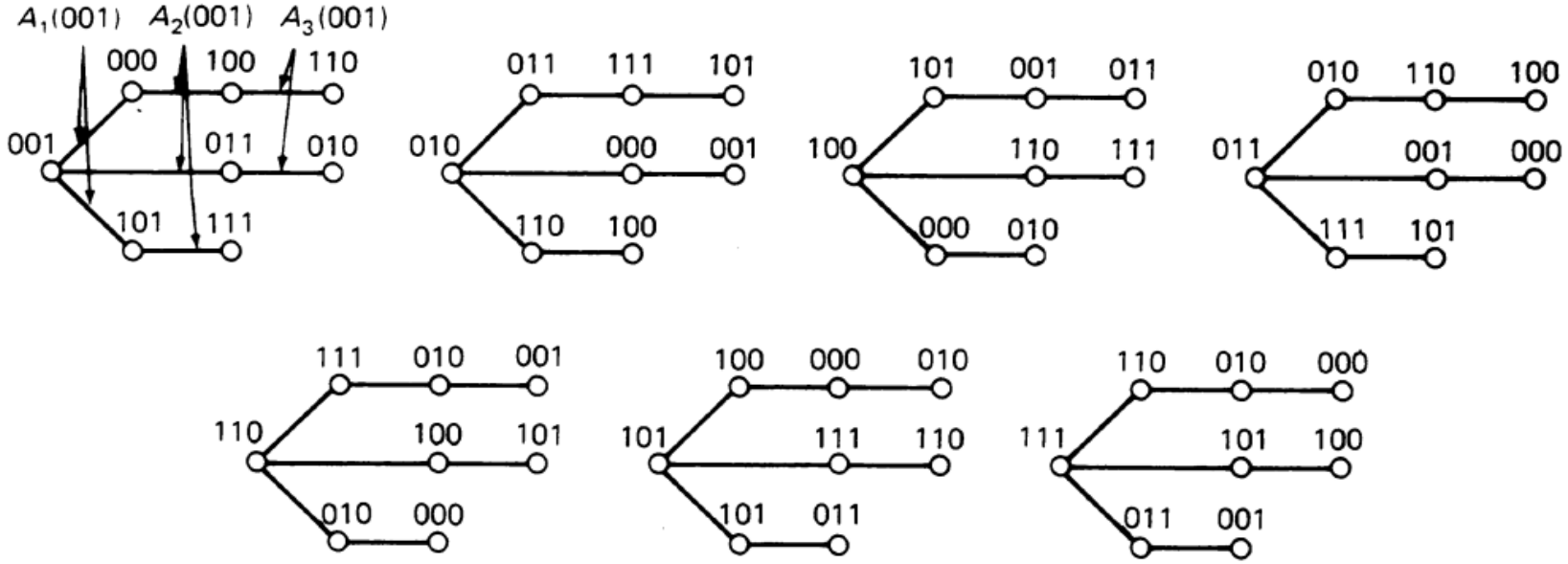
$A_1, \ldots, A_{m-1}$ au d arce, iar A_m are cel mult d arce

- *Obs:* $T_i = 1$ daca $\forall (x,y)$ si $(z,w) \in A_i$, x si y nu difera in acelasi bit ca z si w

Algoritm optimal de difuzare generala



(a)



(b)

Algoritm optimal de difuzare generala

- alegere optimala:

  A_i a.i. $T_i = 1$

  $card(A_i) = d \qquad \forall\, 1 \leq i \leq m-1$

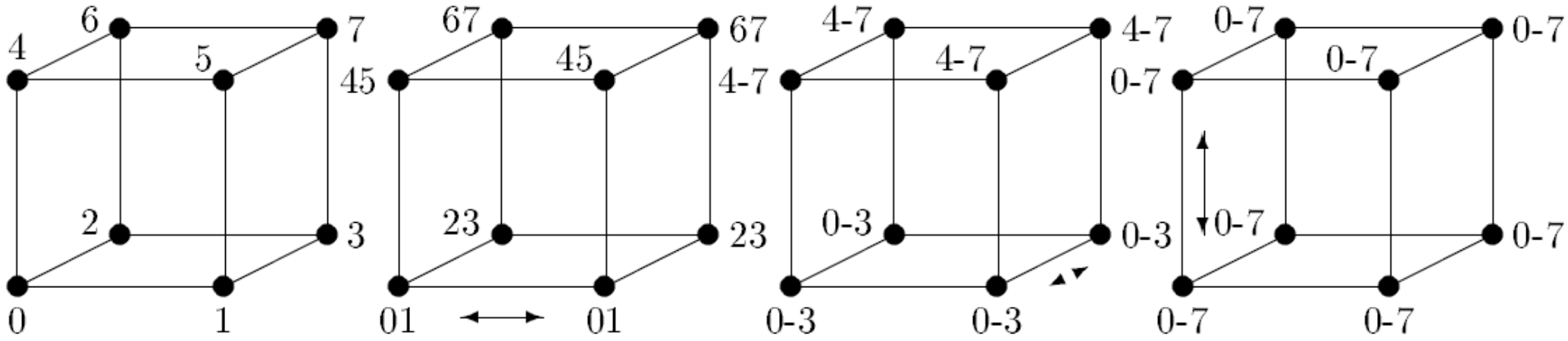
  $card(A_m) < d$

- arborele de acoperire generat de reuniunea multimilor de arce $A_1, \ldots, A_m$ are evident $2^d - 1$ arce

$\Rightarrow T_1 + T_2 + \ldots + T_m = m = ceiling((2^d - 1) / d)$

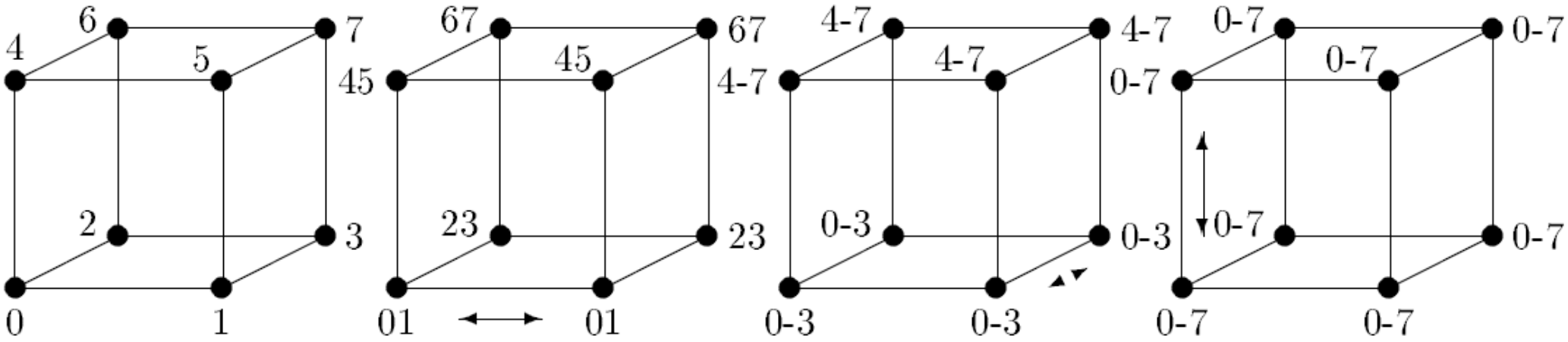
$\Rightarrow$ *timpul optimal difuzare generala pe hipercub de dimensiune d este chiar* $ceiling((2^d - 1) / d)$

Difuzare generala hipercub



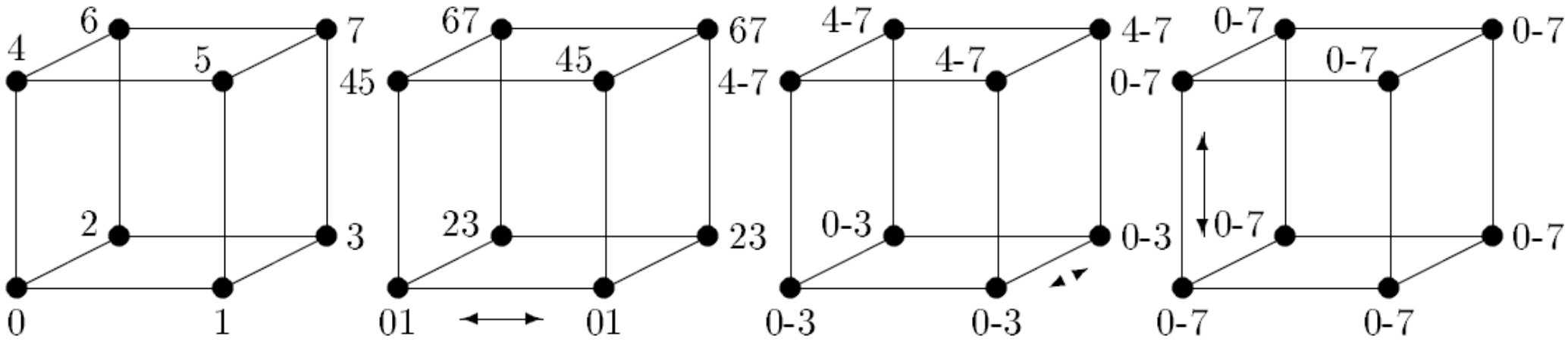
- la fiecare pas se comunica pe o dimensiune: $0, 1, \cdots, d-1$
- fiecare nod transmite vecinului sau din acea dimensiune toate mesajele
- la fiecare pas dimensiunea mesajelor se dubleaza

