TEMPUS: Activity 2

Parallel Algorithms,
Chapter 1 Fundamentals
Section 3 Basic Communication Operations

István Erényi, KFKI-MSZKI

20th July 1997

Overview

Communication Operations
1 Introduction

Executing parallel algorithms processors exchange data

Data exchange significantly affects effectiveness by introducing communication delays

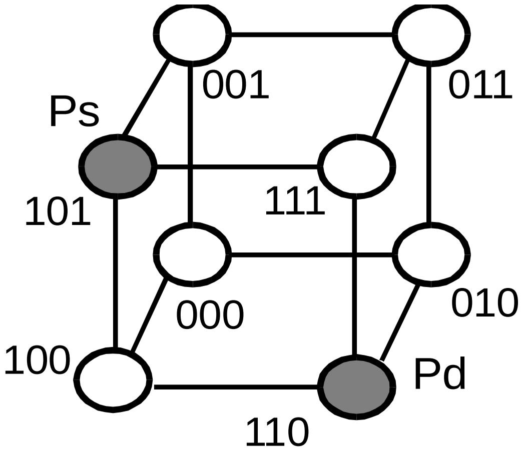
Proper implementation of communication operation on various architectures is a key issue to effective execution of parallel algorithms

Routing mechanism determines the path a message takes through the network from the source to the destination

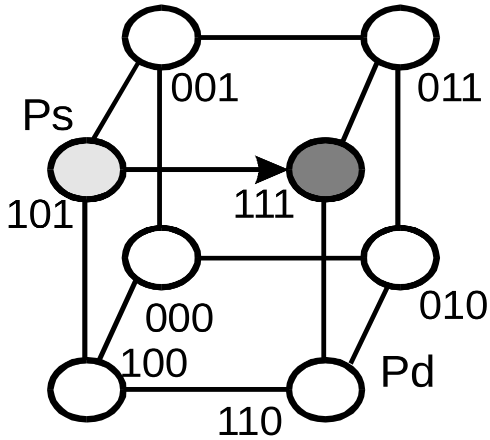
Input is the source and the destination address and also maybe network state

It returns one or *more* paths through the network from the source to the destination.

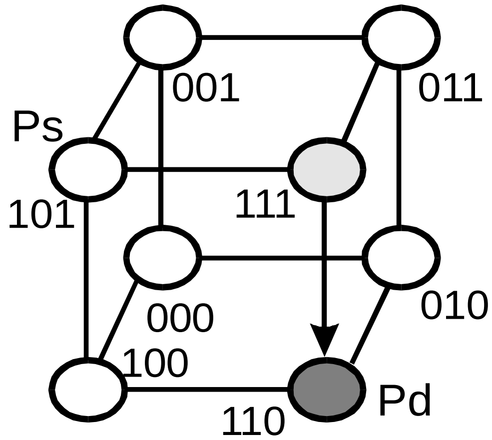
Routing a message in a hypercube



Routing a message from processor Ps (101) to processor Pd (110) in a three dimensional hypercube.



step 1: 101->111



step 2: 111->110

Communication Operations
2 Routing Mechanism

Classification:

On the path length:

minimal

nonminimal

On the use of state of the network:

deterministic (network state not used)

adaptive (detect congestion in network)

A commonly used deterministic and minimal routing technique *in static networks* is the **dimension-ordered routing**

for two-dimensional mesh: **XY-routing**

for hypercube: **E-cube routing.**

XY-routing: message sent first along the *x* then along the *y dimension*

Path length: $|S_x - D_x| + |S_y - D_y|$

E-cube routing: min. distance is the number of bit-differences in addresses: $P_s \oplus P_d$,, the path is along non-zero bits of this exp.

Communication Operations
3 Communication Cost in Static Networks

Communication Latency

the time taken to communicate
its basic parameters:

Startup time *(t_s):* the time to handle the message at the source processor (prepare, run routing algorithm, establish interface)

Per-hop time or node latency *(t_h):* the time taken to travel between two directly connected processors, depends on defining where the message should be forwarded; appears at each intermediate processor on the route

Per-word transfer time *(t_w):* time for each word to traverse the link. It depends the channel bandwidth *r (in words)*

$$t_w = 1/r.$$

Communication Operations
4 Routing and Switching Techniques

Store-and-Forward Routing

the message traverses the path with multiple links, each intermediate processor receives, stores the entire message and after forwards it

Suppose a message of size m traverses l links.

The communication time:

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

in most cases t_h is small, hence:

$t_{comm} = t_s + m\, t_w\ l.$

Disadvantage: poor usage of communication resources

Communication Operations
4 Routing, Switching Techniques
(continued)

Cut-Through Routing

The message is broken into smaller parts (packages), the intermittent processor receives a part (package) stores it and forwards it before receiving the entire message.

Result: increased utilization of the communication resources, less memory required for storage.

Increased probability for **deadlocks**.

