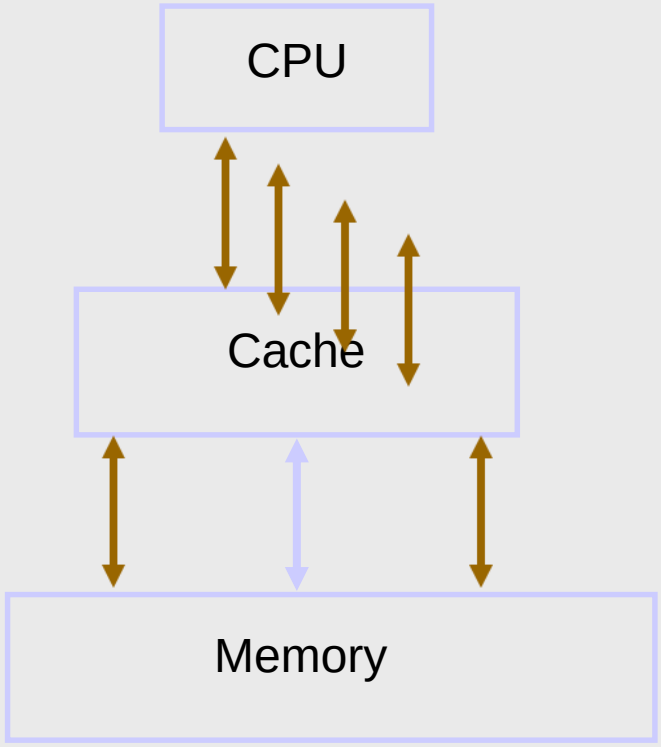
# “Distributed Memory Parallel Architecture”

# Solution: Memory Hierarchy and Concurrency

Multi-core
Multi-threading
Multi-issue

Out-of-order Execution
Speculative Execution
Runahead Execution




Multi-banked Cache
Multi-level Cache

Pipelined Cache
Non-blocking Cache
Data Prefetching
Write buffer

Multi-channel
Multi-rank
Multi-bank

Pipeline
Non-blocking
Prefetching
Write buffer

**Input-Output (I/O)**

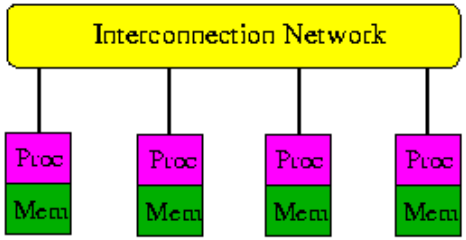
*Parallel File System*

# Parallel Architectures

IBM RS/6000 SP

SGI Power Challenge XL

## Distributed Memory Machines



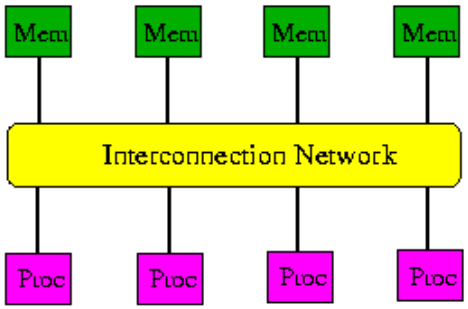

**Advantages:**

+ scalable
+ latency hiding

**Disadvantages:**

- harder to program
- program must be replicated

## Shared Memory Machines



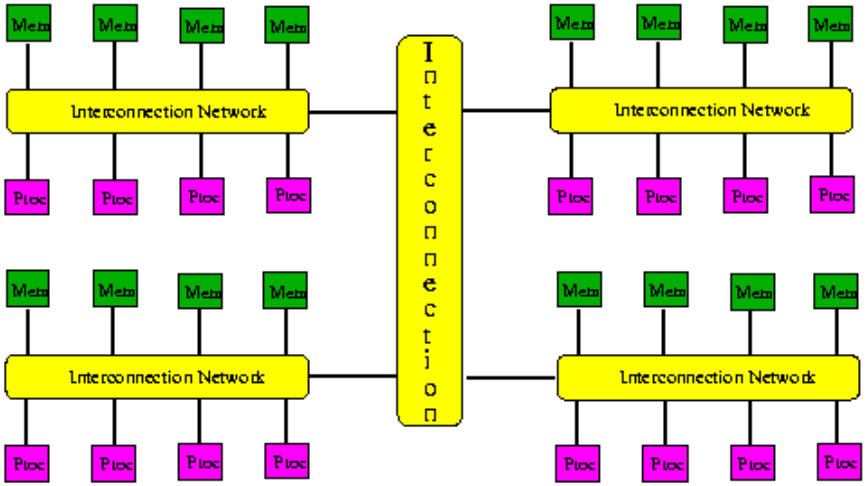

**Advantages:**

+ ease of programming
+ processors share code and data

**Disadvantages:**

- scalability problem

## Cluster of Symmetric Multiprocessor Systems (SMP)



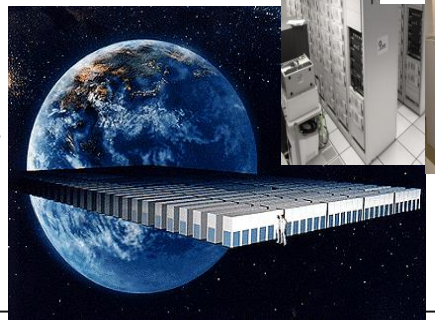

**Advantages:**

+ scalable

**NEC SX-5 multi node**
**(8 CPUs pro Knoten)**

**Disadvantages:**

- programming paradigm unclear



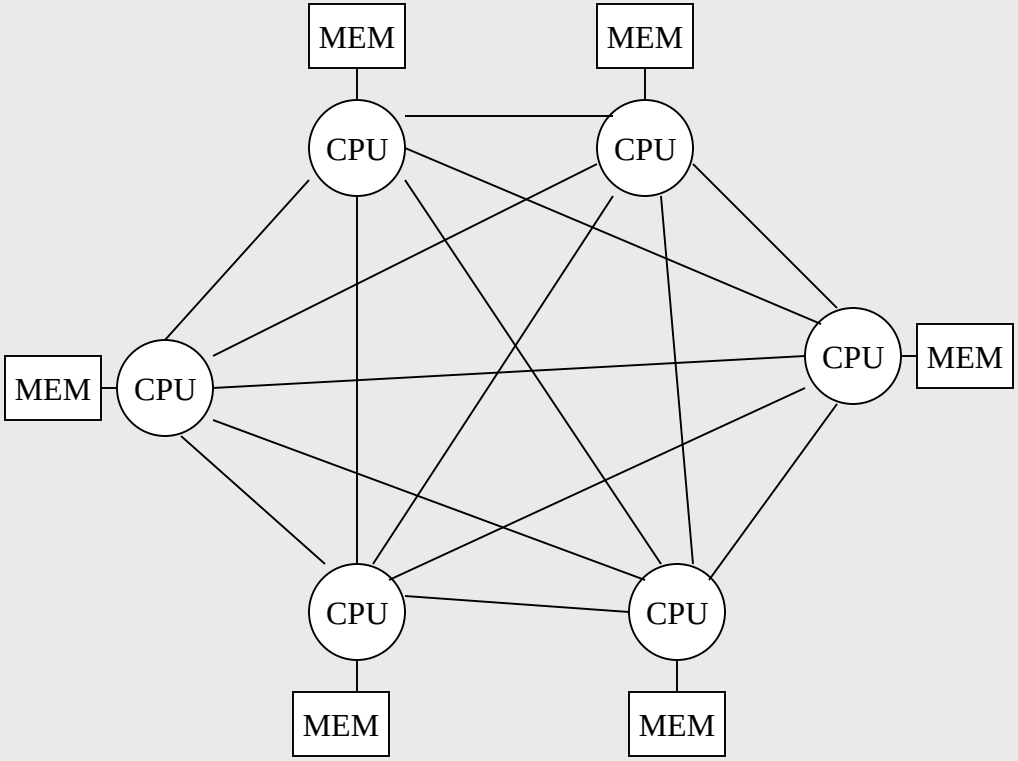
**Earth Simulator**
**(540*8 CPUs)**

# Outline

- Overview of distributed memory parallel architectures
- Interconnection networks
- Message passing issues

# Distributed Address Space Machines

- Each processor has its own local address space
- Processors share data via explicit message passing



# Distributed Address Space Machines

- Important aspects are:
  - Interconnect
  - Message routing mechanism
- These aspects determine:
  - Performance: programming techniques
  - Scalability
  - cost

# Interconnects: Design Choices

- Operational characteristics:
  - Topology: dynamic or static
  - Timing protocol: synch or asynch
  - Switching method:circuit or packet
  - Control strategy: centralized or distributed
- Performance criteria:
  - Functionality: types of support for various services
  - Network latency: worst case time for unit message
  - Bandwidth: maximum data transfer rate
  - Hardware complexity: cost of implementation
  - Scalability: ease of expansion

# Interconnects: Evaluation

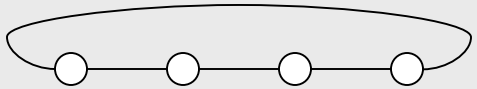
- Diameter: maximum distance between any two processors. The distance is the least number of links (hops) that need to be transversed between processors

- Connectivity: the multiplicity of paths between processors. Arc connectivity is the minimum number of links that need to be removed in order to bread the network into two disjoint parts; it is a measure of connectivity. High connectivity is desirable for avoiding contention

- Bisection width: the minimum number of links that need to be removed in a network to separate the processor into two halves. Bisection width is a measure of the volume of the traffic that can be handled by the network

# Interconnects: Ring

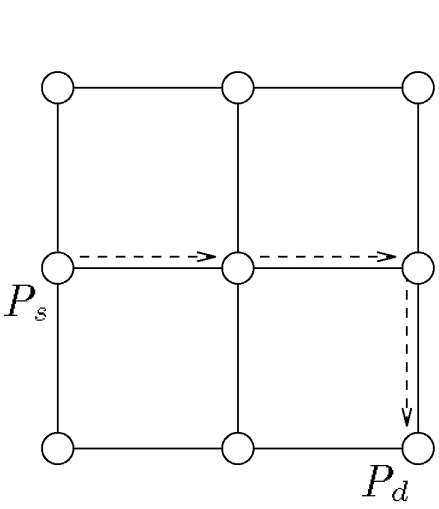
- Each processor connected to 2 other processors
- Diameter of a ring is $\left\lfloor \frac{p}{2} \right\rfloor$
- Arc connectivity of a ring is 2
- Bisection width of a ring is 2