Difuzare generala hipercub



- dupa 2 pasi nodul 1 poseda mesajele: 0,1,2,3
- la pasul k se incheie difuzarea pe 2^{d-k+1} hipercuburi cu dimensiunile 0,1, …, k-1
- se folosesc doar arcele dintr-o singura dimensiune

Difuzare generala hipercub



- se arata o paralelizare optima, cu folosirea tuturor cailor independente
- complexitate: $O\left(\frac{p}{\log(p)}\right) = O\left(\frac{p}{d}\right)$
- complexitate lant liniar: $O(p)$

Algoritmi optimali de scatter/gather, total exchange pe hipercub

- **single node scatter/gather**
  - algoritmul optimal necesita cel mult *ceiling*$((2^d - 1) / d)$ pasi (multinode broadcast, difuzare generala)
  - $(2^d - 1)$ pachete trimise pe cele d linkuri incidente radacinii => algoritmul optimal necesita cel putin *ceiling*$((2^d - 1) / d)$ pasi

- **total exchange**
  - descompunem d-cubul in doua $d - 1$ subcuburi conectate prin 2^{d-1} linkuri

  $\Rightarrow (2^{d-1})^2$ pachete transmise pe linkuri

  $\Rightarrow$ cel putin 2^{d-1} unitati de timp

  - exista algoritm care atinge acest timp minim

  $\Rightarrow \Theta(2^d)$

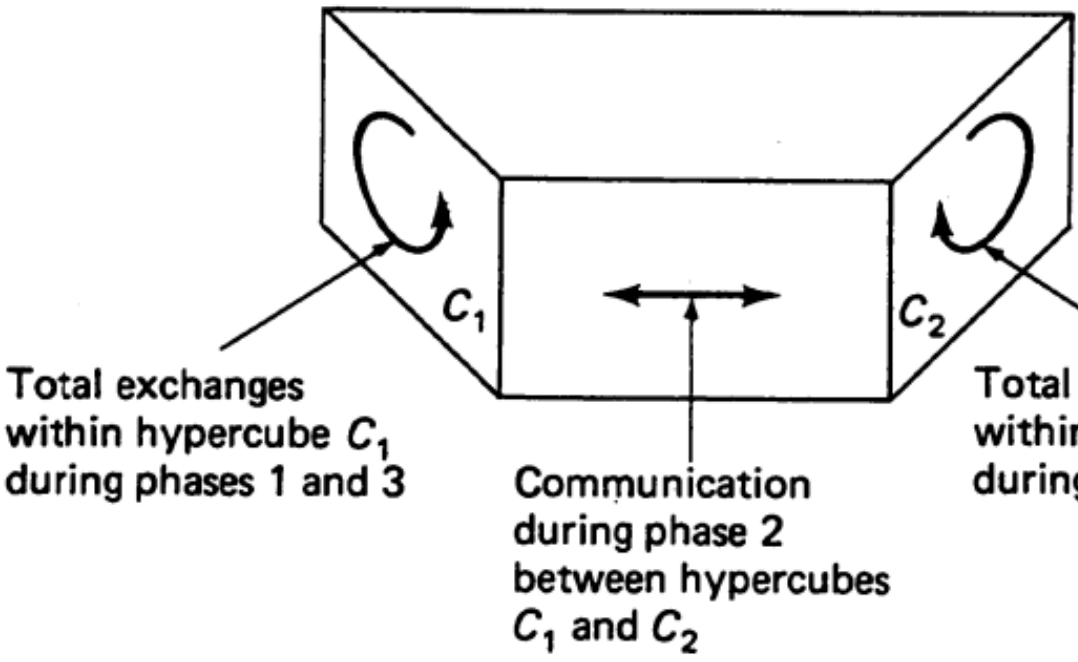
Algoritm optimal total exchange pe hipercub