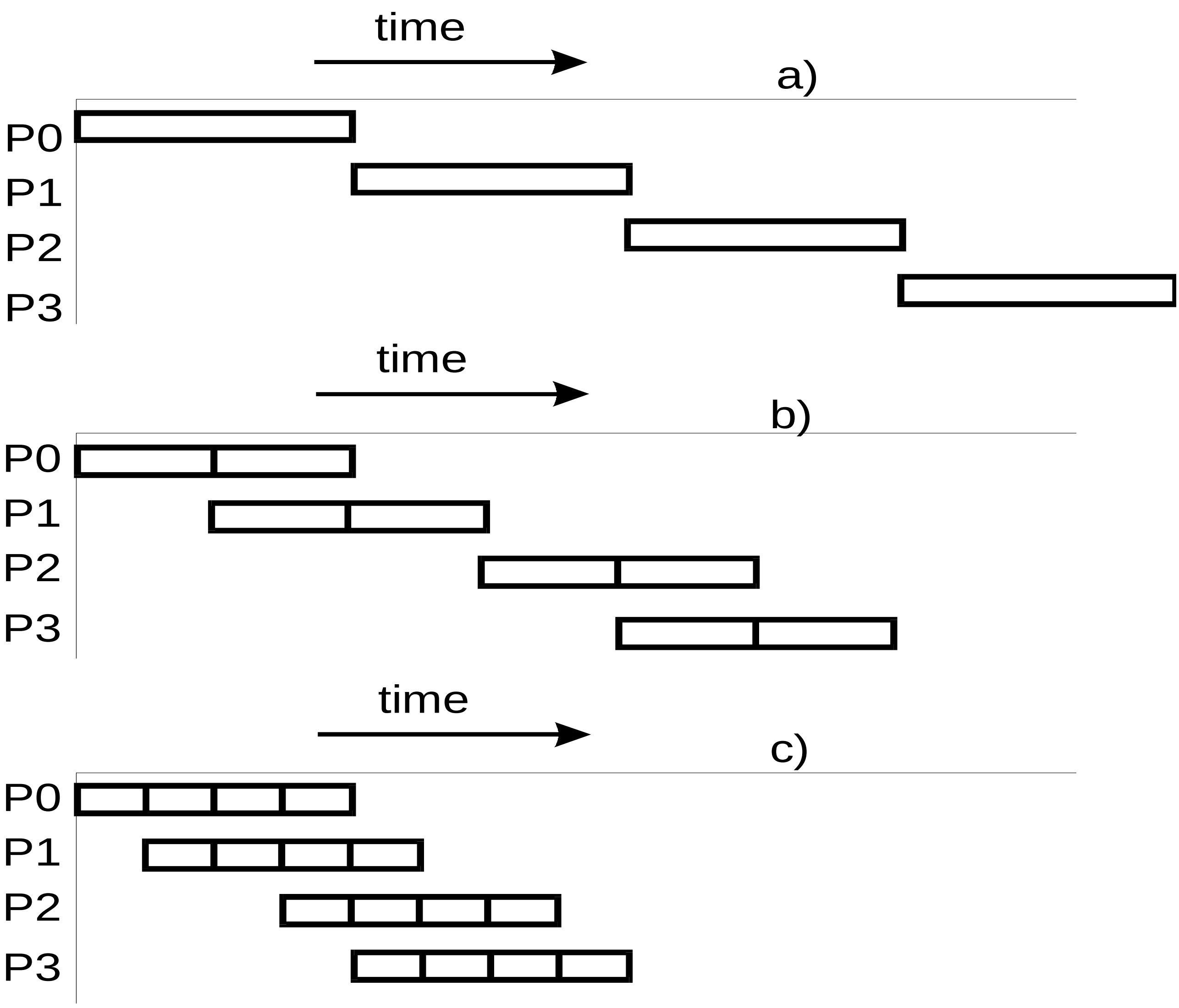
Wormhole routing is Cut-Through Routing with small units called *flow-control digits (flits).*

Pipelined communication.

Suppose a message of size *m* traverses *l* links.
The communication time is:

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

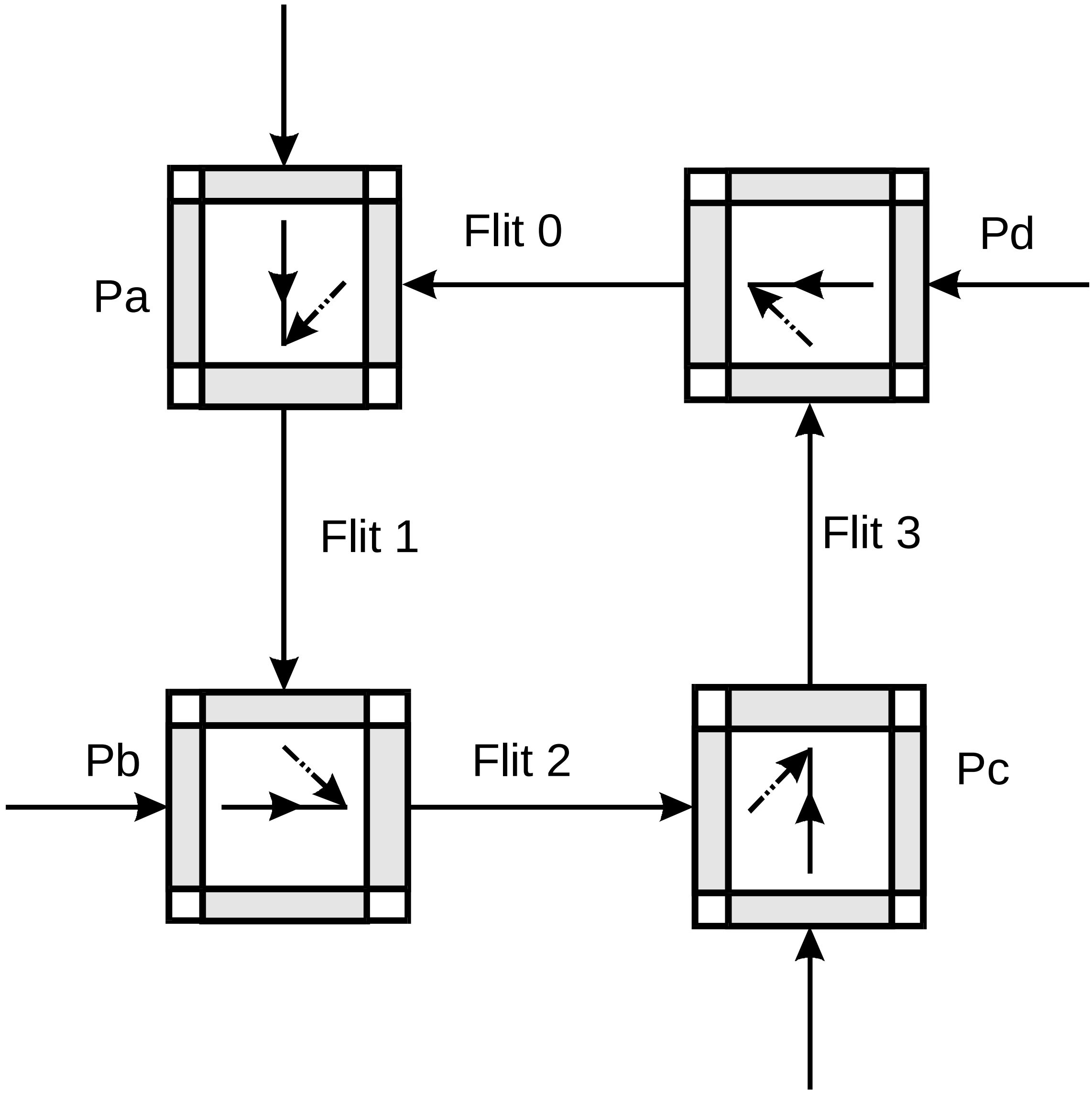
From **Store-and-Forward Routing** To **Cut-Through Routing**



Sending a message from processor P0 to P3
a.) through Store-and-Forward technique, then extending the concept to Cut-Through routing
b.) the message is broken into two parts
c.) the message is broken into four parts

Deadlock in Wormhole routing

Deadlock example in wormhole
routing network



Shaded rectangles are flit buffers
The dotted line shows the desired
direction of message travel

Communication Operations
5 Basic Communication Operations:

The operations differ from each other in:
how many senders and receivers are foreseen, the destination is addressed or not, simple sending or shifting is required.

Classification of the basic types of Communication operations:

Simple message transfer between two processors
One-to-All Broadcast
All-to-All Broadcast
One-to-All Personalized Communication
All-to-All Personalized Communication
Circular Shift

Communication Operations
6 Simple message transfer between two processors

Most basic operation.

Store-and-Forward (SF)

$t_{comm} = t_s + m\, t_w\, l$

The maximal communication lengths and SF times are summed up as follows:

Network type	*lmax*	SF Time max
Ring	$p/2$	$t_{comm}=t_s+mt_w p/2$
2-D torus	$\sqrt{p}$	$t_{comm}=t_s+mt_w \sqrt{p}$
Hypercube	$\log p$	$t_{comm}=t_s+mt_w \log p$

In the case of small sized messages the transfer time gets close (equal) to the one in Cut-Through routing

Communication Operations
6 Simple message transfer between two processors
(continued)

Most basic operation.

Cut-Through Routing (CT)

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

The maximal communication lengths and CT times are summed up as follows:

NW type	Max. l	CT Time
Ring	$p/2$	$t_{comm}=t_s+mt_w+t_h\, p/2$
2-D torus	$\sqrt{p}$	$t_{comm}=t_s+mt_w+t_h\, \sqrt{p}$
Hypercube	$\log p$	$t_{comm}=t_s+mt_w+t_h\, \log p$

If the message is large, the distance among the processors becomes unimportant for the communication time. The time becomes approximately the same as in the case of directly connected processors.

Communication Operations

7 *Dual operations:* One-to-All broadcast and Single-Node Accumulation

Definition

1. In *One-to-All Broadcast* or *Single-Node Broadcast* operation a single processor sends identical data to all others or a subset of them.
2. The operation when a single node accumulates information from every processor is called *Single-Node Accumulation.* every processor has a message of size *m*. Data are combined through associative operation in the destination processor, the message size remains unchanged.

Application: matrix-vector multiplication, Gaussian elimination, vector inner product, shortest path, etc. algorithms.

Communication Operations

7 One-to-All broadcast with Store and Forward routing

(continued)

Each processor sends the message to its neighbor, that stores it for itself and forwards to others.

The process is shown for various topologies on figures. The communication time is as follows:

Network type	Communication time
Ring	$T_{onetoall}=(t_S+mt_W)\ p/2$
2-D torus	$T_{onetoall}=(t_S+mt_W)\ \sqrt{p}$
Hypercube	$T_{onetoall}=(t_S+mt_W\ )\log p$

We obtained the shortest time for the hypercube structure, because it has the highest connectivity.