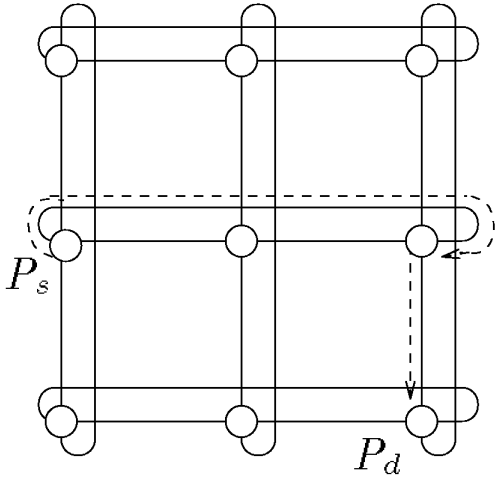
# Interconnects: Multidimensional Meshes

- Each processor in an *d* dimensional mesh is connected to *2d* other processors except for the corner processors
- In practice, only 2 or 3 dimensional meshes are constructed
- Mesh with wrap around: torus

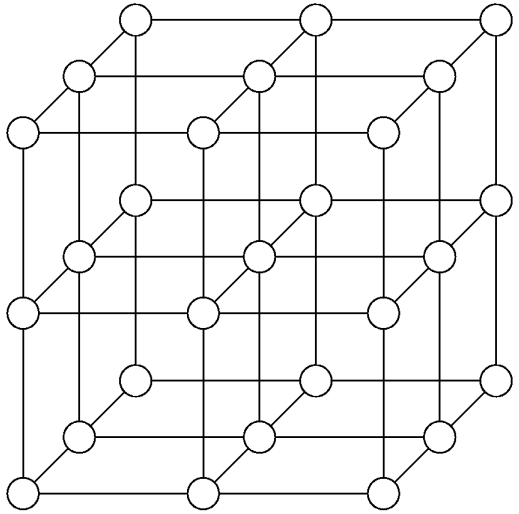
# Interconnects: Multidimensional Meshes




(a)




(b)



(c)

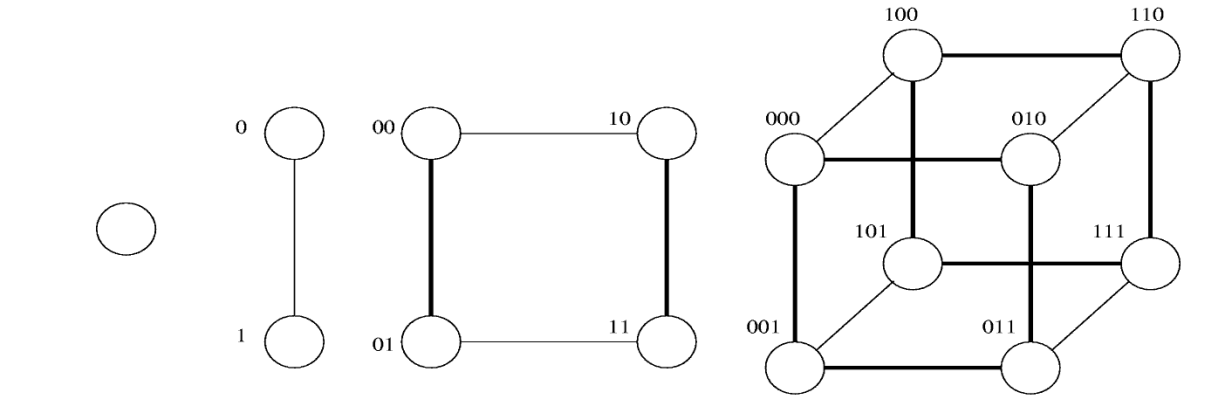
# Interconnects: Multidimensional Meshes

- Diameter of a 2D mesh is $2(\sqrt{p}-1)$; for torus it is $2\left\lfloor \frac{\sqrt{p}}{2} \right\rfloor$
- Arc connectivity of a 2D mesh is 2; for torus it is 4
- Bisection width of a 2D mesh is $\sqrt{p}$ ; for torus it is $2\sqrt{p}$

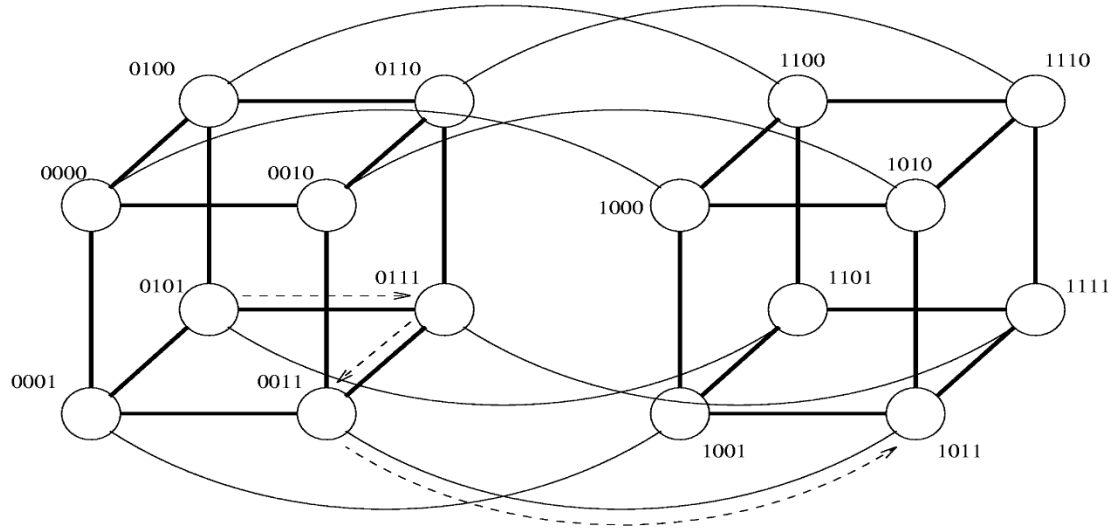
# Interconnects: Hypercube

- A hypercube is a multidimensional mesh with exactly two processors in each dimension
- In a d-dimensional hypercube, each processor is connected with *d* other processors

# Interconnects: Hypercube




0-D hypercube 1-D hypercube 2-D hypercube 3-D hypercube




4-D hypercube

# Interconnects: Hypercube

- Hypercubes can be constructed recursively; when two d-dimensional hypercubes are used to construct a d+1-dimensional hypercube, the labels on the nodes of the original hypercubes are prefixed with a 0 or 1 for the new hypercube
- Two processors are connected if their labels differ in exactly one bit position
- Consider two processors with labels s and t, the Hamming distance for the two processors is defined to be the number of positions in which their labels differ
- For example, if s=001 and t=010, Hamming distance is 2
- The Hamming distance represents the shortest path between two processors

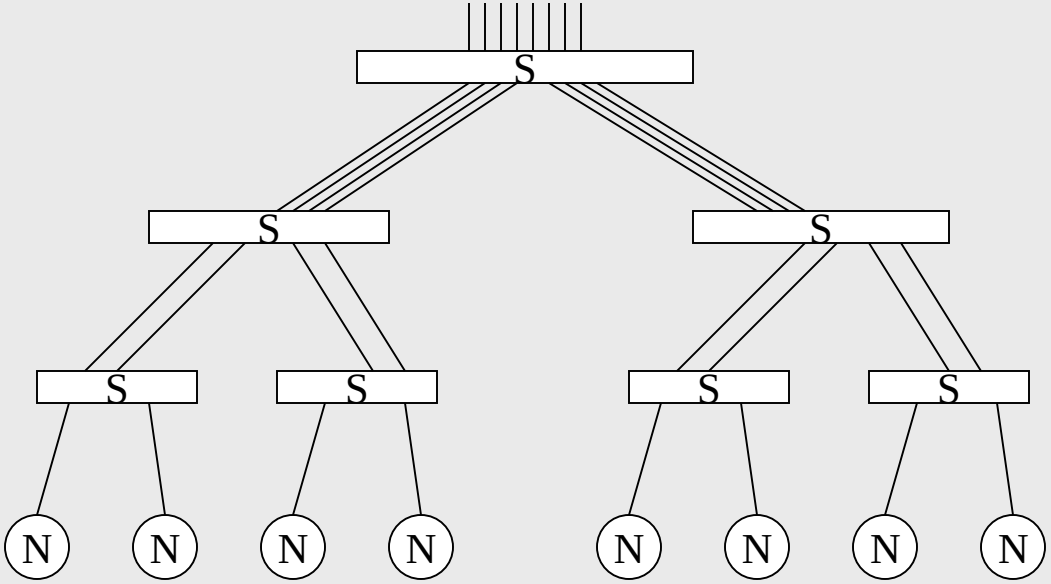
# Interconnects: Hypercube

- Diameter of a hypercube is log(p)
- Arc connectivity of a hypercube is log(p)
- Bisection width of a hypercube is p/2

# Interconnects: Fat Tree

- A tree network is one where there is exactly one path between a pair of processors
- In a simple tree, the bandwidth on higher level links is shared among all processors below creating bottlenecks
- A fat tree solves the problem by using wider links at higher levels in the tree

# Interconnects: Fat Tree



# Interconnects: Tree

- Diameter of a tree is 2log((p+1)/2)
- Arc connectivity is 1
- Bisection width is 1

# Interconnects: Comparison

| Network | Diameter | Bisection Width | Arc Connectivity |
|---|---|---|---|
| Completely connected | $1$ | $\frac{p^2}{4}$ | $p-1$ |
| Ring | $\left\lfloor \frac{p}{2} \right\rfloor$ | $2$ | $2$ |
| Mesh | $2(\sqrt{p}-1)$ | $\sqrt{p}$ | $2$ |
| Torus | $2\left\lfloor \frac{p}{2} \right\rfloor$ | $2\sqrt{p}$ | $4$ |
| Binary tree | $2\log(\frac{p+1}{2})$ | $1$ | $1$ |
| Hypercube | $\log(p)$ | $\frac{p}{2}$ | $\log(p)$ |

# Effects of Interconnects on Programming

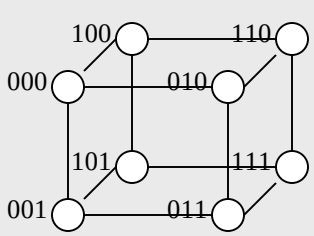
- Congestion
  - Traffic patterns on the network can create hot spots
- Latency hiding
  - Do some other work while a data transfer is in progress
- Latency amortization
  - Try to fetch large amounts of data at the same time

