Grundläggande kommunikationsoperationer

Grama et al.

Introduction to Parallel Computing
Kapitel 4

Robert Granat

Översikt

Kommunikationsoperationer

- Enkla meddelanden
  - 1-till-1
- Broadcast (utsändning)
  - 1-till-alla
  - alla-till-alla
- Reduktion, prefix summering
- Personlig kommunikation
  - 1-till-alla
  - alla-till-alla
- Cirkulära shift

Topologier

- Ring
- 2D-nät
- 3D-nät
- Hyperkub

Routing-tekniker

- Store-and-forward (SF)
- Cut-through (CT)

Kommunikationskostnader

- t_s: start-upp-tid
  - Preparera meddelande & header
  - Exekvera routing-algoritm
- t_h: per-hopp-tid
  - Tid för header från en processor till granne
- t_w: per-ord-tid
  - tid för överföring per ord i meddelandet
  - om bandbredd *r* ord/s så är $t_w = 1/r$

Routing-tekniker

Meddelande av storlek m skickas över l länkar

Store-and-forward (lagra-vidarebefordra)

- Varje länk tar tiden t_h för headern och $m\ t_w$ för resten

  $\Rightarrow t_{comm} = t_s + l(t_h + m\ t_w)$

  (förenklas ofta till $t_{comm} = t_s + l\ m\ t_w$ då t_h litet jämfört med m)

Cut-through (direkt-igenom)

- Tar totalt tiden tiden $l\ t_h$ för headern

  Hela meddelandet anländer $m\ t_w$ efter headern anlänt

  $\Rightarrow t_{comm} = t_s + l\ t_h + m\ t_w \approx t_s + m\ t_w$

Teknikerna likvärdiga om $l = 1$ eller m litet

Förutsättningar & antaganden

- Logiska topologier: ring, 2-dimensionellt nät, hyperkub
- Dubbelriktade länkar
- 2 proc. kan samtidigt skicka meddelande av storlek *m* till varandra på tiden $t_s + m\ t_w$
- t_s = startupptid, t_w = per-ord-tid
- 1 Processor kan endast
  - sända på en länk i taget
  - ta emot på en länk i taget

  ... men ...
- 1 processor kan både skicka och ta emot samtidigt (1 eller olika länkar), jmf MPI_Sendrecv

Tid för ett meddelande (1 proc till 1 annan)

Lagra-vidarebefordra (store-and-forward)

- Storlek m över l länkar: $t_s + m\, l\, t_w$
  - l är maximalt: $\lfloor p/2 \rfloor$ för ring
  - l är maximalt: $2 \lfloor p^{1/2}/2 \rfloor$ för 2-dim nät (kvadr., wrap-around)
  - l är maximalt: log p för en hyperkub
- Övre gräns för ett meddelande
  - $t_s + m\, t_w \lfloor p/2 \rfloor$ för ring
  - $t_s + m\, t_w \lfloor p^{1/2}/2 \rfloor$ för 2-dim nät
  - $t_s + m\, t_w \log p$ för en hyperkub

Direkt-igenom (cut-through)

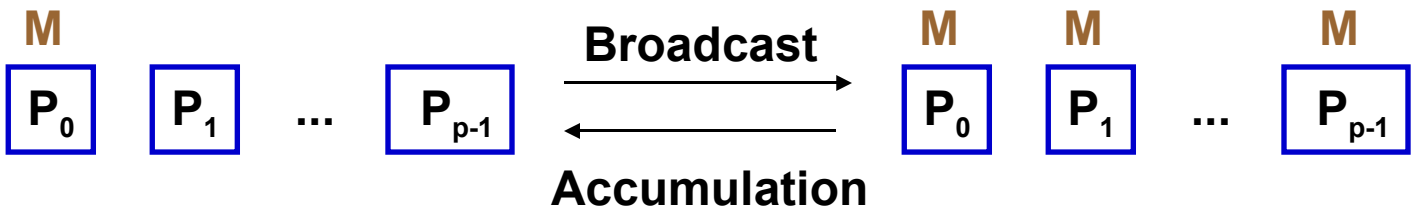
- $t_s + m\, t_w + l\, t_h \approx t_s + m\, t_w$, där t_h är tid för ett hopp

1-till-alla utsändning (broadcast)

- 1 skickar samma data till alla
- Ex: matris-vektormult, gausselimination, kortaste vägen
- Dual operation: 1-nods-ackumulering (sum, max, etc.)

Lagra-vidarebefordra (store-and-forward)

- Undvik onödiga meddelanden:
  - källprocessor skickar till de den når direkt (grannar)
  - de som fått meddelande i ett steg: skicka vidare i nästa steg



M: m ords meddelande

1-till-alla på ring

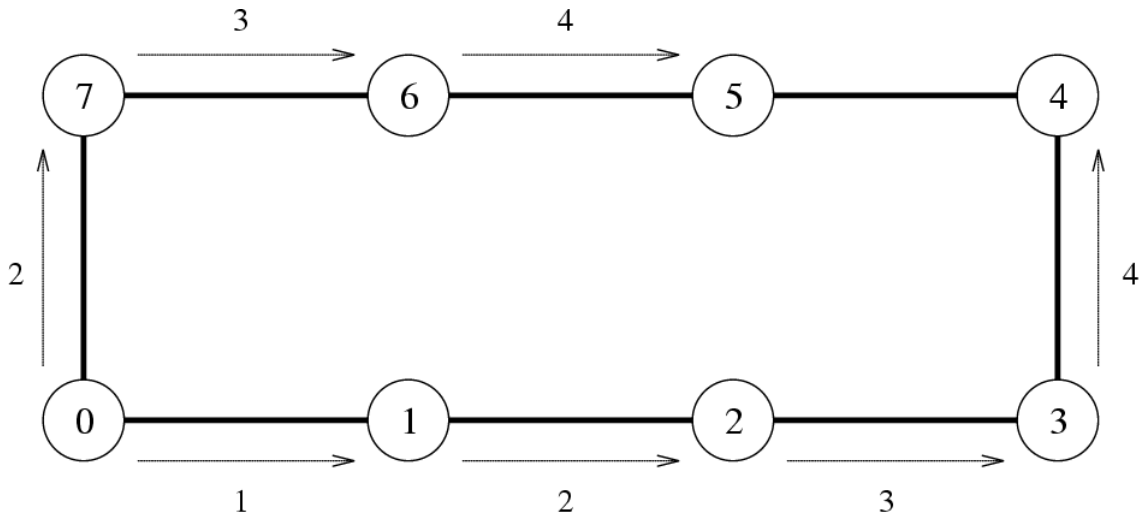


Figure 3.2 One-to-all broadcast on an eight-processor ring with SF routing. Processor 0 is the source of the broadcast. Each message transfer step is shown by a numbered, dotted arrow from the source of the message to its destination. The number on an arrow indicates the time step during which the message is transferred.

1-till-alla på ring

Lagra-vidarebefordra

- Tar totalt $\lceil p/2 \rceil$ steg
- Varje steg tar $t_s + m\, t_w$ tid

Totalt krävs $(t_s + m\, t_w)\lceil p/2 \rceil$ tid

Hur kan detta nyttjas i en kolumn av ett 2-dimensionellt nät?

1-till-alla på 2-dim nät

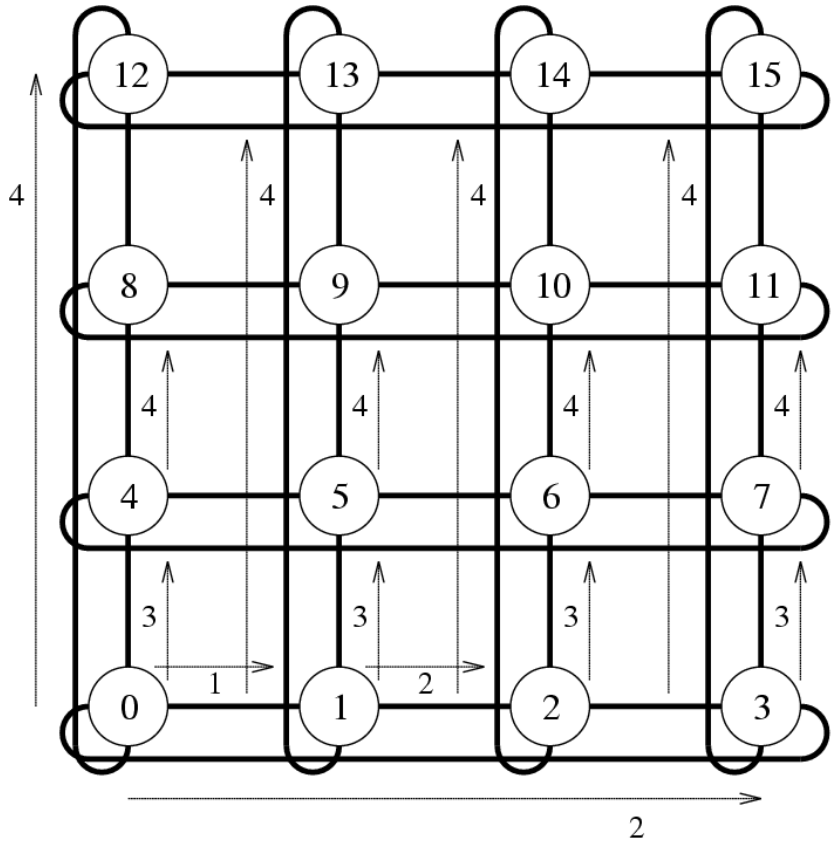


Figure 3.4 One-to-all broadcast on a 16-processor mesh with SF routing.