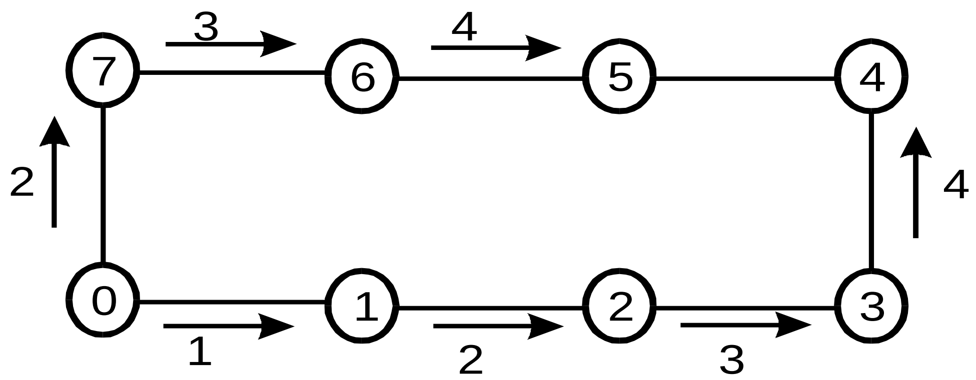
Communication Operations
7 One-to-All broadcast with Cut-Through routing
(continued)

Each processor sends the message to its neighbor, that stores it for itself and forwards to others.

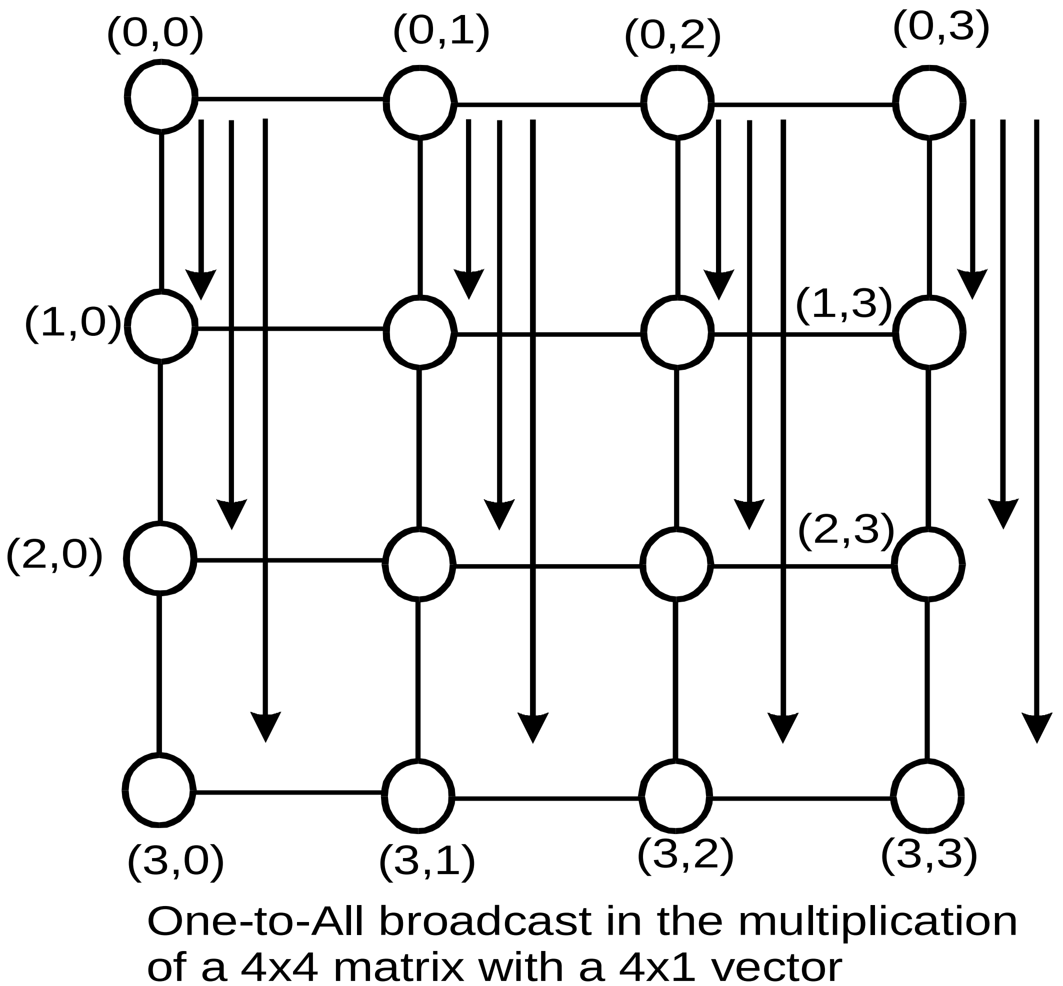
The process is shown for various topologies on figures. The communication time is as follows:

Network type	Communication time
Ring	$T_{onetoall}= logp(t_s + mt_{w)} + t_h\ (p-1)$
2-D torus	$T_{onetoall}= (t_s + mt_w) logp + 2t_{h(}\sqrt{p}-1)$
Hypercube	same as at SF due of its local connectivity
Balanced tree	$T_{onetoall}= ((t_s + mt_w t_h\ (logp+1))logp$