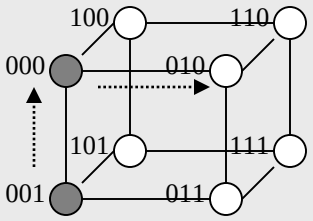
# Message Routing Mechanisms

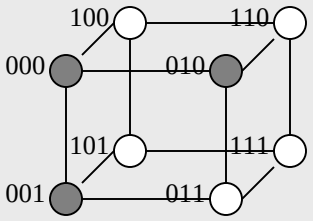
- Routing mechanism determines the path a message takes through the network when going from a source to a destination processor
- Minimal or non-minimal routing
  - Minimal always takes the shortest path
  - Non-minimal may sometimes take a longer path to avoid congestion
- Deterministic or adaptive
  - Deterministic routine always takes the same path between processors
  - Adaptive routing can take different paths depending on congestion

# Example:
# E-cube Routing in Hypercubes

- If the message is at processor $P_i$ and needs to go to $P_d$, compute $s = P_i \oplus P_d$ Send message along dimension k from $P_i$ where k is the least significant non-zero bit in s
- Continues this process at each successive processor until $P_d$ is reached



# Example: XY Routing in 2D Mesh

- Message sent along X dimension until destination's X coordinate is reached
- Then message is sent along Y dimension until it reaches destination processor
- Called dimension-order routing

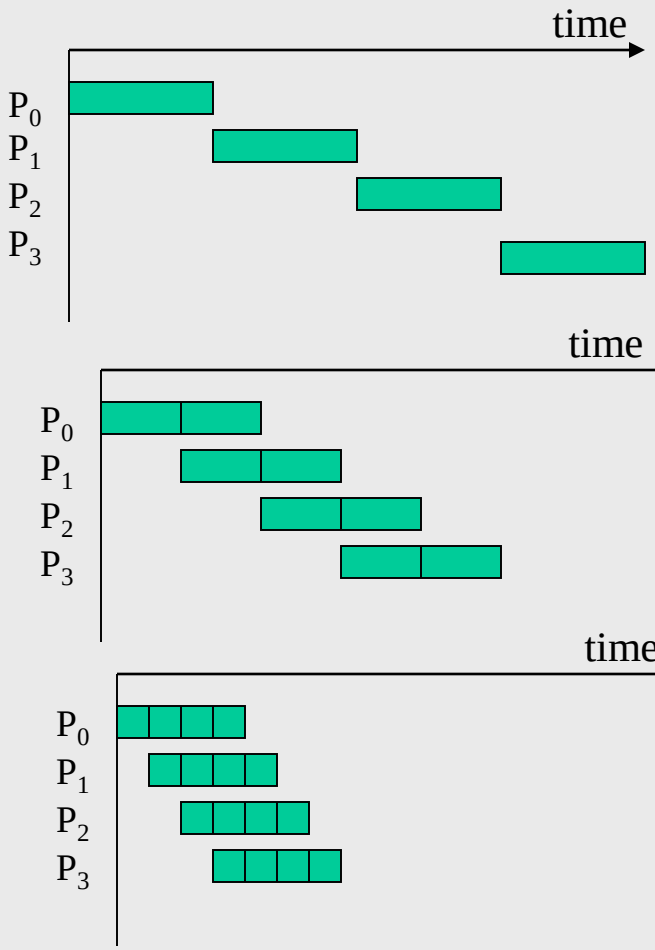
# Message Transfer Mechanisms

- Message typically consist of:
  - A header which contains information about the destination
  - The data that needs to be transmitted
  - A trailer which signals the end of a message
- Message transfer mechanism determines how message data is actually transferred across network links in the chosen message route
- Three components to message transfer cost:
  - Startup time($t_s$): cost of handling message at sending processor
  - Per-hop time($t_h$): time taken by the header to traverse a link
  - Per-work transfer time ($t_w$): time taken for a word to traverse a link

# Message Transfer Mechanisms

- Store-and-forward routing:
  - the message is transferred completely from link to link along the route between processors
  - it is buffered at intermediate processors.
- Cut-through routing:
  - the message is chopped into small units called flow-control digits or flits.
  - flit is transmitted in a pipelined fashion between processors along the message route

# Message Transfer Mechanisms




(1) A single message sent over A store-and-forward network

(2) The same message broken into two parts and send over the network

(3) The same message broken into four parts and sent over the network

# Message Transfer Mechanisms

- Cost of message transfer for a message length of m words along a route of *l* links:
  - Store-and-forward routing:

$$t_{comm} = t_s + (mt_w + t_h)l \approx (t_s + mt_w)l$$

  - Cut-through routing:

$$t_{comm} = t_s + mt_w + t_h l$$

  - Point-to-point and Wormhole routing communication:

$$t_{comm} = t_s + mt_w$$

# IBM SP2

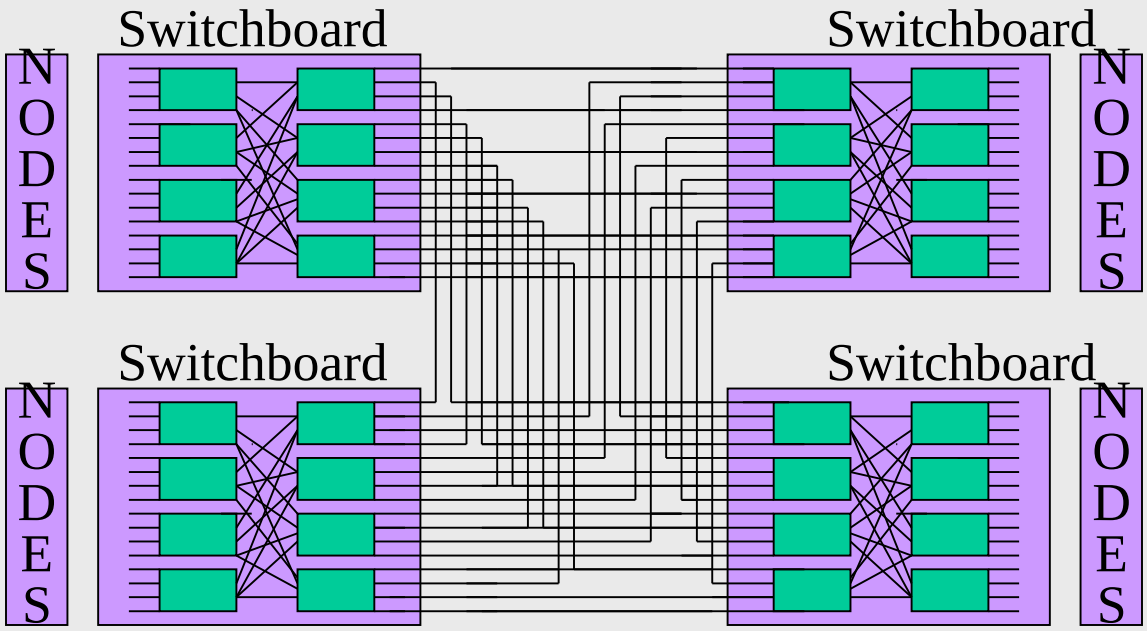
- IBM SP2 is a general-purpose scalable parallel system based on distributed memory message passing architecture
- SP2 systems range from 2 to 59 nodes (or processing elements)
- Each node is POWER2 technology RISC/6000 processor
- Interconnected by high-performance, multistage, packet-switched network for interprocessor communication
- Each node contains copy of standard AIX OS
- Supports both message passing and data parallel programming models

# Organization of the SP2 Nodes

- Consists of wide nodes and thin nodes
- Both processors have two fixed point units, two floating-point units(each capable of a multiply and add each cycle), and an instruction and branch control unit
- Peak processing 267 MFLOPS
- Wide node memory is 2GBytes, and bandwidth of 2.1GB/s
- Thin nodes have less memory and less bandwidth

# Organization of the SP2 Nodes

- High Performance Switch of 64 node SP2
- Each switch is a 4*4 bidirection cross-bar switch
- Multiple paths between any two nodes
- Network scales with added nodes