- algoritm recursiv pt un d-cub care necesita timp in limita unui factor 2 de limita inferioara 2^{d-1}

$$T_d \leq 2^d - 1$$

- pt. $d = 1$, evident ($T_1 = 1$)
- pp. avem algoritm de schimb total pt d-cub cu timpul de mai sus
- construim algoritmul de total exchange pt $(d+1) - cub$
  - descompunem hipercubul in doua d-cuburi C_1 si C_2
  - *faza 1*: schimb total in fiecare dintre cele doua d-cuburi
  - *faza 2*: fiecare nod trimite corespondentului sau din celalalt d-cub toate cele 2^d pachete destinate nodurilor din celalalt cub
  - *faza 3*: schimb total in fiecare d-cub a pachetelor primite in faza 2
  - fazele 1 & 2 ruleaza simultan

Algoritm optimal total exchange pe hipercub



Analiza algoritm optimal de total exchange pe hipercub

- faza 1 dureaza T_d care din ipoteza inductiva este $< 2^d$ pt ca

$$T_d \leq 2^d - 1$$

=> faza 1 & 2 dureaza $< 2^d$

- faza 3 dureaza si ea T_d

=> $T_{d+1} \leq T_d + 2^d \leq 2^{d+1} - 1$

- concluzie: schimbul total poate fi facut in $\Theta(2^d)$ pe un d-cub

Timpi comunicare optimala diverse topologii

Problem	Ring	Tree	Mesh	Hypercube
Single node broadcast (or single node accumulation)	$\Theta(p)$	$\Theta(\log p)$	$\Theta(p^{1/d})$	$\Theta(\log p)$
Single node scatter (or single node gather)	$\Theta(p)$	$\Theta(p)$	$\Theta(p)$	$\Theta(p/\log p)$
Multinode broadcast (or multinode accumulation)	$\Theta(p)$	$\Theta(p)$	$\Theta(p)$	$\Theta(p/\log p)$
Total exchange	$\Theta(p^2)$	$\Theta(p^2)$	$\Theta(p^{(d+1)/d})$	$\Theta(p)$

Comunicare folosind cel mult un link per node

- la orice moment de timp, orice procesor poate folosi cel mult o interconexiune incidenta
- daca *d(p)* este gradul maxim al nodurilor oricarui tip de topologie (*p* = nr de procesoare/noduri) ⇒
  - *d(p)* = factor de slowdown vs. cazul cand nodul poate trimite simultan pe toate link-urile incidente
  - daca transmisia necesita 1 unitate de timp => algoritmii anteriori de transmisie se pot emula in *d(p)* unitati de timp
  - topologii cu *d(p)* independent de *p* (inel, arbore binar echilibrat, grid simetric): timpul optimal identic cu algoritmii cu transmisie simultana pe toate arcele
  - pt. hipercub, se contorizeaza nr total de pachete trimise si se imparte la nr de noduri

Timpi comunicare optimala diverse topologii folosind cel mult un link per nod

Problem	Ring	Tree	Mesh	Hypercube
Single node broadcast (or single node accumulation)	$\Theta(p)$	$\Theta(\log p)$	$\Theta(p^{1/d})$	$\Theta(\log p)$
Single node scatter (or single node gather)	$\Theta(p)$	$\Theta(p)$	$\Theta(p)$	$\Theta(p)$
Multinode broadcast (or multinode accumulation)	$\Theta(p)$	$\Theta(p)$	$\Theta(p)$	$\Theta(p)$
Total exchange	$\Theta(p^2)$	$\Theta(p^2)$	$\Theta(p^{(d+1)/d})$	$\Theta(p \log p)$