7 One-to-All Broadcast



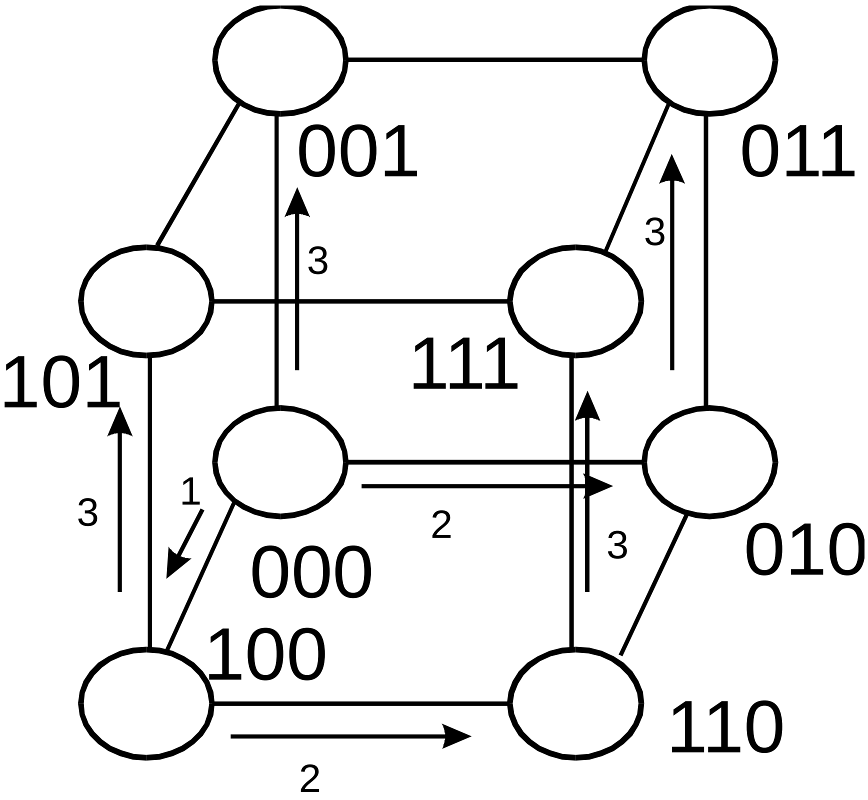
One-to-All broadcast with SF routing on an 8-processor ring



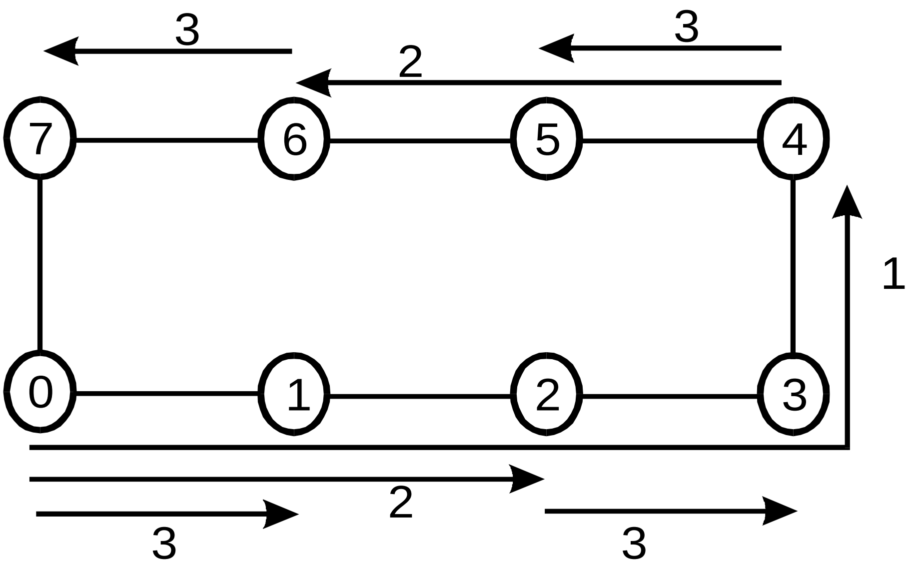
One-to-All broadcast in the multiplication of a 4x4 matrix with a 4x1 vector

7 One-to-All Broadcast In a 3-D Hypercube

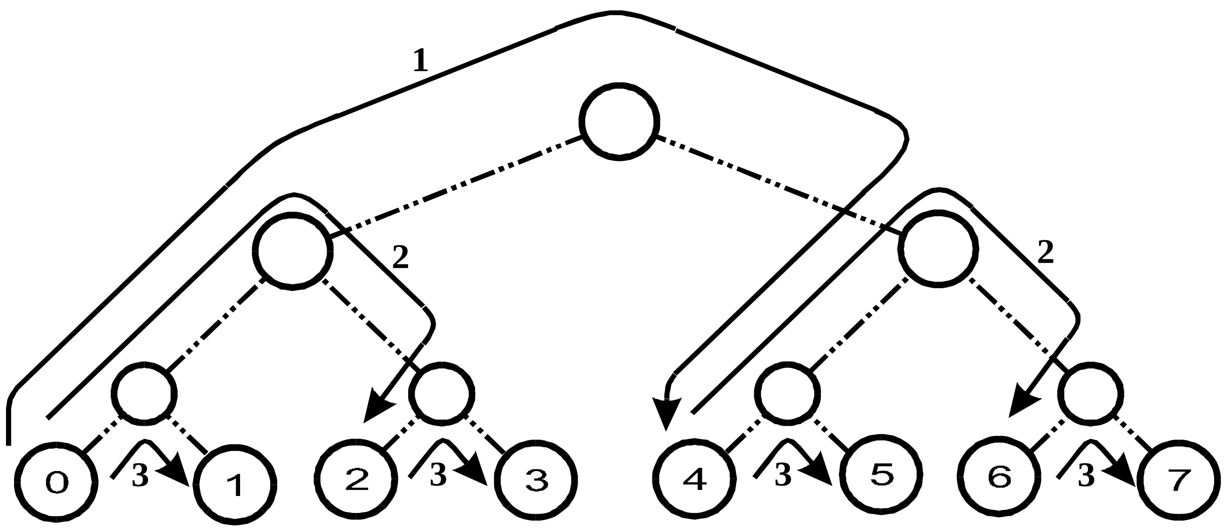
One-to-All broadcast on a three-dimensional hypercube