# BG/P Overview

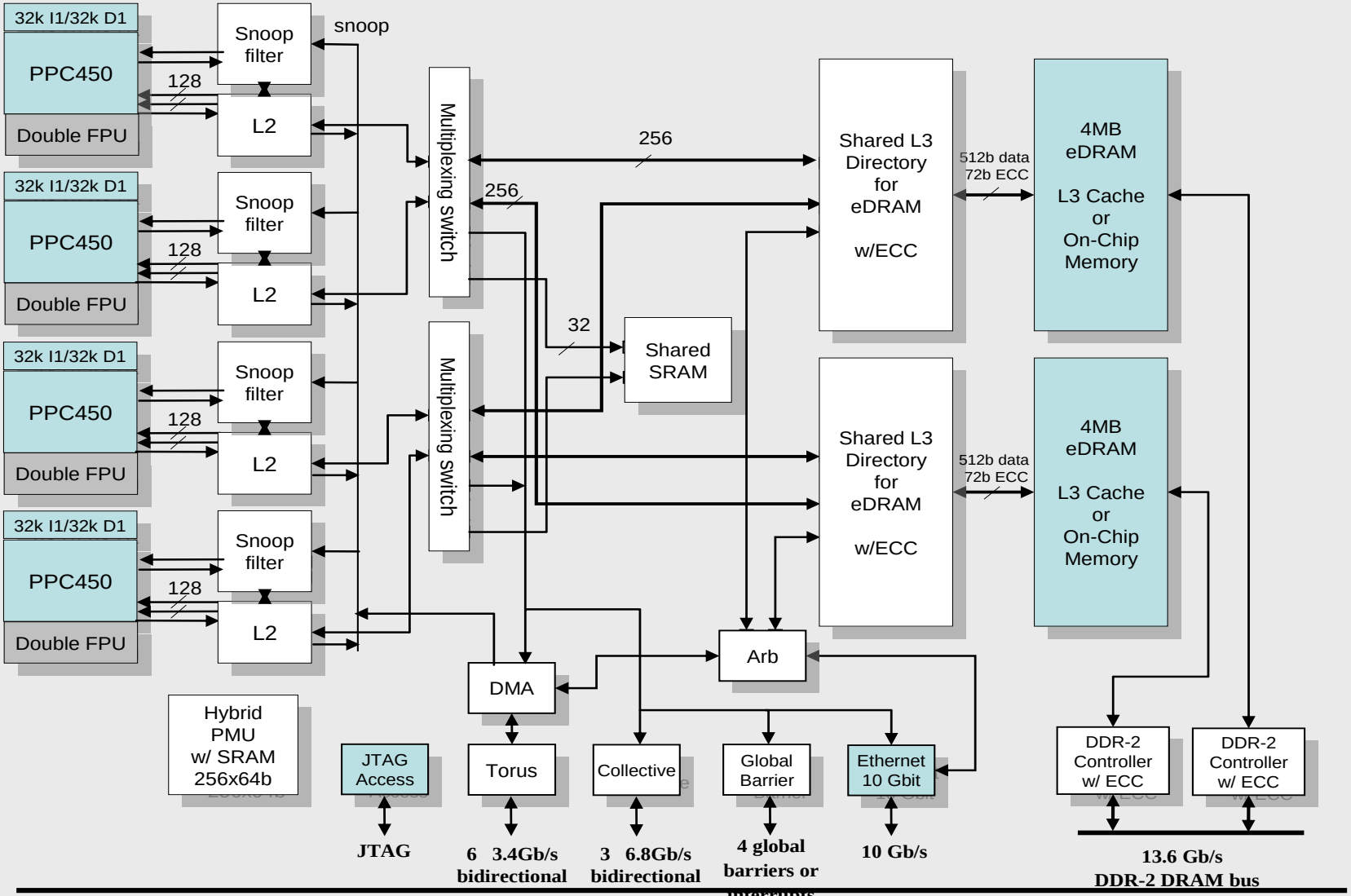
# Overview

- BlueGene/P Quick Introduction
  - BG/P Hardware
  - Network
  - Software
- MPI Implementation on BlueGene/P

# BG/P Hardware Introduction

- Up to $256^3$ compute processors
  - Largest machine: 40960 nodes at ANL
  - Relatively slow processors (850 MHz)
  - But -- low power, low cooling, very high density
- System-on-a-chip technology (4 cores, 8 FPUs, memory controllers, networks, etc on single ASIC)
- 3 very high-speed application networks
  - Torus network has a DMA engine

# BG/P ASIC


32k I1/32k D1
PPC450
Double FPU
Snoop filter
L2
128
snoop
Multiplexing switch
256
32
Shared SRAM
Shared L3 Directory for eDRAM w/ECC
512b data 72b ECC
4MB eDRAM L3 Cache or On-Chip Memory
Arb
DMA
Hybrid PMU w/ SRAM 256x64b
JTAG Access
Torus
Collective
Global Barrier
Ethernet 10 Gbit
DDR-2 Controller w/ ECC
JTAG
6 3.4Gb/s bidirectional
3 6.8Gb/s bidirectional
4 global barriers or interrupts
10 Gb/s
13.6 Gb/s DDR-2 DRAM bus

# PowerPC 450 Processor

- Offshoot of PPC440 Processor
- 32-bit architecture at 850 MHz
- Single integer unit (fxu)
- Single load/store unit
- Special double floating-point unit (dfpu)
- L1 Data cache : 32 KB total size, 32-Byte line size,
  - 64-way associative, round-robin replacement
  - 4 cores on BG/P are L1 cache coherent
- L2 Data cache : prefetch buffer, holds 16 98-byte lines
- L3 Data cache : 8 MB
- Memory : 2 GB DDR, ~13.6GB/s bandwidth
- Double FPU has 32 primary floating-point registers, 32 secondary floating-point registers, and supports :
  - standard powerpc instructions, which execute on fpu0 (fadd, fmadd, fadds, fdiv, …), and
  - SIMD instructions for 64-bit floating-point numbers (fpadd, fpmadd, fpre, …)
- Floating-point pipeline : 5 cycles

# BG/P Hardware

- Double Hummer FPUs
  - 2 64bit FPUs
  - Not independent though
  - Requires careful alignment considerations
  - Compilers are good now, but hand-tuning critical sections might be necessary/valuable

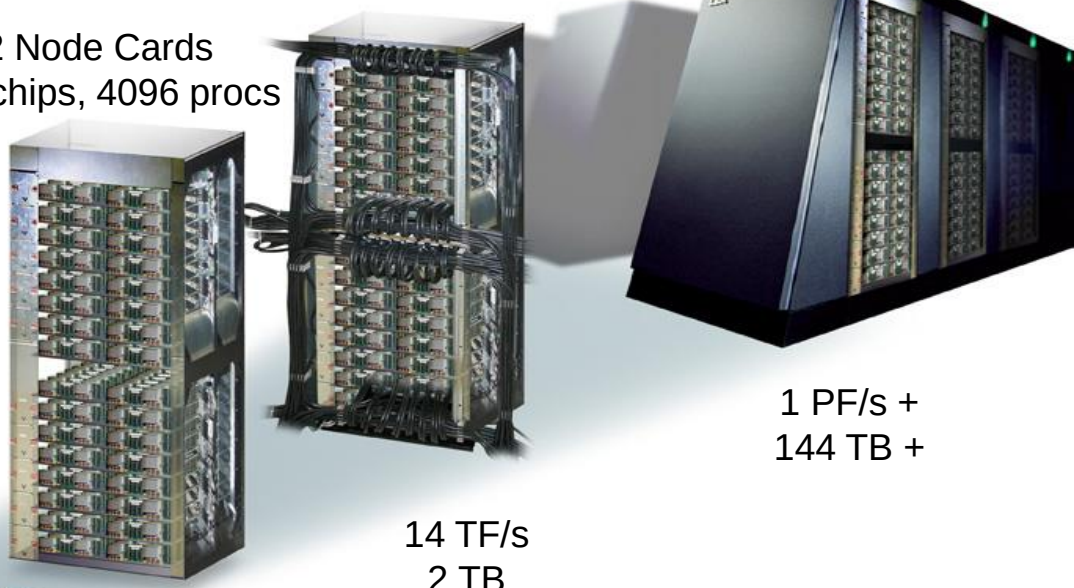
# Mandatory Scaling Slide

## Blue Gene/P

**System**
1 to 72 or more Racks

Cabled 8x8x16

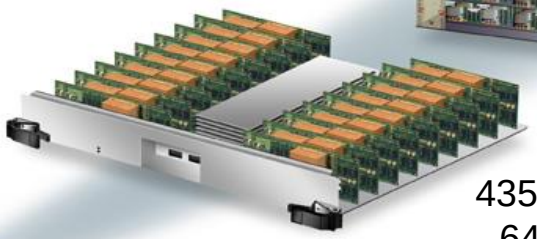
**Rack**
32 Node Cards
1024 chips, 4096 procs



1 PF/s +
144 TB +

14 TF/s
2 TB

**Node Card**
(32 chips 4x4x2)
32 compute, 0-2 IO cards



435 GF/s
64 GB

**Compute Card**
1 chip, 20 DRAMs

13.6 GF/s
2.0 GB DDR
Supports 4-way SMP

**Chip**
4 processors

13.6 GF/s
8 MB EDRAM

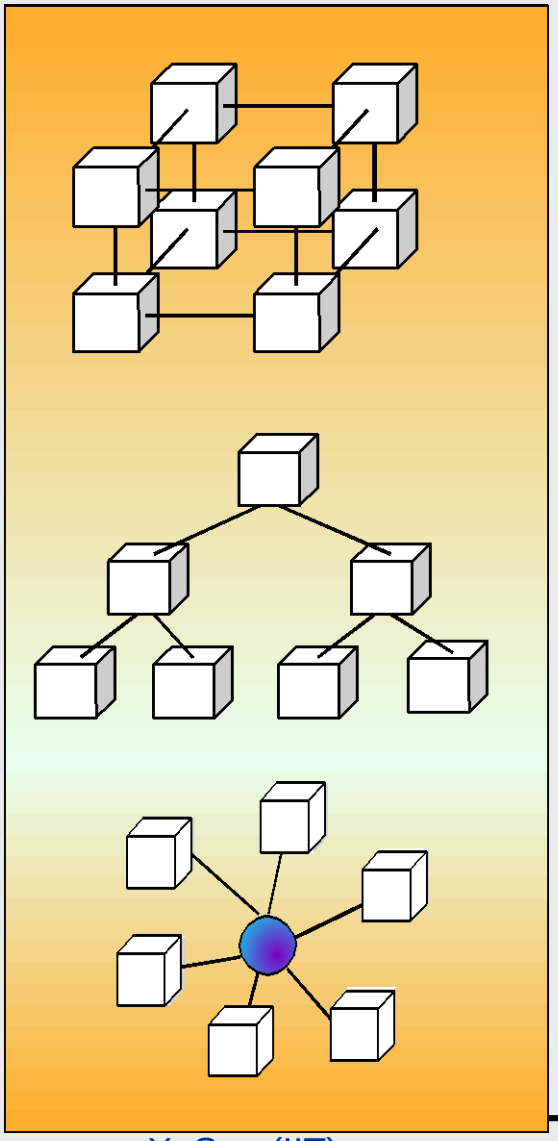
# Performance (June 2008 Top 500)

- 4 of the top 20 machines are BlueGene/P

Intrepid at ANL

| Racks | Installs | Ranking |
|---|---|---|
| 40 | 1 | 3 |
| 16 | 1 | 6 |
| 10 | 1 | 8 |
| 8 | 1 | 13 |
| 4 | 1 | 37 |
| 3 | 2 | 51 |
| 2 | 2 | 74 |

# Blue Gene/P Interconnection Networks



## 3 Dimensional Torus

- Interconnects all compute nodes
  - Communications backbone for computations
- Adaptive cut-through hardware routing
- 3.4 Gb/s on all 9 node links (5.1 GB/s per node)
- 0.5 µs latency between nearest neighbors, 5 µs to the farthest
  - MPI: 3 µs latency for one hop, 10 µs to the farthest
- 1.7/2.6 TB/s bisection bandwidth

## Collective Network

- Interconnects all compute and I/O nodes
- One-to-all broadcast functionality
- Reduction operations functionality
- 6.8 Gb/s of bandwidth per link
- Latency of one way tree traversal 2 µs, MPI 5 µs

## Low Latency Global Barrier and Interrupt

- Latency of one way to reach all 72K nodes 0.65 µs, MPI 1.6 µs

## Other networks

- 10Gb Functional Ethernet
  - I/O nodes only
- 1Gb Private Control Ethernet
  - Provides JTAG access to hardware. Accessible only from Service Node system

# BG/P Software

- Compute Node Kernel (CNK)
  - Minimal kernel – handles signals, function shipping syscalls to I/O nodes, starting/stopping jobs, threads
  - Not much else
  - Very “linux-like”
    - Missing some system calls (fork() mostly)
      - Limited support for mmap(), execve()
    - But, most apps that run on Linux work out-of-the-box on BG/P

# BG/P Software

- I/O Node Kernel
  - Linux (MCP)
  - Very minimal distribution (almost everything on the I/O node is in busybox)
  - Only connection from compute nodes to outside world
  - Handles syscalls (ie fopen()) and I/O requests

# BG/P Software

- Control System
  - Runs on service node
  - Compute nodes are stateless
    - State information is stored in db2 databases
  - Database also monitors performance, environmentals, etc
  - Boots blocks, monitors jobs, etc
  - Interaction via Navigator, LoadLeveler, etc.