1-till-alla på kvadratiskt 2-dim nät

Lagra-vidarebefordra

- Broadcast i startnodens processorrad

  tid: $(t_s + m\, t_w)\lceil p^{1/2}/2 \rceil$

- Varje processor i startnodens rad startar broadcast i sin processorkolumn (// i alla kolumner)

  tid: $(t_s + m\, t_w)\lceil p^{1/2}/2 \rceil$

Totalt krävs $2\,(t_s + m\, t_w)\lceil p^{1/2}/2 \rceil$ tid

3-dim nät:

- Samma algoritm i tre steg - 1 för varje dimension:

Totalt krävs $3\,(t_s + m\, t_w)\lceil p^{1/3}/2 \rceil$ tid

1-till-alla på hyperkub

Lagra-vidarebefordra

- Betrakta 2^d-kub som *d*-dimensionellt nät med 2 processorer i varje dimension
- Samma algoritm, nu i *d* steg (= *log p*)
- Varje steg är enbart en vanlig sändning
  (1-till-alla broadcast i ring med enbart 2 processorer)

Totalt krävs $(t_s + m\,t_w)\log p$

1-till-alla på hyperkub

2^3-kub $\rightarrow$ d = 3

3D nät

2 proc i varje dim

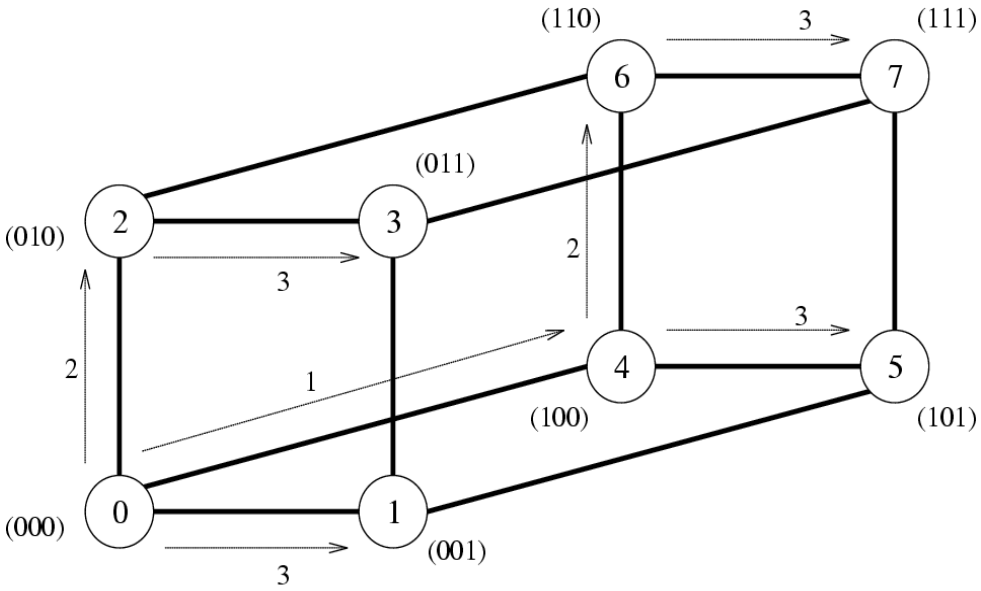


Figure 3.5 One-to-all broadcast on a three-dimensional hypercube. The binary representations of processor labels are shown in parentheses.

Källa: proc 0

I steg i: skicka till processor som skiljer sig åt i bit i

Börja (t.ex.) med mest signifikanta biten

1-till-alla på hyperkub - algoritm

```
1.  procedure ONE_TO_ALL_BC(d, my_id, X)
2.  begin
3.      mask := 2^d − 1;              /* Set all d bits of mask to 1 */
4.      for i := d − 1 downto 0 do    /* Outer loop */
5.      begin
6.          mask := mask XOR 2^i;     /* Set bit i of mask to 0 */
7.          if (my_id AND mask) = 0 then
            /* If the lower i bits of my_id are 0 */
8.              if (my_id AND 2^i) = 0 then
9.              begin
10.                 msg_destination := my_id XOR 2^i;
11.                 send X to msg_destination;
12.             endif
13.             else
14.             begin
15.                 msg_source := my_id XOR 2^i;
16.                 receive X from msg_source;
17.             endelse;
18.     endfor;
19. end ONE_TO_ALL_BC
```

Program 3.1 One-to-all broadcast of a message X from processor 0 of a d-dimensional hypercube. AND and XOR are bitwise logical-and and exclusive-or operations, respectively.

1-till-alla utsändning - (cut-through)

- Hyperkub: ingen tidsvinst
- Ring: använd hyperkubsalgoritmen
  - Tid för ett meddelande begränsas av θ(m+l)
  - 1:a steget : l = p/2
  - 2:a steget: l = p/4 etc
- Totalt krävs:

$$T = \sum_{i=1}^{\log p} (t_s + t_w m + t_h p/2^i)$$
$$= t_s \log p + t_w m \log p + t_h (p-1)$$

Hint :

$$\sum_{i=1}^{\log p} 1/2^i = (p-1)/p$$

1-till-alla på ring med hyperkubsalgoritm

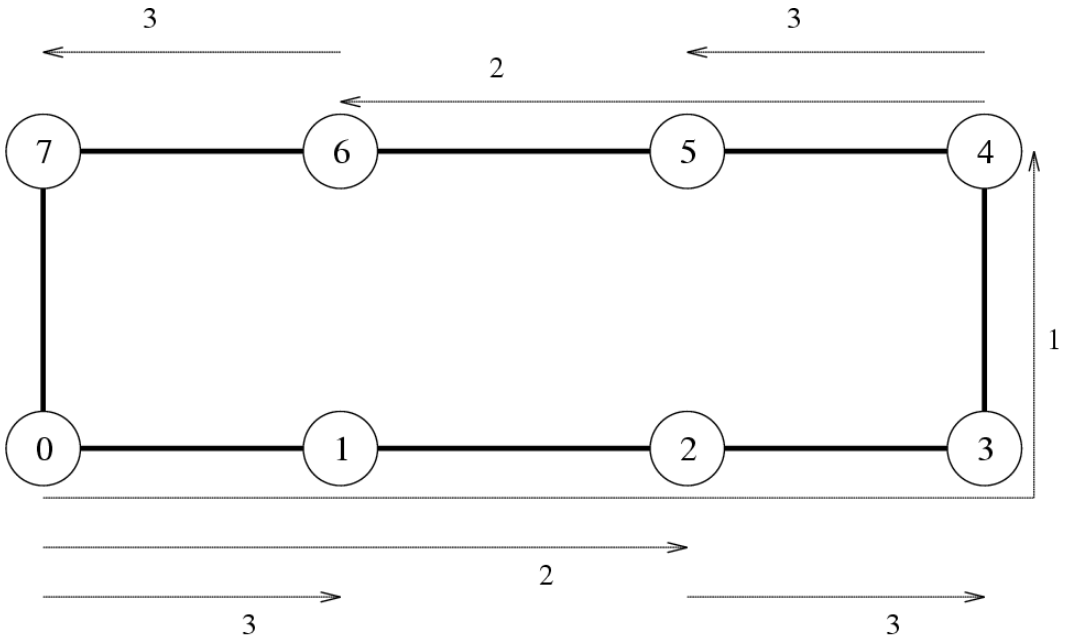


Figure 3.6 One-to-all broadcast with CT routing on an eight-processor ring.