Trade-off concurenta-comunicatie

- concurenta = masura a nr. de procesoare simultan active care colaboreaza la rularea unui algoritm paralel
- depinde de modalitatea de impartire a calculului global in subtaskuri si asignarea lor pe procesoare pt. executie paralela (granularitate)
- nevoia de eficienta implica o distributie relativ uniforma a taskurilor paralele pe procesoare, i.e. *load balancing*
- in general, nr de pachete schimbate pt coordonarea subtaskurilor paralele creste cu nr subtaskurilor
  - cu atat mai mult cu cat dimensiunea subtaskurilor e relativ uniforma
- deci, concurenta crescuta => nr de mesaje de sincronizare crescut => CP crescut => scaderea timpului de executie folosind tot mai multe procesoare necesita solutii pt tratarea penalizarii de comunicatie sporite
- din acest motiv, se poate limita superior dimensiunea problemelor de un anumit tip care in teorie, cel putin, s-ar putea rezolva cu un nr nelimitat de procesoare

Reducere CP prin pipelining-ul calculului cu comunicatia

- suprapunearea calcului cu comunicatia (eg, un procesor comunica rezultate partiale catre altul in timp ce alte rezultate sunt calculate)
- ex: metode iterative

$$x_i(t+1) = f_i\big(x_1(t), \dots, x_p(t)\big) \quad i = 1, \dots, p$$

  - x_i vector cu k elemente asignat procesorului i
  - $n = pk$ dimensiunea problemei

Reducerea CP

- mai multe variabile asignate unui singur procesor => variabilele deja actualizate intr-o iteratie pot fi trimise care alte procesoare, in timp ce se asteapta pt actualizarea altora
- header-ul mesajelor inter-procesor e in general constant => pachete mari reduc overhead-ul per bit => transmisia mai multor variabile actualizate in acelasi mesaj reduce CP
- chiar in absenta pipelining-ului, CP se reduce daca dimensiunea k a vectorilor x_i creste
- pp. timp actualizare variabile $\Theta(nk)$ (i.e. f_i liniara fara structura sparse)
- actualizarea x_i pp. ca procesorul i trimite celorlalte procesoare cele k variabile pt. urmatoarea iteratie (sa zicem prin broadcast)
- daca se foloseste un lant liniar (array), timpul optim de comunicare pt broadcast multinod este $\Theta(n)$, daca pp. $\Theta(k)$ timpul de transmisie a k variabile pe un link

Reducerea CP (2)

- concluzie: cand creste k, penalizarea prin comunicare devine insignifianta

$$\text{CP} = T_{comm} \text{ per iteratie} / T_{calcul} \text{ per iteratie} = \Theta(1/k)$$

Obs: 1. raportul e independent de dimensiunea problemei n, depinde doar de k, dimensiunea problemei de calcul per iteratie de procesor

2. accelerarea poate creste cu dimensiunea problemei n

Cresterea accelerarii

- pp timpul serial per iteratie este $T_1 = \Theta(n^2)$ (cf ipoteza anterioara era $\Theta(nk)$, iar pt. o masina seriala $p = 1$ si $k = n$)
- speed-up folosind $k = n / p$ variabile si un lant liniar cu p procesoare