# Summary

- Overview of distributed memory parallel architectures
- Interconnection networks
- Message passing issues
- Case study: IBM SP2, Blue Gene/P

# Basic Communication Operations

# Outline

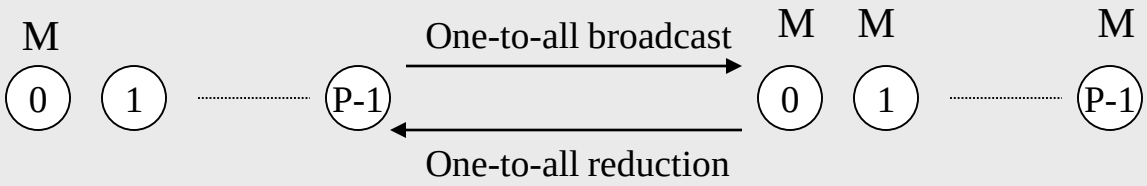
- One-to-all broadcast
- All-to-all broadcast
- One-to-all scatter
- All-to-all scatter
- Circular shift

- Reading:
  - Chpt 4 of Kumar's book

# One-to-all Broadcast

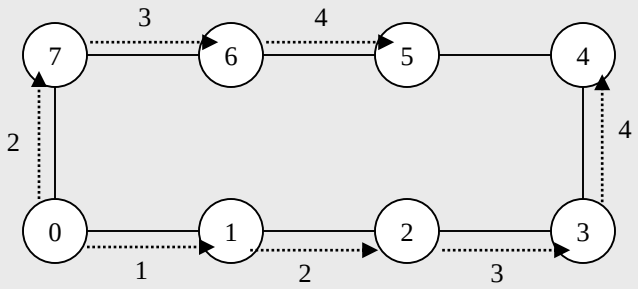
- Algorithms often require a processor to send identical data to all other processors (to a subset of the processors is called multicast). This operation is called a one-to-all broadcast or singlenode broadcast
- At the start of a singlenode broadcast, the sender has *m* words of data that needs to be sent, at the end there are *p* copies of this data, one on each processor
- The *dual* of a broadcast operation is a all-to-one reduction or singlenode reduction
- At the start of singlenode reduction each process has *m* words of data, the reduction combines all the data from processors using an associative operator to produce *m* words at the receiver
- Naïve singlenode broadcast or reduction using *p-1* steps

# One-to-all Broadcast



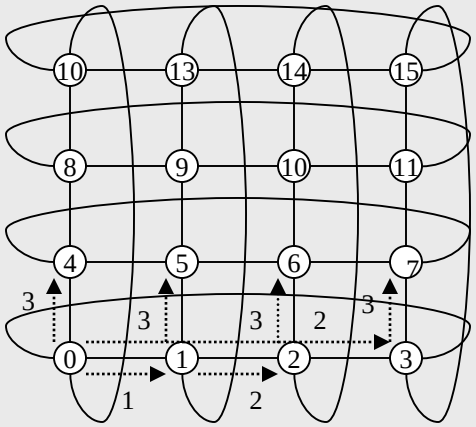
# SF Routing on Ring

- Source send message on both outgoing links in first step
- All other processors receive on a link and transmit on other link
- $\left\lceil \frac{p}{2} \right\rceil$ steps and $(t_s + t_w m)\left\lceil \frac{p}{2} \right\rceil$ cost



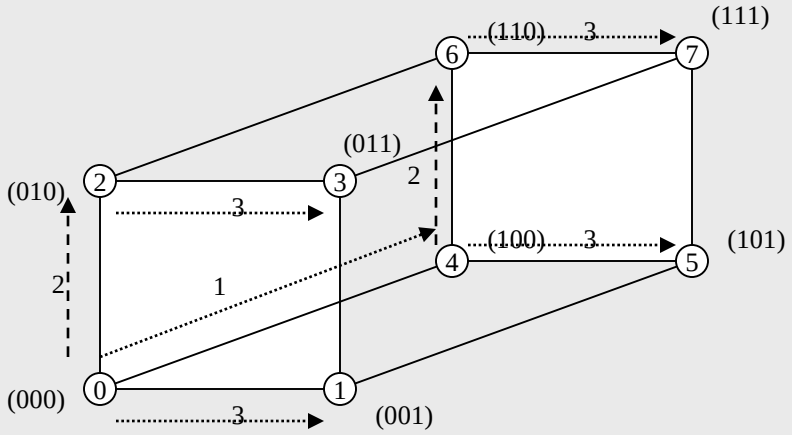
# SF Routing on 2d Torus

- Each row or column of the torus can be regarded as a ring
- Using ring method for the row to which the sending processor belongs; then use ring method for every column
- $2\left\lceil \frac{\sqrt{p}}{2} \right\rceil$ steps and $2(t_s + t_w m)\left\lceil \frac{\sqrt{p}}{2} \right\rceil$ cost

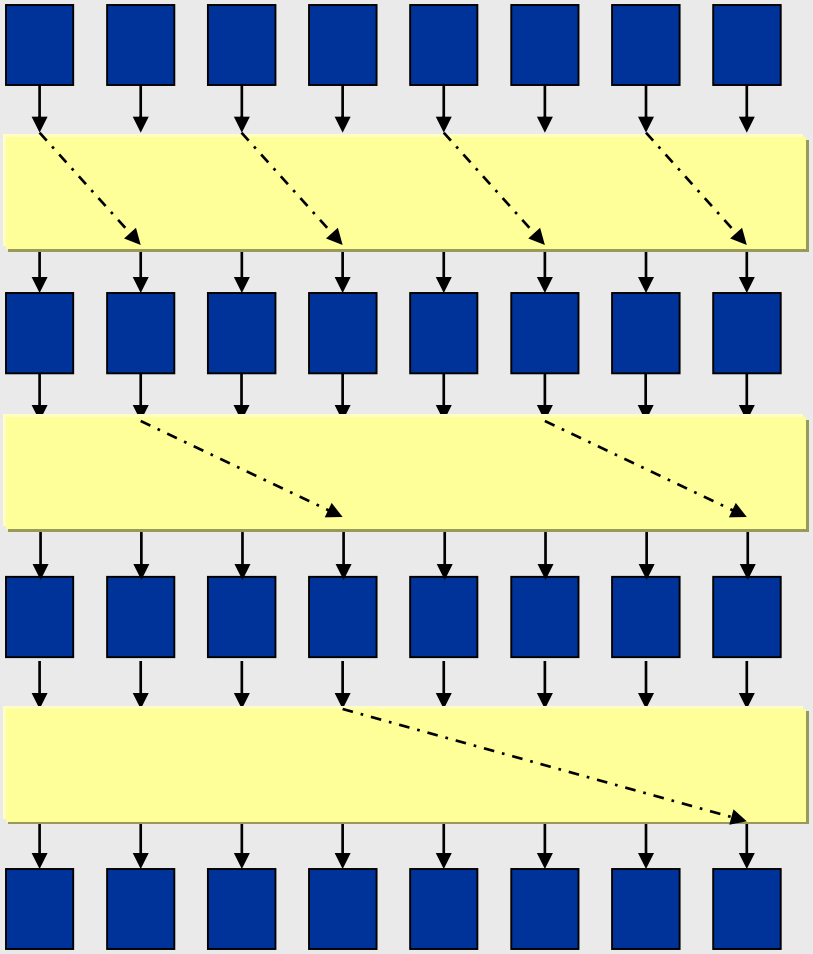


# SF Routing on Hypercube

- Takes *log(p)* steps for a *p* processor hypercube
- In the *i*-th step, all processors that have the message transmit it to the neighboring processor that differs in the *i-th* most significant bit
- $(t_s + t_w m)\log(p)$ cost



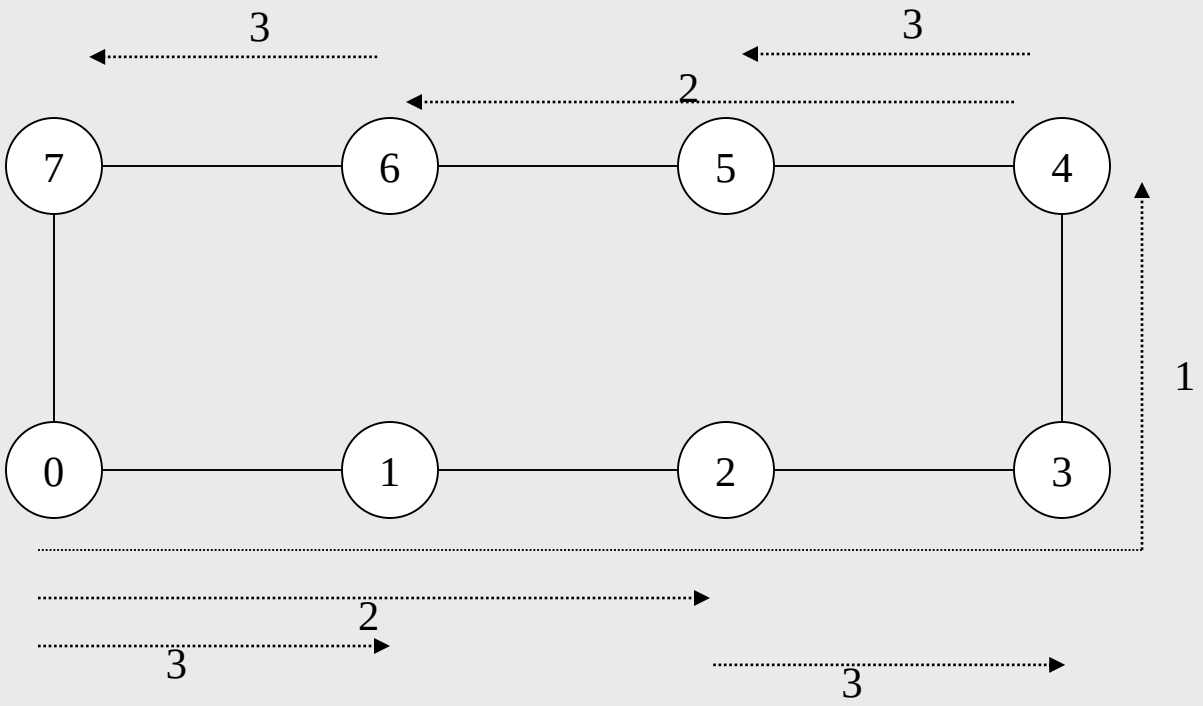
# Tree Reduction (Data gathering/scattering)