Alla-till-alla broadcast

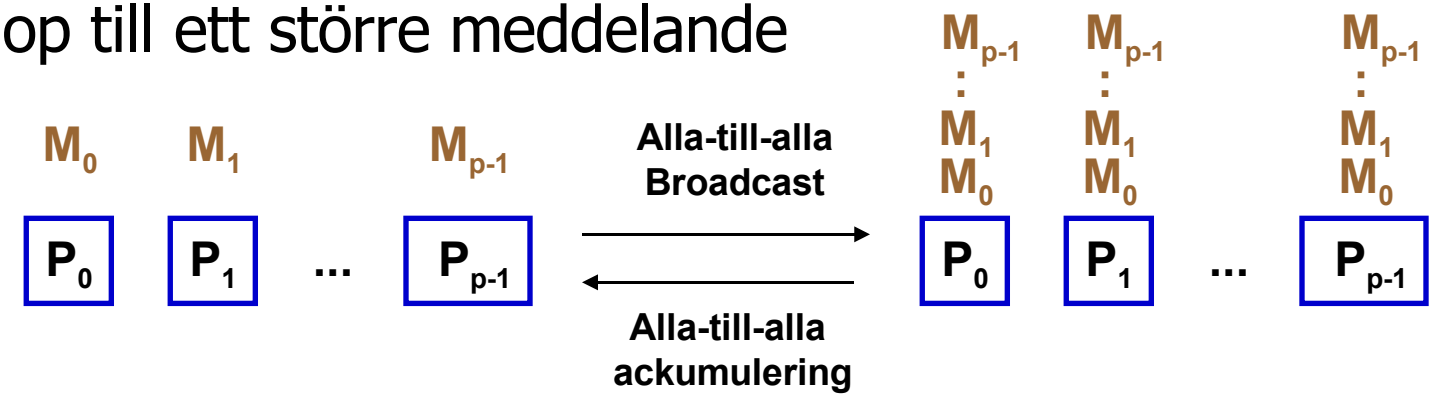
- *p* processorer gör samtidigt broadcast (olika meddelanden från olika proc.)
- Dual operation: alla-till-alla ackumulering

Naiv metod:

- Alla gör 1-till-alla utsändning

Effektivare metod:

- Meddelanden som i ett steg skall gå längs samma väg slås ihop till ett större meddelande



Alla-till-alla på ring

- 1-till-alla utsändning ger utnyttjande av maximalt 2 länkar samtidigt
- För alla-till-alla utsändning har varje processor i varje steg något den kan vidarebefordra

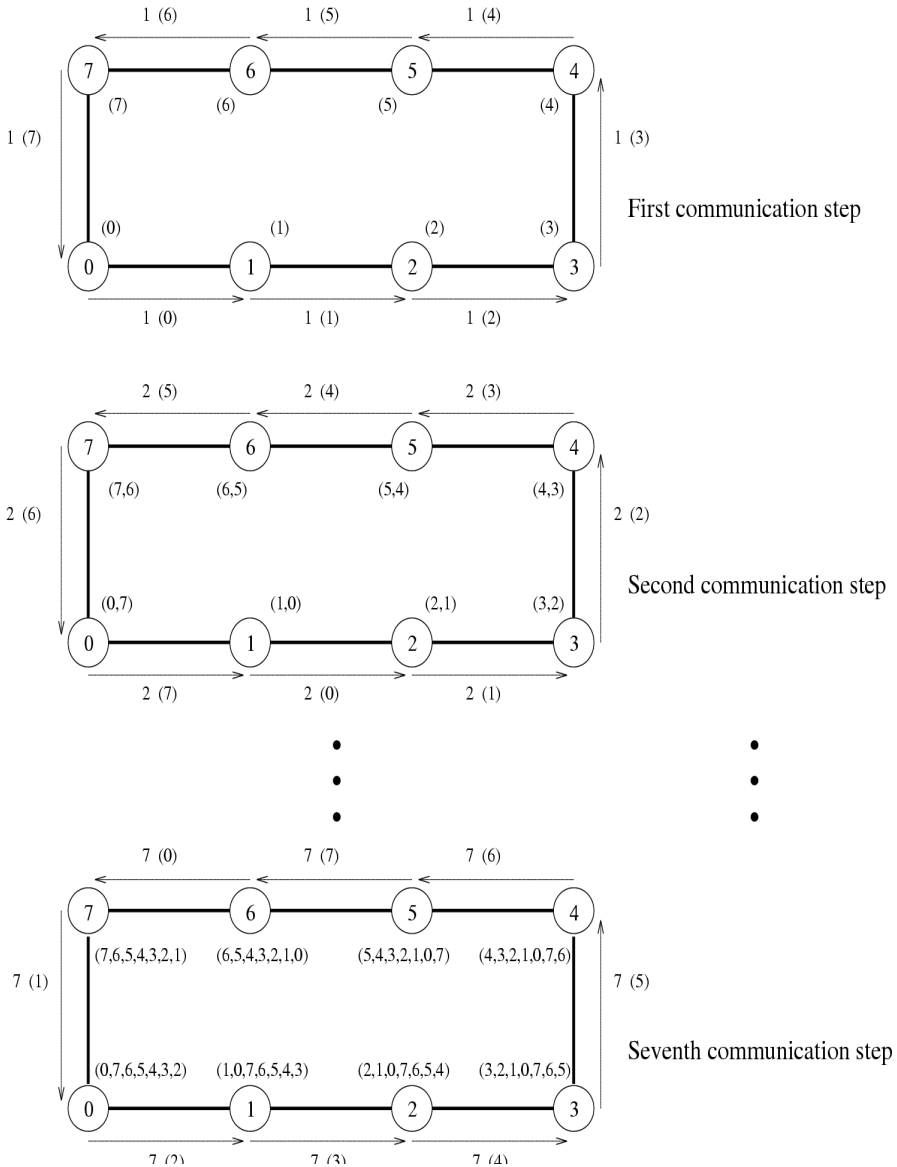
Steg 1: Varje processor skickar sitt meddelande till en granne

Steg 2, 3...: Varje processor skickar vidare det den tog emot i förra steget

- Varje processor tar emot p-1 olika meddelanden i p-1 steg

Totalt krävs: $(t_s + m\, t_w)\,(p-1)$

Alla-till-alla på ring



Alla-till-alla på 2-dim nät

Steg 1:

- Alla-till-alla utsändning i varje processorrad

  Tid = $(t_s + t_w m)(p^{1/2}-1)$

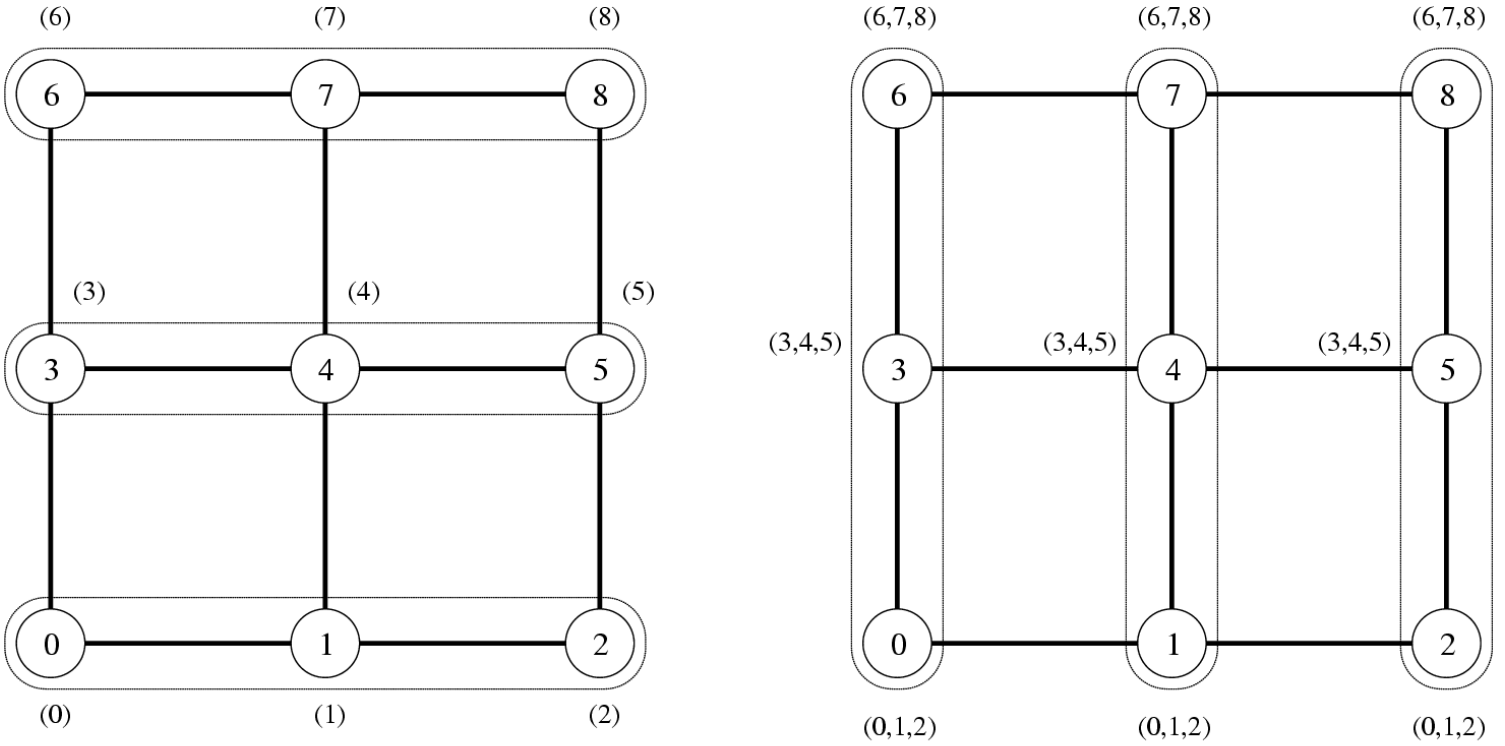
Steg 2

- Alla-till-alla utsändning i varje processorkolumn (varje meddelande är $m\ p^{1/2}$

  Tid = $(t_s + t_w m\ p^{1/2})(p^{1/2}-1)$

Totalt krävs: $2\ t_s(p^{1/2}-1) + t_w m (p-1)$

Alla-till-alla på 2-dim nät



(a) Initial data distribution

(b) Data distribution after rowwise broadcast

Figure 3.11 All-to-all broadcast on a 3×3 mesh. The groups of processors communicating with each other in each phase are enclosed by dotted boundaries. By the end of the second phase, all processors get (0,1,2,3,4,5,6,7) (that is, a message from each processor).