7 One-to-All Broadcast
In a ring and a tree



One-to-All broadcast with CT routing on an 8-processor ring



One-to-All broadcast on an 8-processor tree

Communication Operations
8 All-to-All broadcast

Generalization of One-to-All type, all *p* processors simultaneously initialize broadcast (**Multinode Broadcast)**. A processor sends the same message to all, but processors may send different messages.

A way to do it is starting *p* one-to-all broadcasts

Multinode accumulation is its dual operation.

Applications: matrix operations, like matrix-matrix and matrix-vector multiplication.

The process is shown for various topologies on figures.

Communication Operations
8 All-to-All broadcast
(continued)

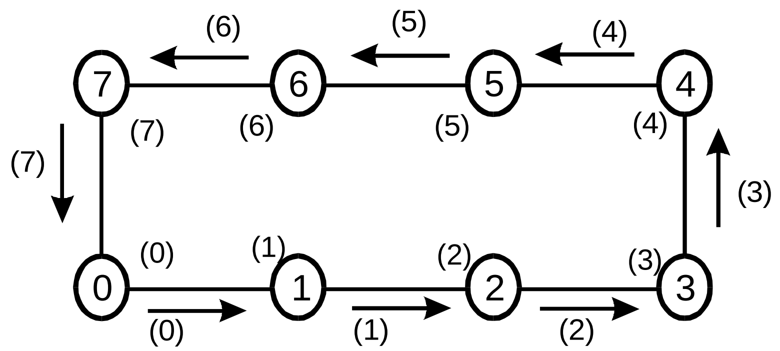
<u>Store-forward routing</u>

All processors send and gather messages first in one dimension, then in the other and so on. In a 2-D mesh, broadcasting is implemented first in a row, all processors gather all messages from their row. Then they sent data column-wise. Hypercube works similarly in *d* dimensions.

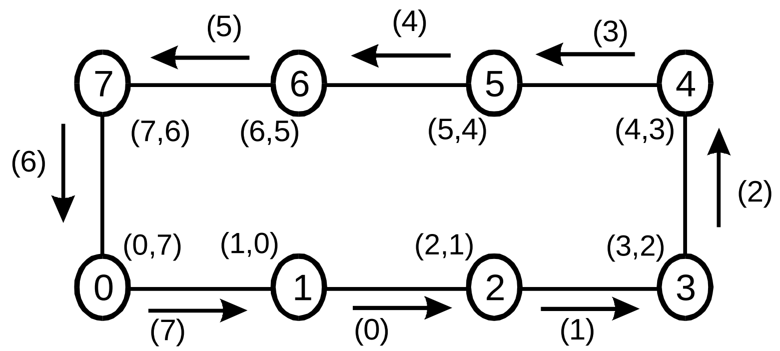
The communication time is as follows:

Network type	Communication time
Ring	$T_{alltoall} = (t_S + mt_W)(p\text{-}1)$
2-D torus	$T_{alltoall} = mt_W(p\text{-}1) + 2t_S(\sqrt{p}-1)$
Hypercube	$T_{alltoall} = \log p\ t_S + mt_W(p\text{-}1)$