# CT Routing on Ring

- Algorithm similar to one used for hypercube; takes *log(p)* steps
- In step *i*, message is sent to processor at a distance $\frac{p}{2^i}$
- All messages flow in the same direction
- $t_s \log(p) + t_w m \log(p) + t_h (p-1)$

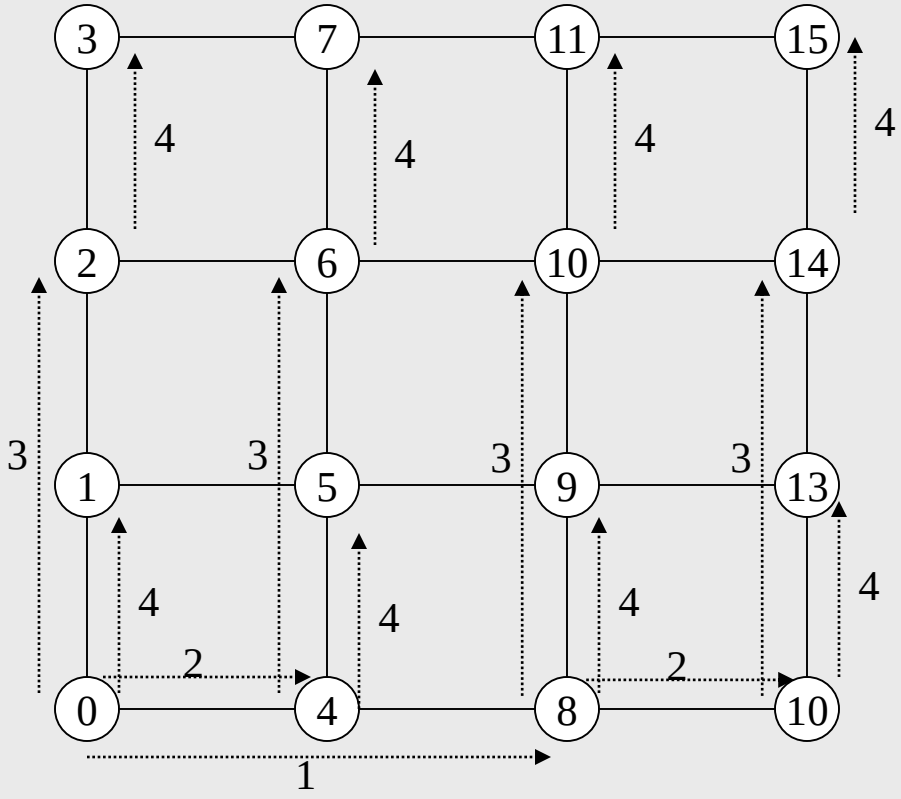
# CT Routing on Ring



# CT Routing on 2d Torus

- Apply ring algorithm for the processor row of sender
- Now use ring algorithm for all processor column
- $2\log(\sqrt{p})$ steps
- $(t_s + t_w m)\log(p) + 2t_h(\sqrt{p} - 1)$ cost

# CT Routing on 2d Torus



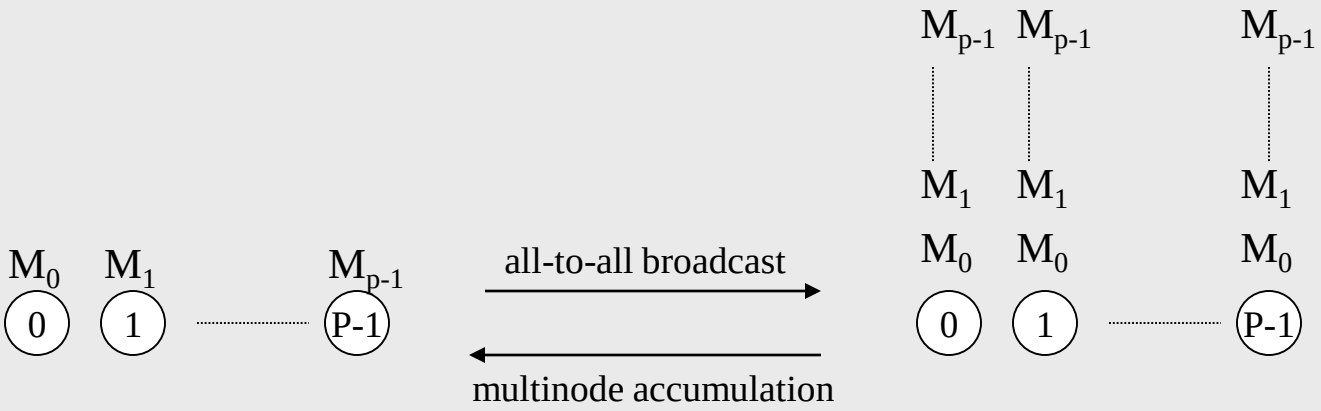
# CT Routing on Hypercube

- For hypercube, cut-through does not provide benefits because of the use of only single link communication

# All-to-all Broadcast

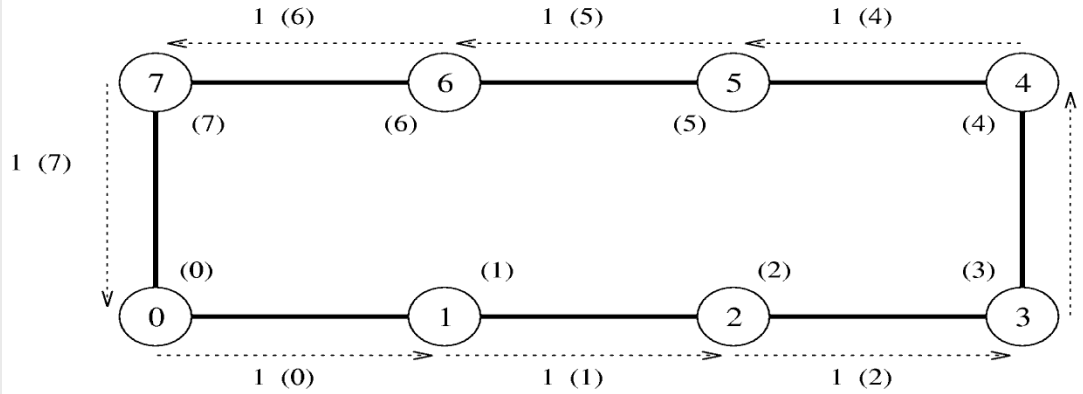
- Algorithms often require each processor to send identical data to all other processors or a subset of processors. This operation is called all-to-all broadcast or a multinode broadcast
- At the start of a multinode broadcast, each processor has *m* words of data; at the end each processor has a copy of the *m* words that originated at each of the other processors
- The dual of this operation is a all-to-all reduction or a multinode reduction
- At the start of a multinode reduction each processor has *m* words of data, the reduction combines all the data from processors using an associative operator to produce *m* words that are available at all the processors
- Naïve multinode broadcast or reduction using p singlenode broadcasts

# All-to-all Broadcast

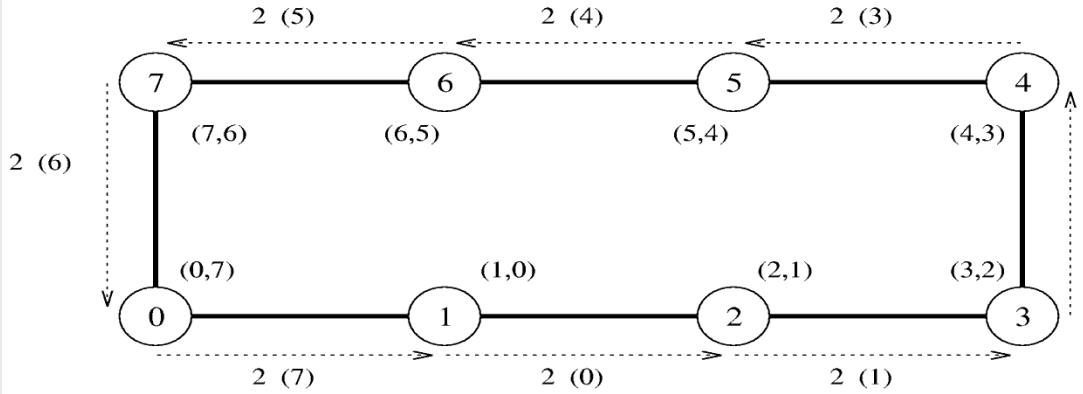


# SF Routing on Ring

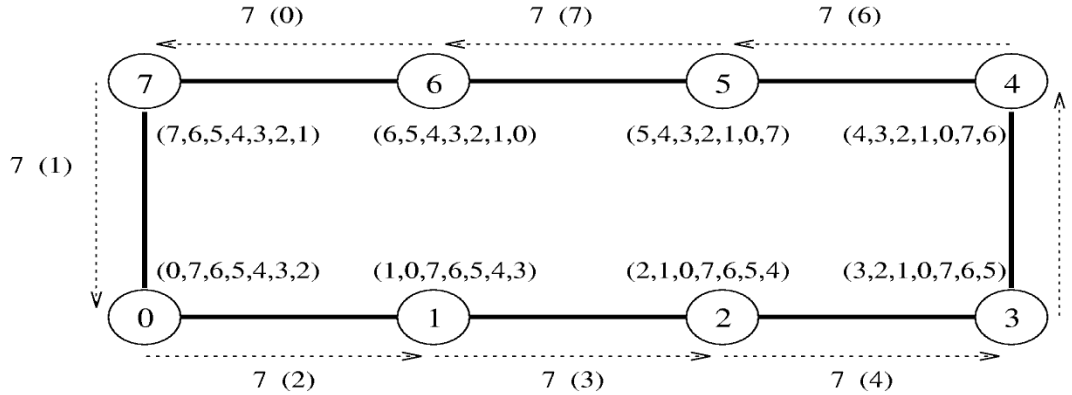
- Every processor sends its message to the next processor on the ring in the first step
- In every subsequent step, all processors receive a message from the previous processor and send it to the next processor after retaining a copy for themselves
- $p-1$ steps
- $(t_s + t_w m)(p-1)$ cost