Alla-till-alla på hyperkub

- 2-dim-nät-algoritmen generaliserad till *log p* dimensioner (i *log p* tidssteg)
- I varje steg utbyter par av processorer meddelanden
- Meddelandena slås ihop till ett (dubbel storlek) i nästa steg

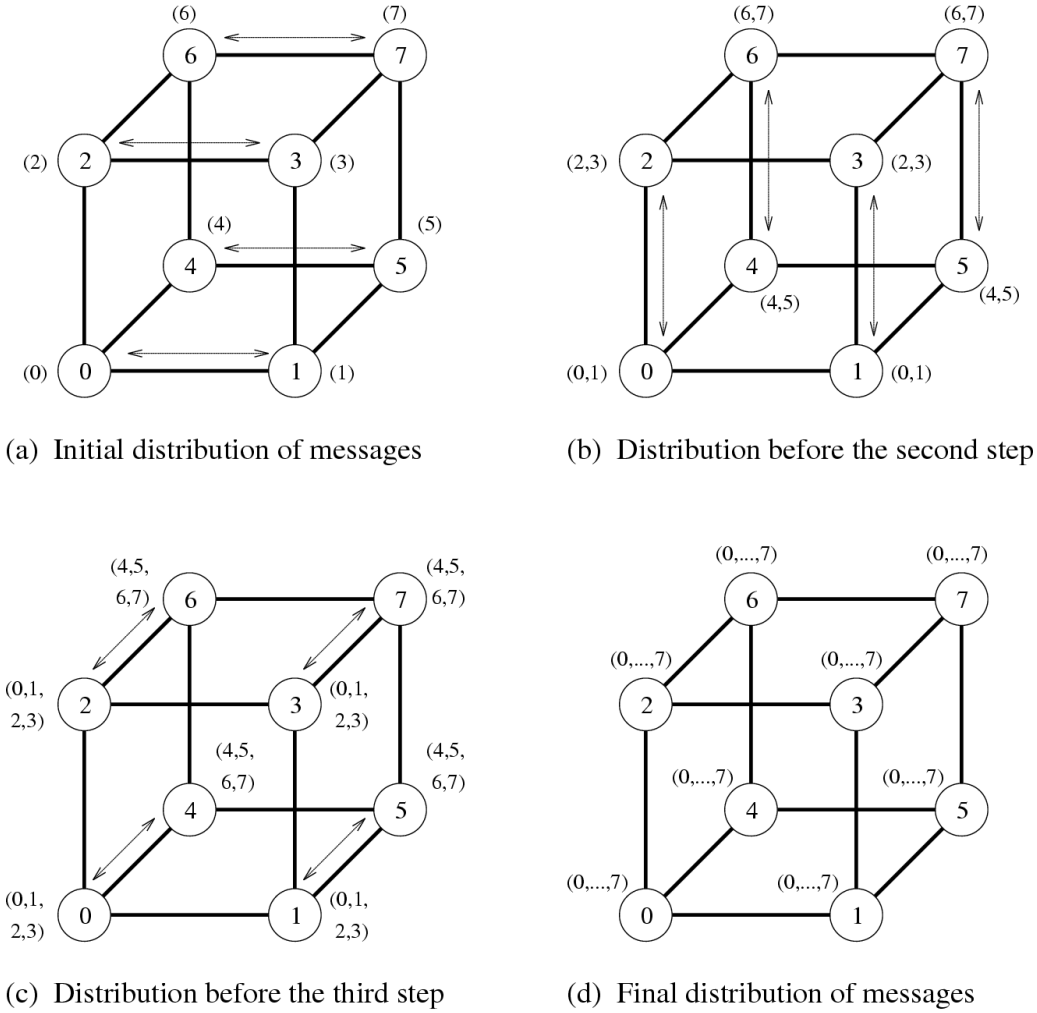
- Storlek av meddelande i steg *i* är $2^{i-1}m$
- Tid för detta: $t_s + 2^{i-1} t_w m$

Totalt krävs: $\sum_{i=1}^{\log p} (t_s + 2^{i-1} t_w m) = t_s \log p + t_w m(p-1)$

Hint:

$$\sum_{i=0}^{\log p - 1} 2^i = p - 1$$

Alla-till-alla på hyperkub



(a) Initial distribution of messages

(b) Distribution before the second step

(c) Distribution before the third step

(d) Final distribution of messages

Figure 3.12 All-to-all broadcast on an eight-processor hypercube.

Alla-till-alla broadcast på hyperkub

```
1.   procedure ALL_TO_ALL_BC_HCUBE(my_id, my_msg, d, result)
2.   begin
3.       result := my_msg;
4.       for i := 0 to d do
5.       begin
6.           partner := my_id XOR 2^i;
7.           send result to partner;
8.           receive msg from partner;
9.           result := result ∪ msg;
10.      endfor;
11.  end ALL_TO_ALL_BC_HCUBE
```

Program 3.6 All-to-all broadcast on a d-dimensional hypercube.

(Över alla dimensioner med början från den lägsta dimensionen)

Reduktion på hyperkub

Exempel: global summering

- Alla processorer har ett värde
- Alla processorer vill veta summan av alla processorers motsvarande värde
- Andra operationer: max, min, OR, AND etc.

- **Alternativ 1:** alla-till-1 ackumulering följt av summering på en nod och 1-till-alla utsändning $\rightarrow$ $2\,(t_s + m\,t_w)\,\log p$
- **Alternativ 2:** Kommunikationsmönster som i alla-till-alla utsändning, men istället för att konkatenera meddelanden så adderas resultatet (eller min, max etc).
  - Varje meddelande är endast av storlek m

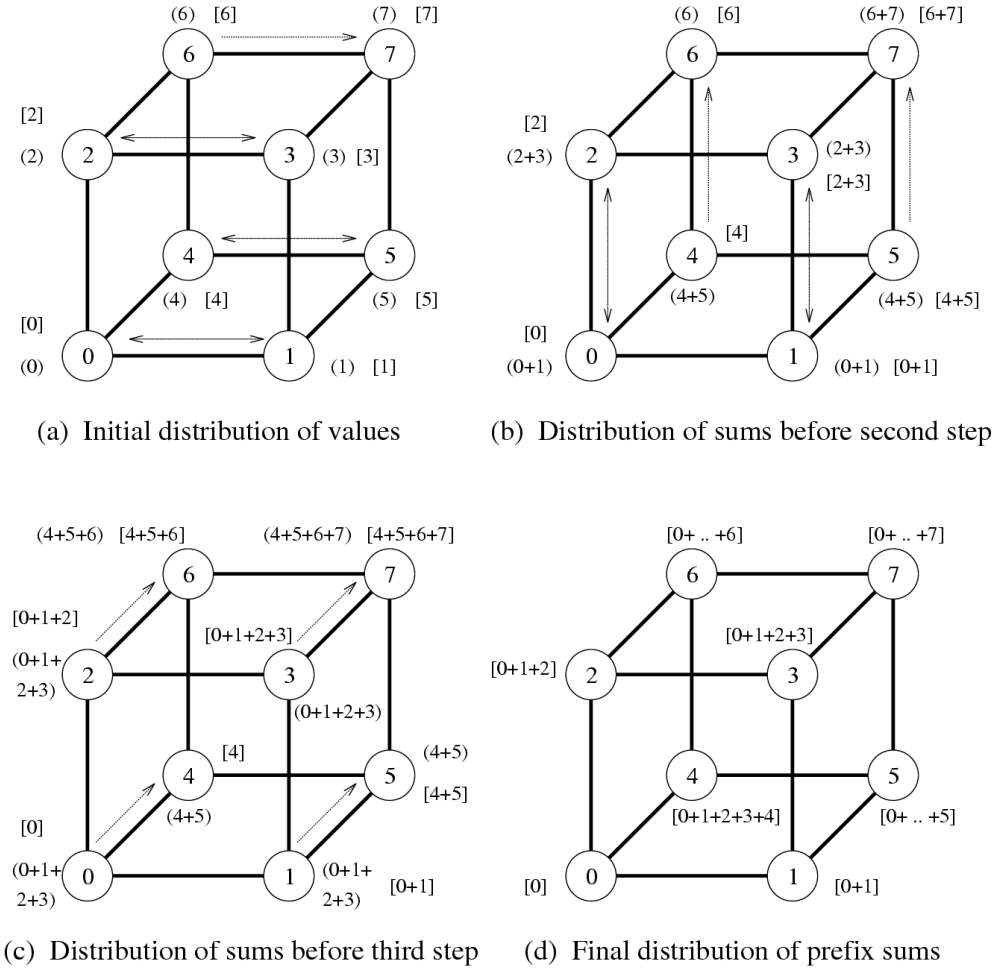
Totalt krävs: $(t_s + m\,t_w)\,\log p$ (m = 1 om reduktion av ett tal)