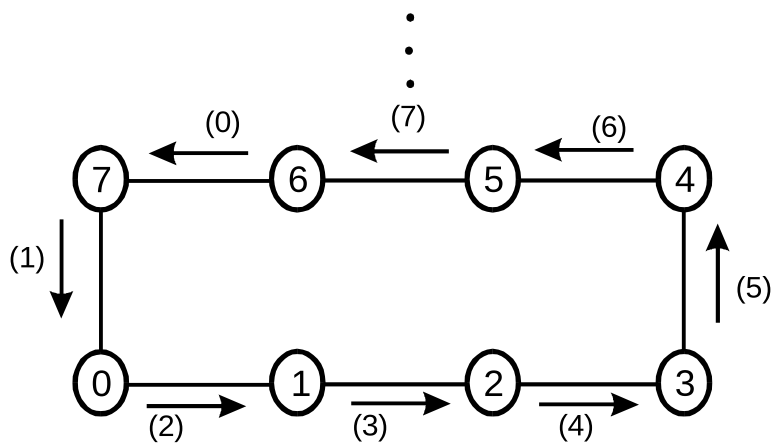
8 All-to-All Broadcast in a ring



First communication step



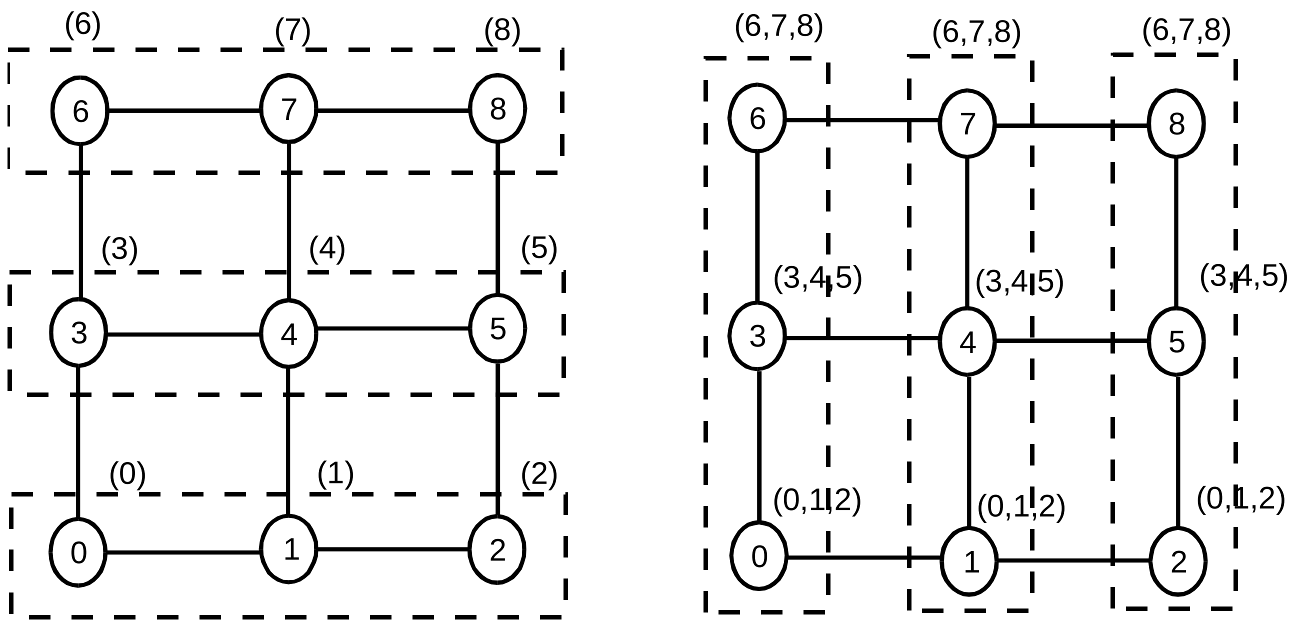
Second communication step



Seventh communication step,
(the last message is coming in)

All-to-All broadcast on on 8-processor ring

8 All-to-All Broadcast in a 2-D mesh

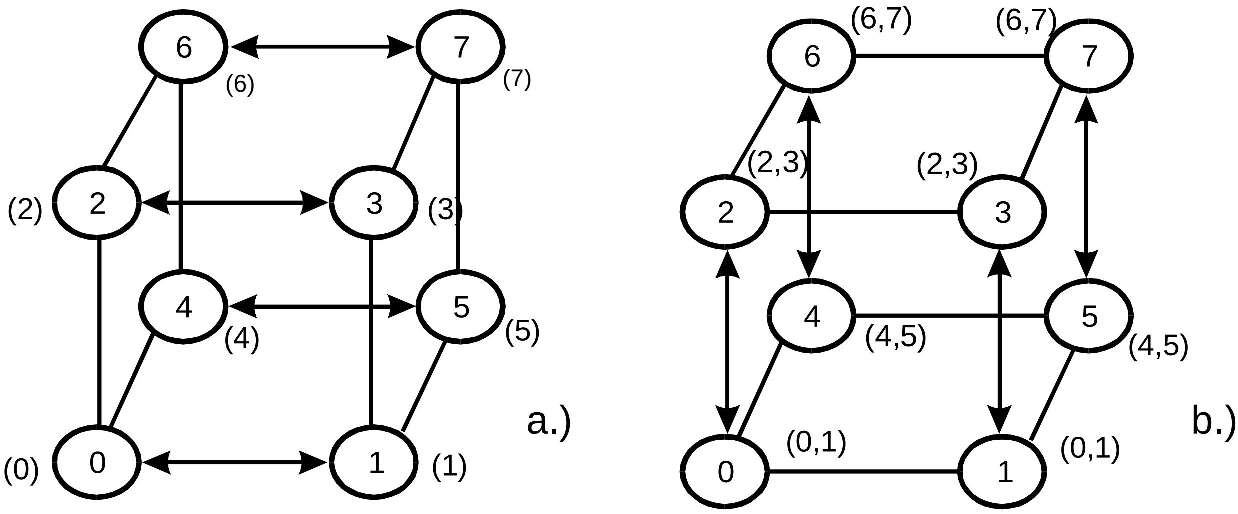


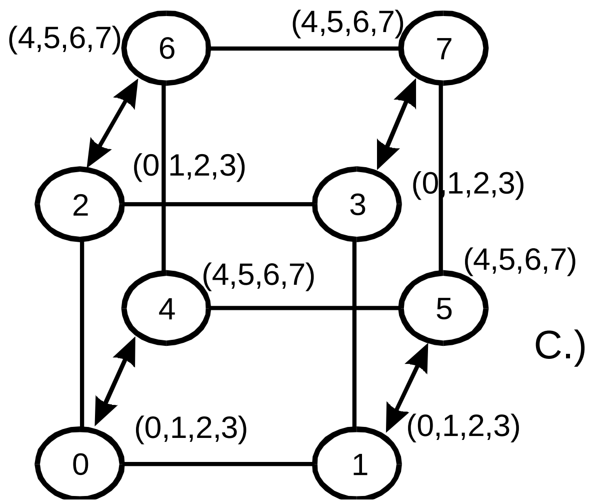
Initial data distribution

Data distribution after rowwise broadcast

All-to-all bradcast on a mesh. Dotted boundaries show processors communicating with each other. By the end of the second phase all processors get (0,1,2,3,4,5,6,7), i.e. all messages