First communication step




Second communication step

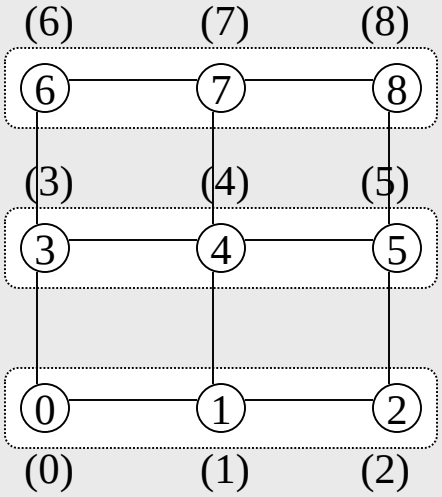



Seventh communication step

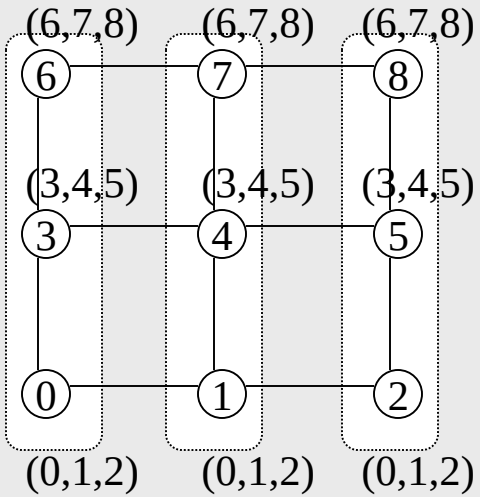
# SF Routing on 2d Torus

- Use ring method for each row of the torus; compose all $\sqrt{p}$ messages received into a single message and use the ring method for every column
- $2(\sqrt{p}-1)$ steps
- $2t_s(\sqrt{p}-1)+t_w m(p-1)$ cost

# SF Routing on 2d Torus



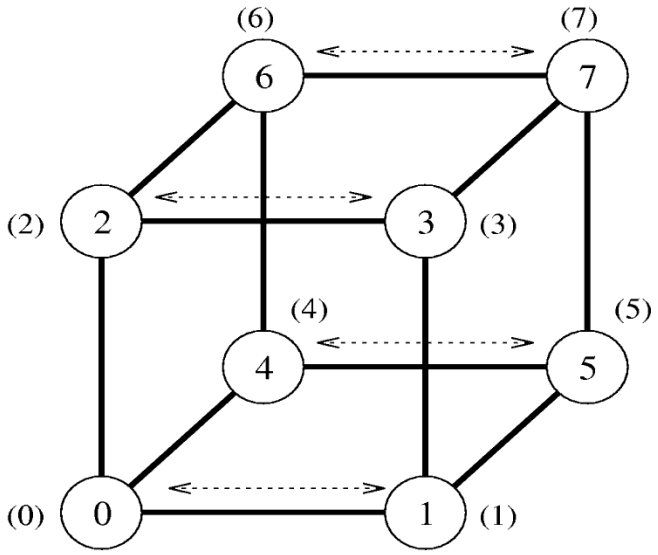

(a) Initial data distribution



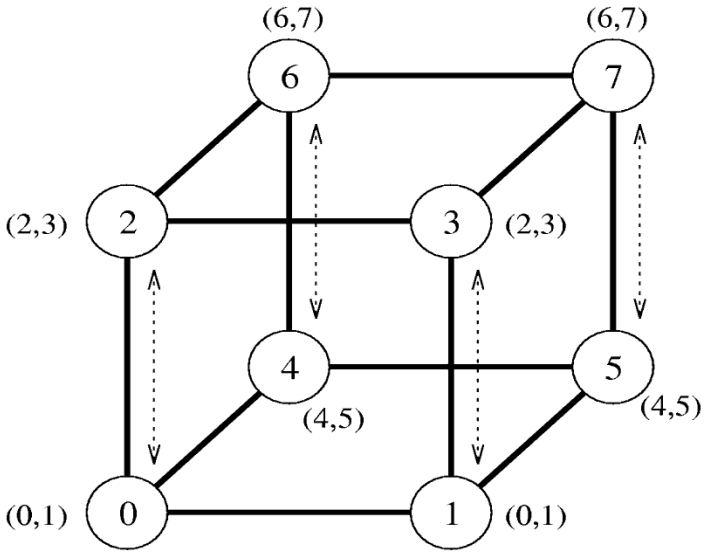

(b) Data distribution after rowwise broadcast

# SF Routing on Hypercube

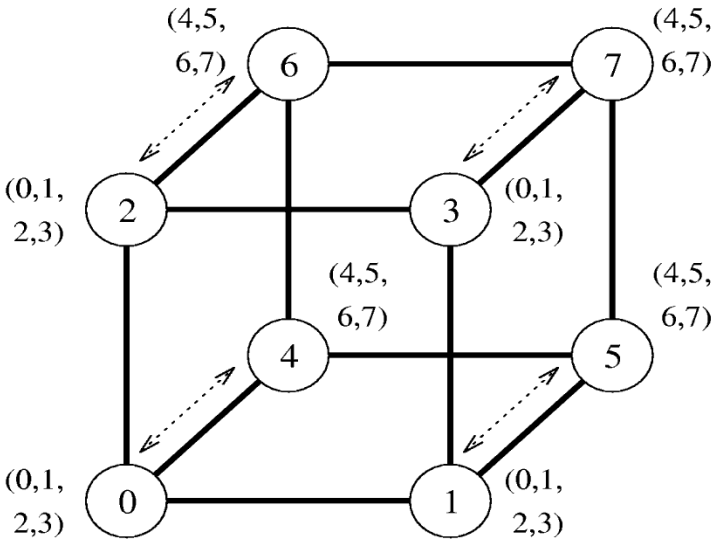
- Takes $\log(p)$ steps for a *p*-processor hypercube
- In the ith step, every processor exchanges messages with the neighboring processor that differs in the ith most significant bit. At each step larger message are built out of smaller messages for subsequent steps
- $t_s \log(p) + t_w m(p-1)$ cost