Prefix summering på hyperkub

- Givet p tal, $n_0, n_1, \ldots, n_{p-1}$, (ett på varje processor)
- Beräkna $s_k = n_0 + \ldots + n_k$, för alla $k = 0, \ldots p-1$, så att s_k ligger på processor p_k

- Som global summering (reduktion med +), men spara varje processors delresultat i speciell buffert
- Inkommande resultat adderas till buffert endast om avsändare är processor med lägre processornummer

Prefix summering på hyperkub



(a) Initial distribution of values

(b) Distribution of sums before second step

(c) Distribution of sums before third step

(d) Final distribution of prefix sums

Figure 3.13 Computing prefix sums on an eight-processor hypercube. At each processor, square brackets show the local prefix sum accumulated in a buffer and parentheses enclose the contents of the outgoing message buffer for the next step.

Alla-till-alla utsändning med cut-through

- Hyperkubsalgoritmen på ring och 2-dimensionellt nät ger ej förbättrad prestanda p g a kollisioner
- Generellt ej bättre prestanda med direkt-igenom för dessa kommunikationsmönster

- För alla arkitekturer är $mt_w(p-1)$ undre gräns för tid att utföra alla-till-alla utsändning (mängden data alla tar emot)

Med lagra-vidarebefordra fick vi:

- ring: $(t_s + m\,t_w)\,(p-1)$
- 2-dim nät: $2\,t_s(p^{1/2}-1) + m\,t_w\,(p-1)$
- hyperkub: $t_s \log p + m\,t_w\,(p-1)$

1-till-alla personlig kommunikation

- Benämns även single-node scatter
- 1 proc skickar olika meddelanden till alla
- Dual operation: single-node gather
- Komplexitet ungefär som alla-till-alla utsändning: undre gräns ges av *p-1* meddelanden av storlek *m*, dvs $mt_w(p-1)$

På hyperkub:

- Steg 1: 1 proc skickar hälften av meddelandena till granne
- Steg 2 ,..,log p: skicka hälften av meddelanden till granne
- Kommunikation som 1-till-alla utsändning, större meddel.
- Tid = $t_s \log p + mt_w(p-1)$

(Samma tid som för alla-till-alla broadcast)

1-till-alla personlig komm.:hyperkub

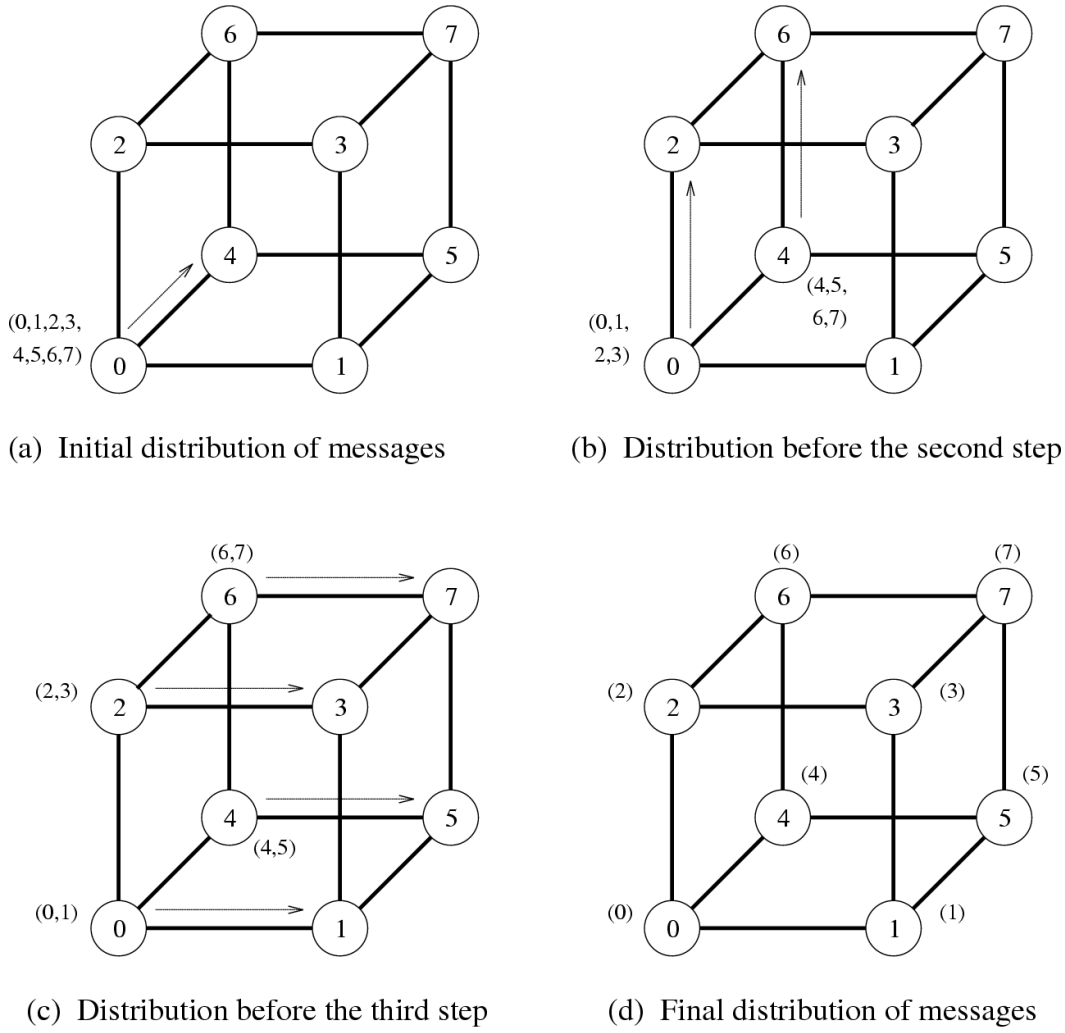


Figure 3.16 One-to-all personalized communication on an eight-processor hypercube.

Alla-till-alla personlig kommunikation

- Kallas även *total exchange*
- Alla skickar olika meddelande till varje annan processor
- Kommunikationsmönster som alla-till-alla utsändning för samtliga 3 arkitekturer (endast storlek skiljer)

Ring:

- Steg 1: Varje processor skickar alla *p-1* meddelanden till granne (samma riktning)
- Steg *2,..., p-1*: Varje processor behåller sin del av inkommet meddelande och skickar vidare resten
- I steg i skickar alla meddelande av storlek *m(p-i)* till granne

Totalt krävs:

$$\sum_{i=1}^{p-1}(t_s + t_w m(p-i)) = t_s(p-1) + \sum_{i=1}^{p-1} i t_w m$$

$$= (t_s + \frac{1}{2} t_w mp)(p-1)$$

Alla-till-alla personlig kommunikation: Ring

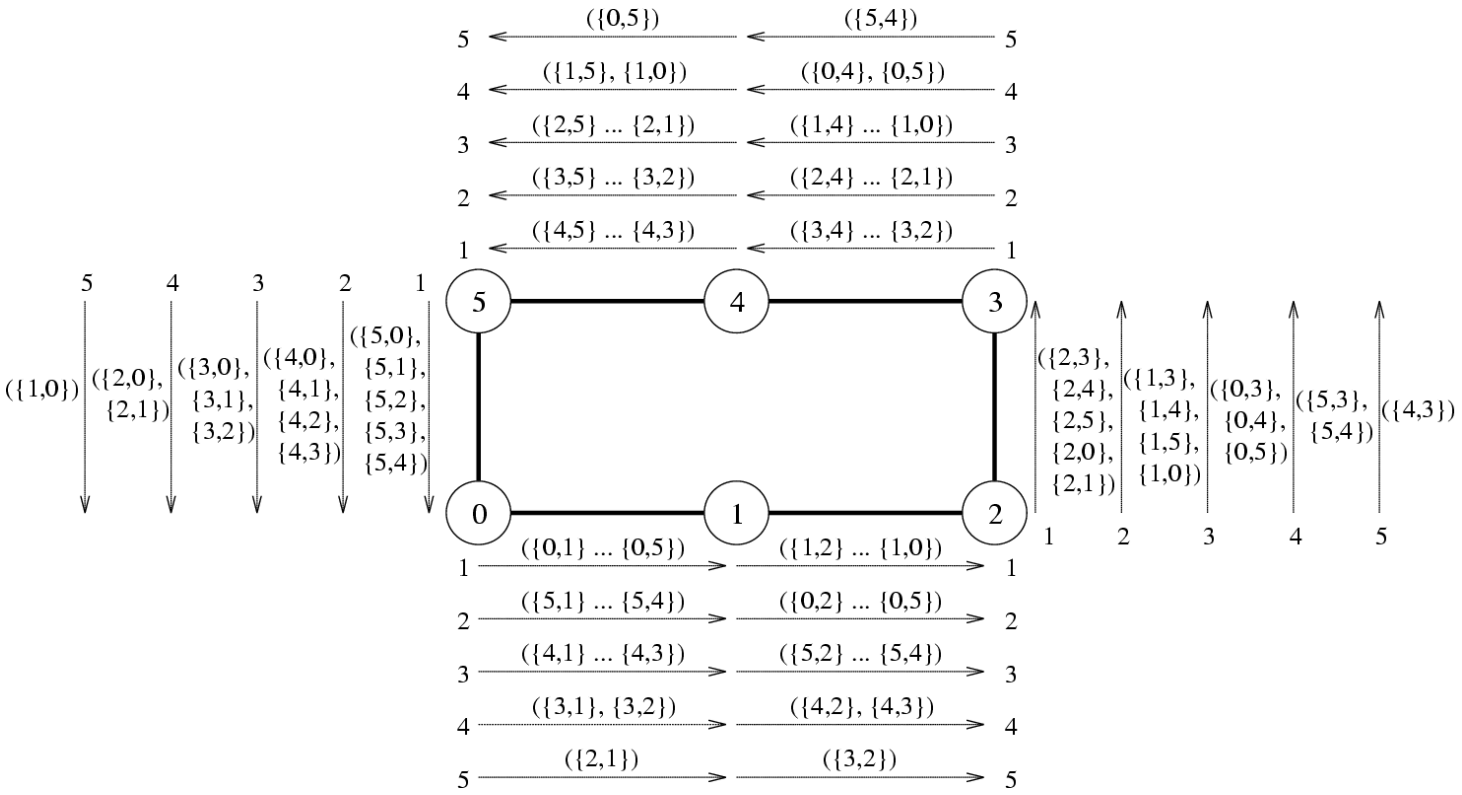


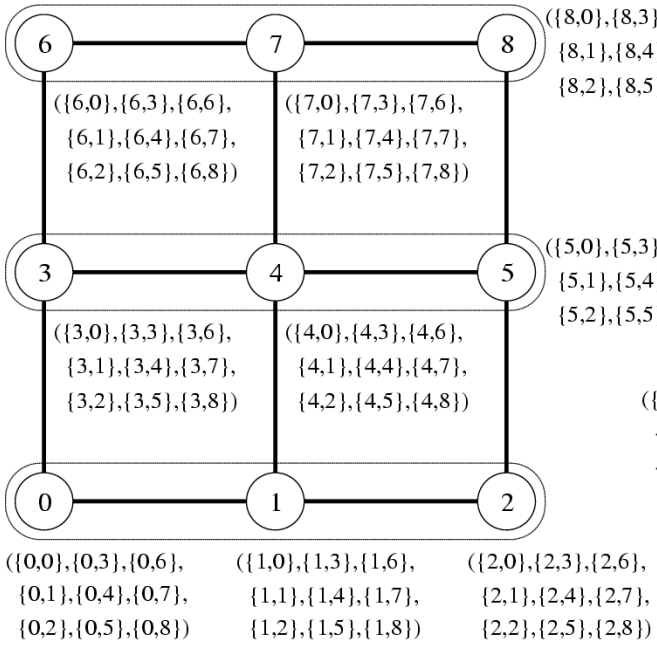
Figure 3.18 All-to-all personalized communication on a six-processor ring. The label of each message is of the form $\{x, y\}$, where x is the label of the processor that originally stored the message, and y is the label of the processor that is the final destination of the message. The label $(\{x_1, y_1\}, \{x_2, y_2\}, \ldots, \{x_n, y_n\})$ indicates a message that is formed by concatenating n individual messages.

Alla-till-alla personlig kommunikation: 2D nät

- Varje processor grupperar data efter mottagarprocessors kolumnplacering
- Steg 1: alla-till-alla personlig kommunikation i varje processorrad. Tid som för ring med $p^{1/2}$ processorer och meddelandestorlek $mp^{1/2}$
- Steg 2: Motsvarande kommunikation i processorkolumn tar samma tid som steg 1
  - Varje processor grupperar data efter ”mottagarens” radplacering

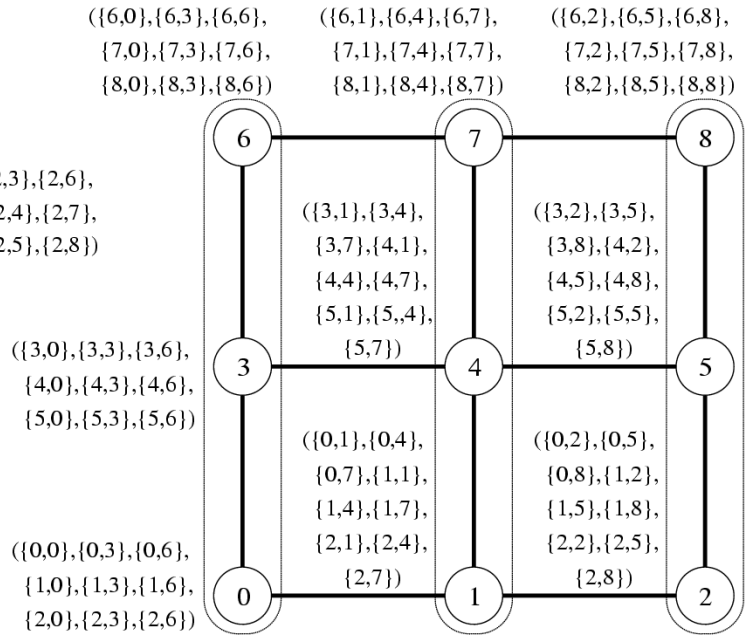
Totalt krävs: $(2t_s + t_w mp)(p^{1/2} - 1)$

Alla-till-alla personlig komm.: 2-dim nät



(a) Data distribution at the beginning of first phase

{x,y}: meddelande ursprungligen från proc x med slutlig destination proc y



(b) Data distribution at the beginning of second phase

Alla-till-alla personlig komm.: hyperkub

- Generalisering av 2-dim-nät-algoritmen till *log p*-dim nät som i *log p* steg utbyter data av storlek *mp/2* mellan 2 processorer varje gång