8 All-to-All Broadcast in a Hypercube





All-to-all bradcast in a hypercube

Arrows show processors exchanging messages

Distribution is shown: a.) initial; b.) after the first; c.) after the second step. After the third step all data are at each processor.

Communication Operations
9 One-to-All personalized Communication or Single-Node Scatter

A single processor sends a unique message of size *m* to every other processor. The source processor starts with *p-1* unique messages, one destined for each processor.
The source processor issues *p-1* messages of size *m*.

Its dual operation: **Single-node gather.**

The communication complexity is similar to the complexity of *All-to-All broadcast.*

Communication Operations
10 All-to-All personalized Communication or Total Exchange

Each processor sends a distinct message of size *m* to every other processor. Messages are different to different processors.

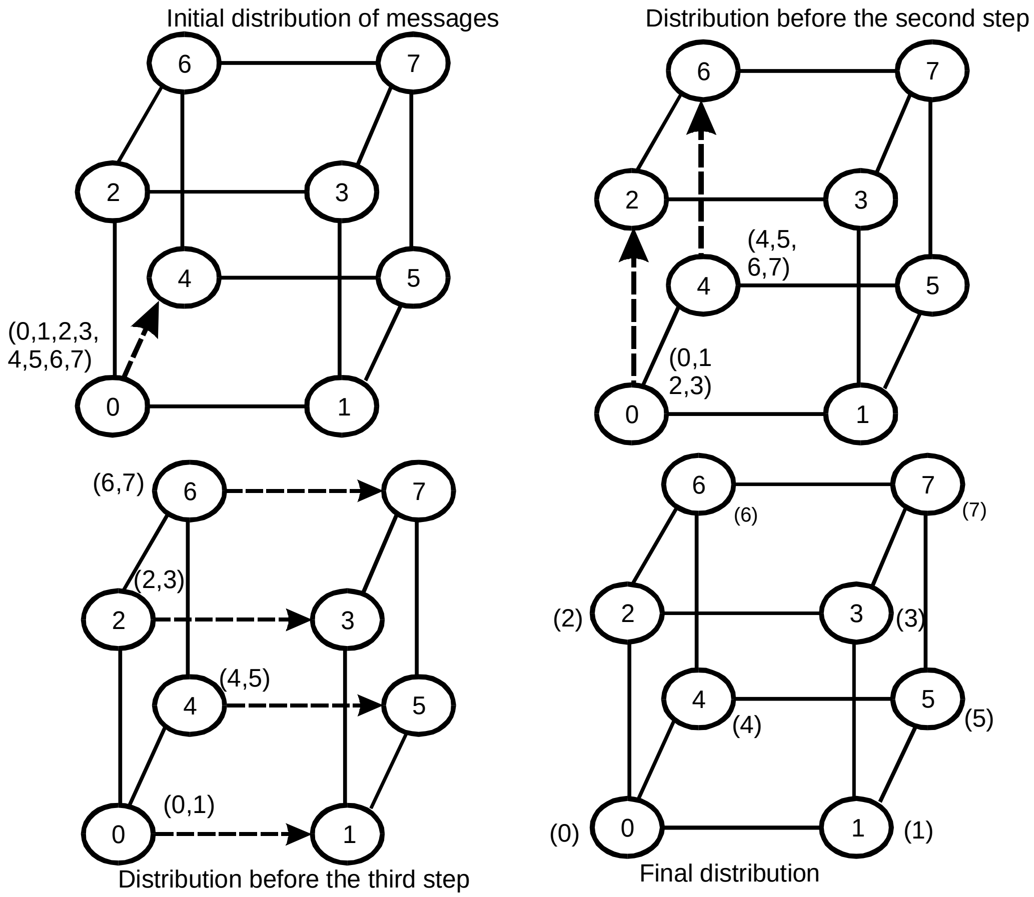
Application: in Fast Fourier Transformation, matrix transpose, parallel database.

The communication patterns are identical to those of *All-to-All broadcast*, the size and the content of the messages are different.

The communication time is as follows:

Network type	Communication time
Ring	$T_{alltoallper} = (t_s + 1/2p\ m\ t_w)\ (p-1)$
2-D torus	$T_{alltoallper} = (mt_w p + 2t_s)(\sqrt{p} - 1)$
Hypercube	$T_{alltoallper} = (t_s + 1/2pmt_w) \log p$

One-to-All Personalized Communication in a Hypercube



One-to-All personalized communication on a hypercube

Communication Operations
11 Circular Shift

Circular shift is a special case of *permutation* operations. In a **circular q-shift** processor i sends data to processor $(i+q) \bmod p$
p is the number of processors.

Application: in matrix operations, string and image pattern matching operations.

The implementation may be fairly intuitive, very much depends on the architecture and the q parameter of the shift operation.