$$S_p(n) = \frac{\Theta(n^2)}{\Theta(n)+\Theta(nk)} = \Theta(p)$$

unde: $\Theta(n)$ = timp de comunicare

$\Theta(nk)$ = timp de calcul

Concluzii

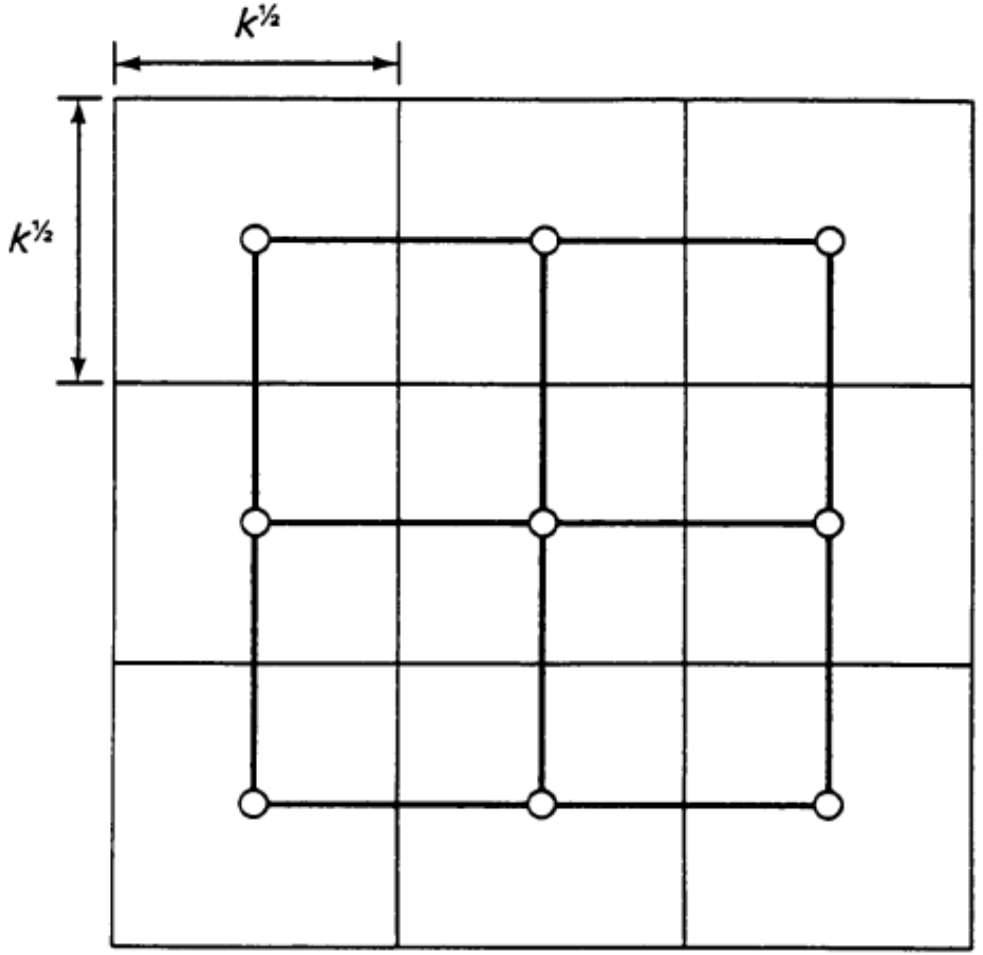
- (pt. metode iterative ca in exemplul nostru)
- penalizarea impusa de comunicare nu impiedica scalarea problemei cu nr de procesoare cand dimensiunea problemei e mare, chiar daca se foloseste cea mai putin performanta topologie de interconectare a procesoarelor
- cresterea dimensiunii n a problemei => cresterea proportionala a nr de procesoare p a.i. nr de variabile k per procesor sa ramana relativ constant (=> comunicatie practic insignifianta)
- Obs: daca folosim un hipercub in loc de lant liniar, timpul optimal de broacast multinod este $\Theta(pk/\log p)$, ceea ce reduce CP de la $\Theta(1/k)$ la $\Theta\left(\frac{1}{k\ log\ p}\right)$

=> daca n creste cu un anumit factor, nr de procesoare p din hipercub poate fi crescut cu un factor si mai mare la un CP relativ nesemnificativ

=> $S_p\ hipercub > S_p\ lant\ liniar$

Concluzii (contd.)

- analiza precedenta nu pp o structura speciala a iteratiei, ci doar ca o singura actualizare de variabile costa $\Theta(n)$ (sau $\Theta(pk)$)
- sunt multe alte cazuri cu structura speciala, in care T_{comm} / T_{calcul} e mic pt valori mari ale lui k
- ex: discretizarea spatiilor fizice bidimensionale (doar variabilele vecine interactioneaza), fiecare dreptunghi contine k variabile
- nr. variabile comunicate de un procesor $\Theta(\sqrt{k})$ cu fiecare dintre vecini
- timp de comunicare per iteratie pe grid/hipercub $\Theta(\sqrt{k})$
- timp constant pt actualizarea fiecarei variabile
- timp paralel de calcul per iteratie $\Theta(k)$
- T_{comm} / $T_{calcul} = \Theta(\frac{1}{\sqrt{k}})$



Concluzii (contd.)

- dincolo de metode iterative, selectia potrivita a dimensiunii taskurilor alocate fiecarui procesor poate minimiza efectele comunicatiei
- daca dimensiunea problemei creste fara limita, si acceleratia poate creste corespunzator daca se alege algoritmul paralel potrivit
  - i.e., nu exista limita a priori impusa de cerintele de comunicatie asupra acceleratiei care se poate atinge