Totalt krävs: $(t_s + t_w mp/2) \log p$

Alla-till-alla personlig komm.: hyperkub

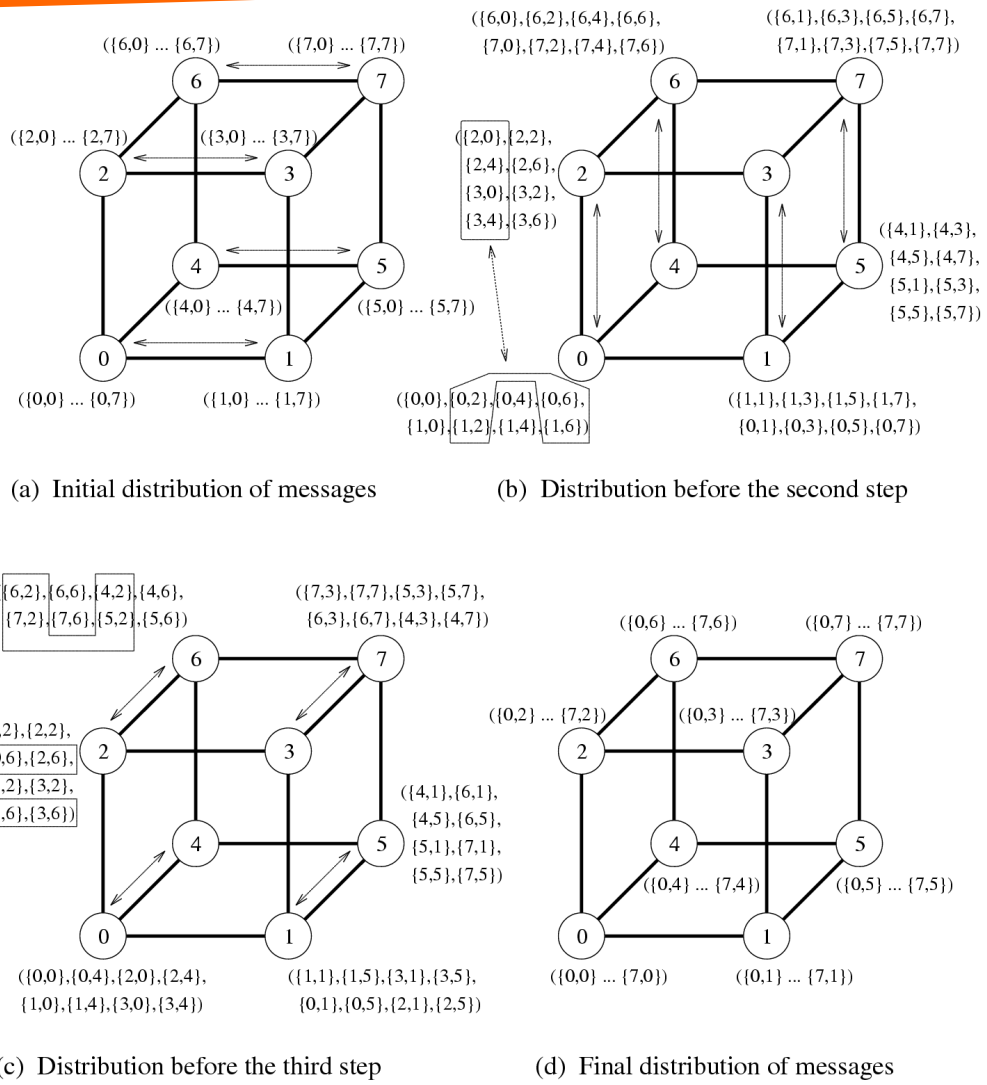


Figure 3.20 All-to-all personalized communication on a three-dimensional hypercube with SF routing.

Alla-till-alla personlig komm.: Cut-through

Ring (och 2-dim nät)

- Varje proc skickar totalt meddelanden av storlek $m(p-1)$
- Genomsnittligt avstånd är $p/2$
- p processorer gör samma sak
- Total nätbelastning = $p\ m(p-1)\ p/2$
- Antal länkar som delar på belastningen är p
- En undre gräns ges av $t_w\ m(p-1)\ p/2$
- Om startupp-tid försummas är detta samma som SF-algoritmerna (som alltså ej är långt ifrån optimala)

- Notera: Vi har endast använt oss av 1 riktning i varje ring. Om båda riktningarna utnyttjas kan tiden för t_w halveras både i algoritmerna och i den undre gränsen!

Alla-till-alla personlig komm.: Cut-through

Hyperkub:

- Genomsnittlig avstånd = *(log p)/2*, total storlek= *m(p-1)* för varje av *p* processorer ger **undre gräns:** $(t_w m(p-1)(p \log p)/2) / ((p \log p)/2) = t_w m(p-1)$

Bättre algoritm med cut-through:

- **Steg i:** varje processor skickar direkt meddelande till proc vars *i* minst signifikanta bitar skiljer sig från de egna
- Tid att skicka över *l* hops = $t_s + t_w m + t_h l$
- Summan av alla *l*: $0+1+\ldots+p-1 = (p \log p)/2$

Totalt krävs: $(t_s + t_w m)(p-1) + t_h(p \log p)/2$

Alla-till-alla personlig komm.: Cut-through

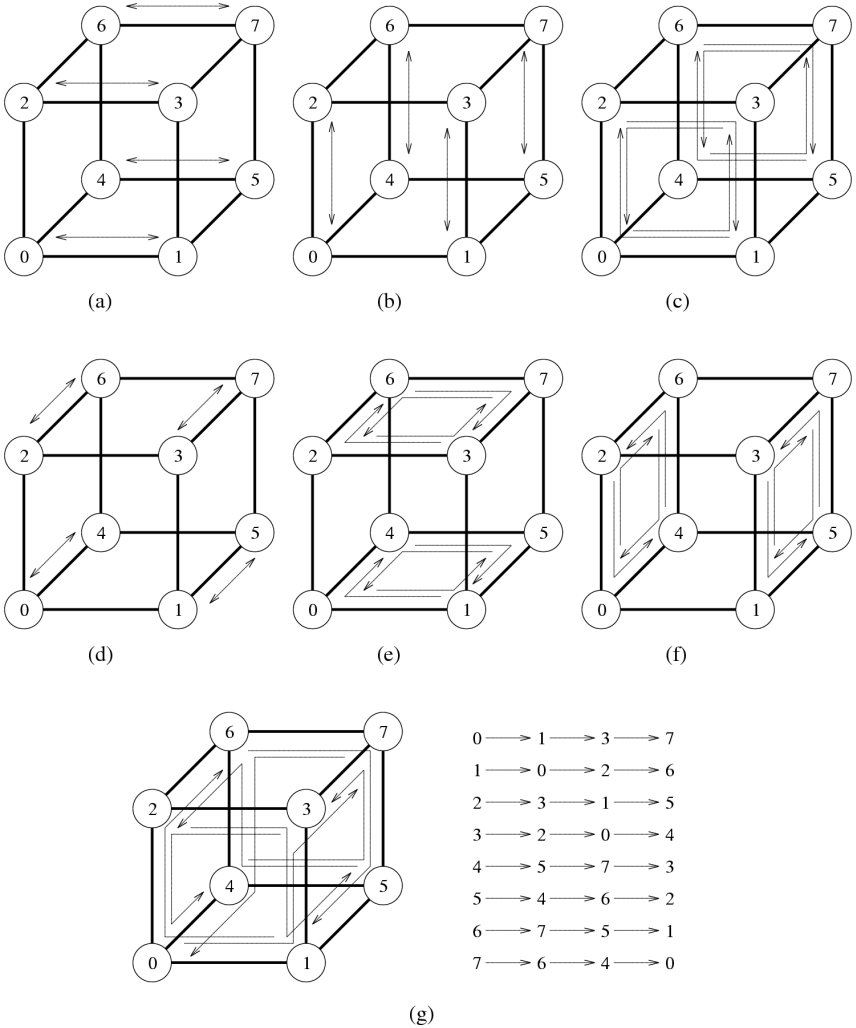


Figure 3.21 Seven steps in all-to-all personalized communication on an eight-processor hypercube with CT routing.

Inga kollisioner
Parvis utbyte
Dubbelriktade länkar

Cirkulär skiftning

- Processor i skickar till processor (i +q) mod p $(0 < q < p)$

2-dim nät:

- **Steg 1:** Allt data skickas (q mod $p^{1/2}$) steg längs raderna.
- **Steg 2:** Allt data skickas $\lfloor q/p^{1/2} \rfloor$ längs kolumnerna

- **Steg 1.5:** I steg 1 skiftades en del data över wrap-around-länken till processor med lägre nummer. Allt detta data skall som korrigering skickas 1 steg längs kolumnerna.

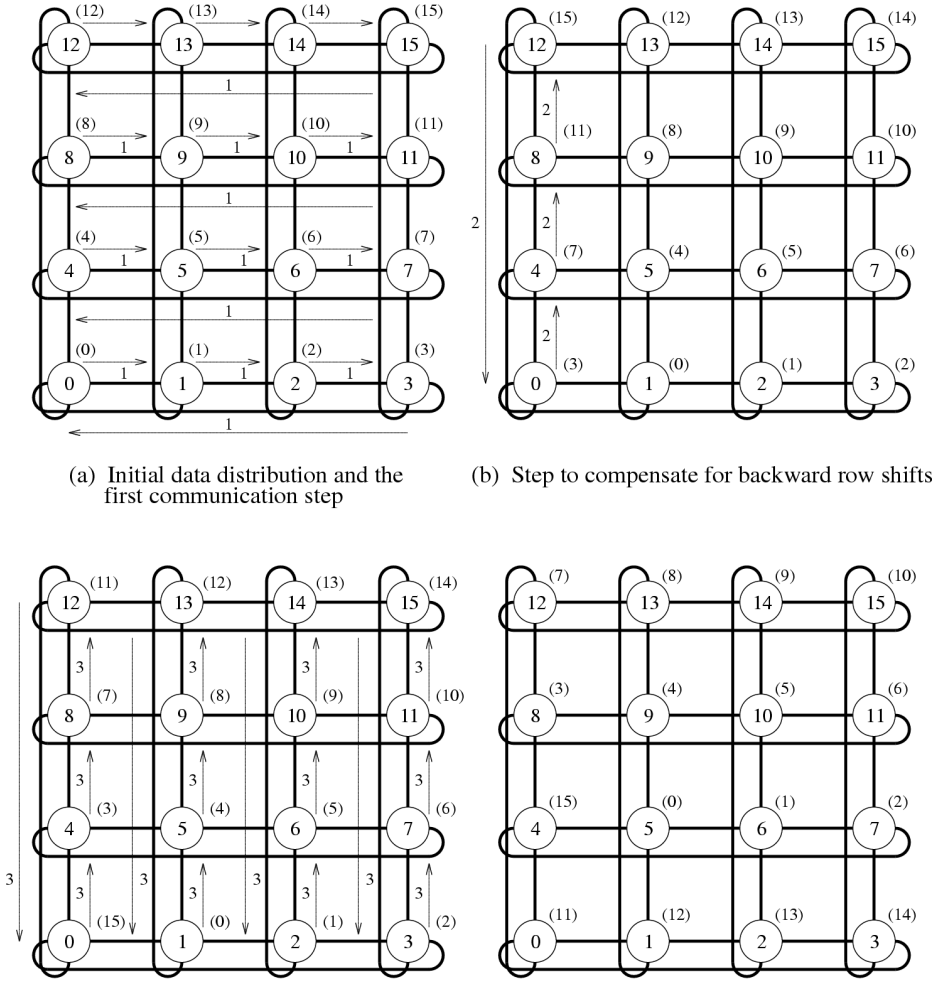
- **Förbättring:** Skift görs i den riktning det är närmast (t ex i 4-processors rad motsvarar ett 3-skift ett 1-stegs bakåtskift). Detta ger maximalt $\lfloor p^{1/2}/2 \rfloor$ steg i varje riktning

Cirkulär skiftning på 2-dim nät

Steg 1

Steg 1.5

q=5
p=16



Steg 2

Figure 3.22 The communication steps in a circular 5-shift on a 4×4 mesh.

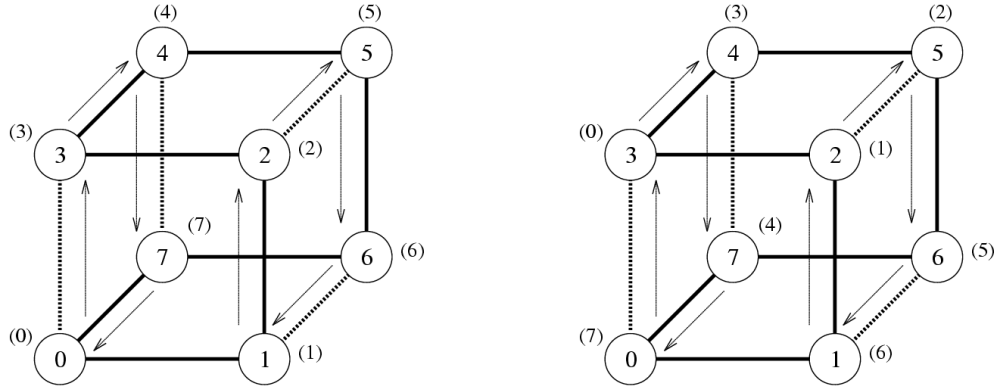
Cirkulär skiftning på hyperkub

- Avbilda en ring på kub med processorer numrerade i BRGC-ordning
- Processorer med differens 2^0 är grannar
- Processorer med differens 2^i, i > 0 är på avstånd 2
- Skiftning i q steg görs genom att q tolkas binärt och ett stegs skift görs för varje 1:a i den binära koden
- T ex q = 5, ger binär kod = 101. Vi skiftar fyra processornummer för 100 och ett 1-skift för 001
- Maximalt antal steg är log p, där alla steg utom granne-granne är över två länkar

Övre gräns för tid: $(t_s + t_w m)(2 \log p - 1)$

- Genom att utnyttja både bakåt och framåt-skift kan detta minskas till $(t_s + t_w) \log p$

Cirkulär skiftning på hyperkub



(a) The first phase (a 4-shift)

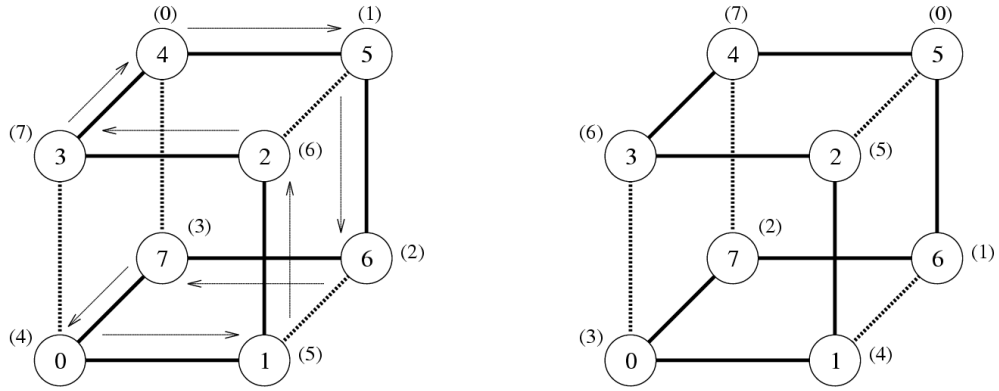


Figure 3.23 The mapping of an eight-processor ring onto a three-dimensional hypercube to perform a circular 5-shift as a combination of a 4-shift and a 1-shift.

Optimering av 1-till-1-kommunikation

Skicka ett meddelande i delar på hyperkub

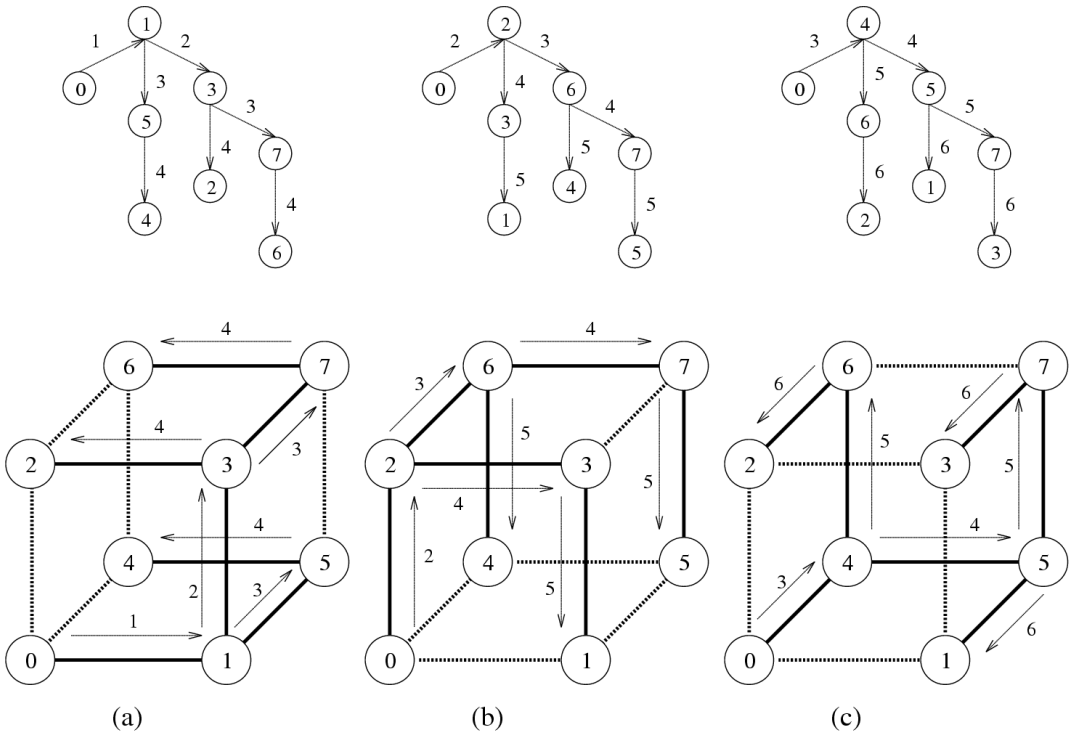
- Mellan varje par av noder existerar log p distinkta vägar, varav l stycken är av längden l och log p - l är av längden l+2 (där l är antal olika bitar i processornumren)
- Meddelandet kan delas i log p delar och skickas längs olika vägar med lång väg först
- Sista paketet tar maximalt log p steg

Totalt krävs maximalt: $2(t_s \log p + t_w m)$

- I vår tidigare algoritm var övre gränsen $t_s + t_w m \log p$

Optimering av 1-till-alla-utsändning

- p stycken olika uppspännade binomiala träd kan avbildas på en hyperkub med samma nod som rot
- Meddelande som skall skickas delas i log p delar. En del skickas längs varje träd



Sammanfattning av kommunikationstider

Table 4.1 Summary of communication times of various operations discussed in Sections 4.1–4.7 on a hypercube interconnection network. The message size for each operation is m and the number of nodes is p.

Operation	Hypercube Time	B/W Requirement
One-to-all broadcast, All-to-one reduction	$\min((t_s + t_w m) \log p, 2(t_s \log p + t_w m))$	$\Theta(1)$
All-to-all broadcast, All-to-all reduction	$t_s \log p + t_w m(p-1)$	$\Theta(1)$
All-reduce	$\min((t_s + t_w m) \log p, 2(t_s \log p + t_w m))$	$\Theta(1)$
Scatter, Gather	$t_s \log p + t_w m(p-1)$	$\Theta(1)$
All-to-all personalized	$(t_s + t_w m)(p-1)$	$\Theta(p)$
Circular shift	$t_s + t_w m$	$\Theta(p)$