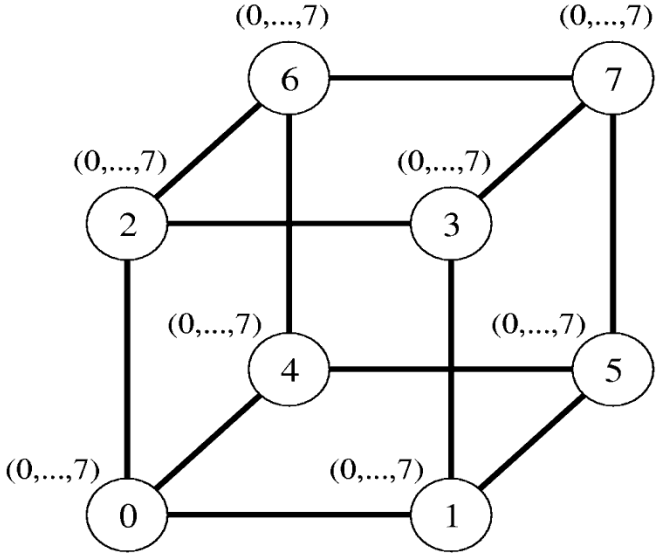

(a) Initial distribution of messages




(b) Distribution before the second step

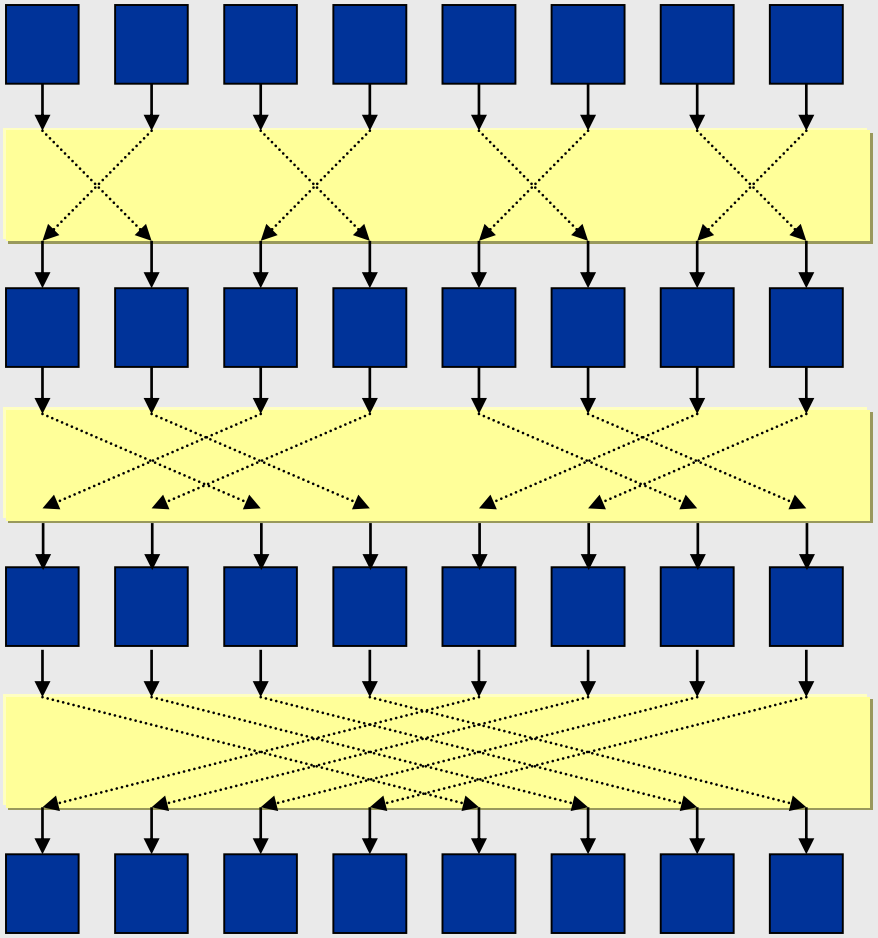



(c) Distribution before the third step




(d) Final distribution of messages

# All-to-All Total Data Exchange



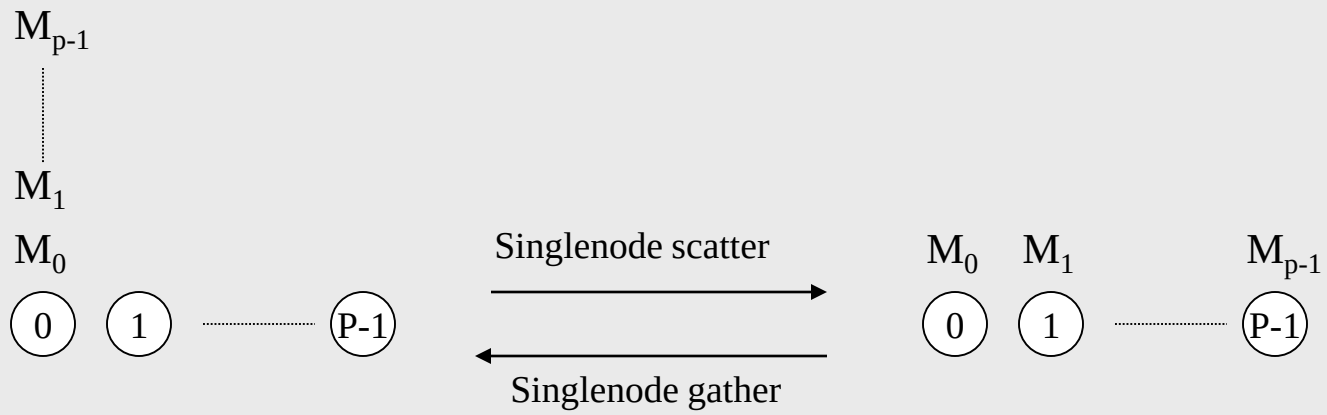
# CT Routing

- Cut-through routing does not provide any benefits over store-and-forward for all-to-all broadcasts
- The one-to-all algorithm used for the ring and the torus creates contention in the all-to-all case which renders it useless
- Hypercube is especially good for SF routing

# One-to-all Scatter

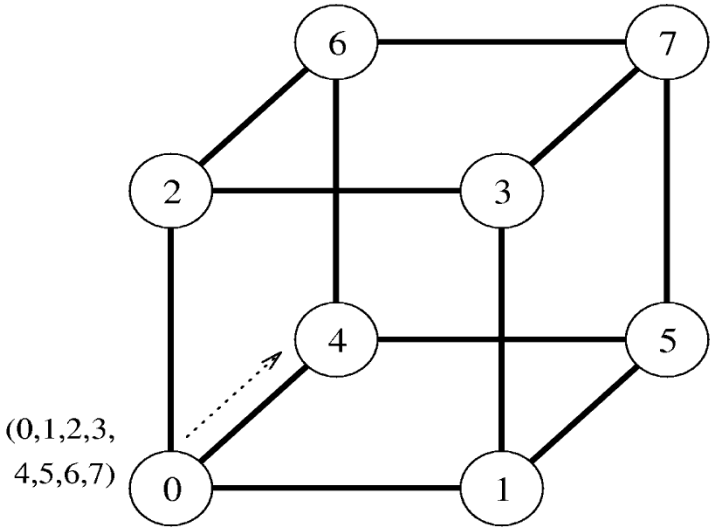
- Algorithms often require a processor to send different data to each of the other processors or each of a subset of processors, this is called a one-to-all personalized communication or singlenode scatter
- At the start of a singlenode scatter, the source processor has (*p-1*) messages of *m* words, each of which needs to be sent to each of the other processors; at the end each of them has *m* words
- The dual of this operation is a all-to-one personalized communication or singlenode gather
- At the start of a singlenode gather each processor has *m* bytes of data, the gather combines all the data from processors to produce *m(p-1)* words at the receiver
- Naïve singlenode scatter or gather using *(p-1)* steps

# One-to-all Scatter

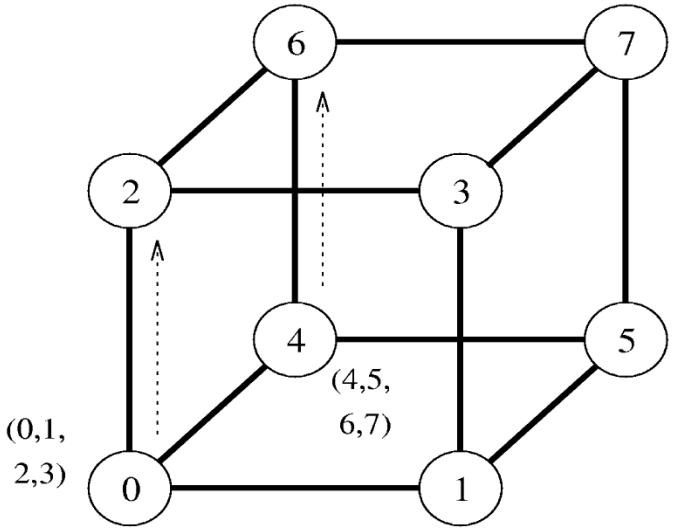


# SF Routing on Hypercube

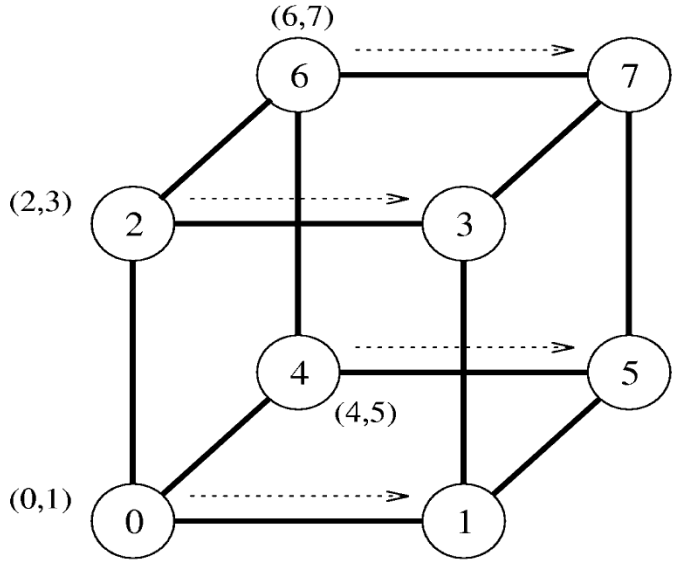
- Takes $\log(p)$ steps for a *p*-processor hypercube
- In the ith step, all processors that have messages transmit half of them to the neighboring processor that differs in the ith most significant bit
- $t_s \log(p) + t_w m(p-1)$ cost



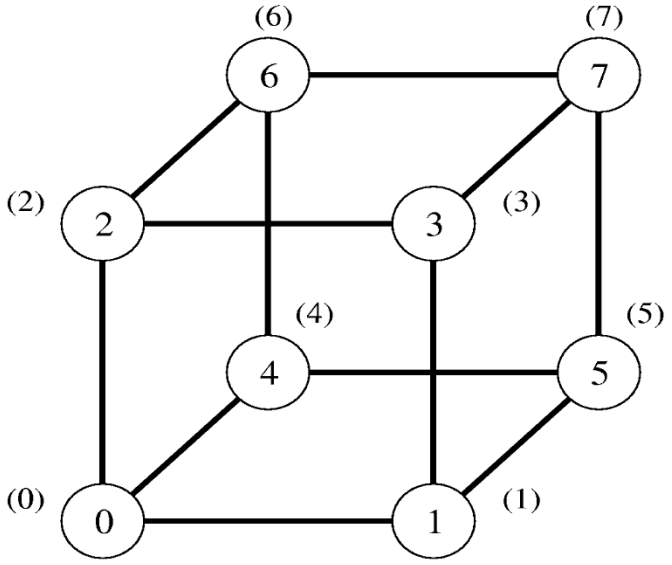

(a) Initial distribution of messages




(b) Distribution before the second step

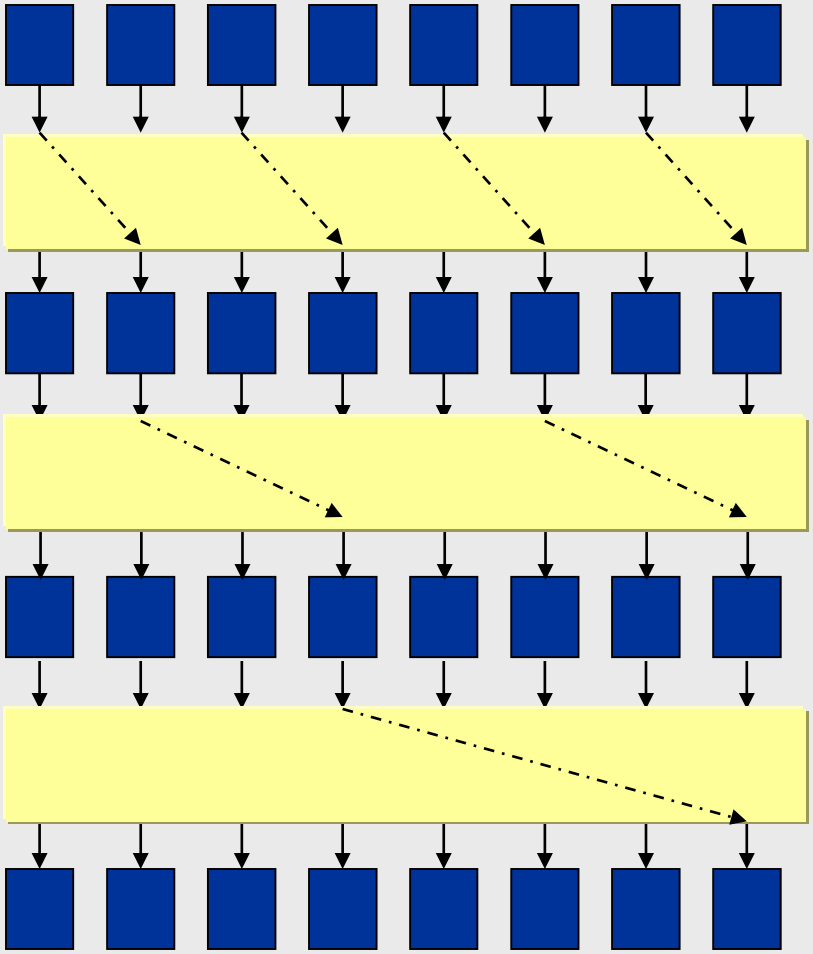



(c) Distribution before the third step




(d) Final distribution of messages

# Tree Reduction (Data gathering/scattering)



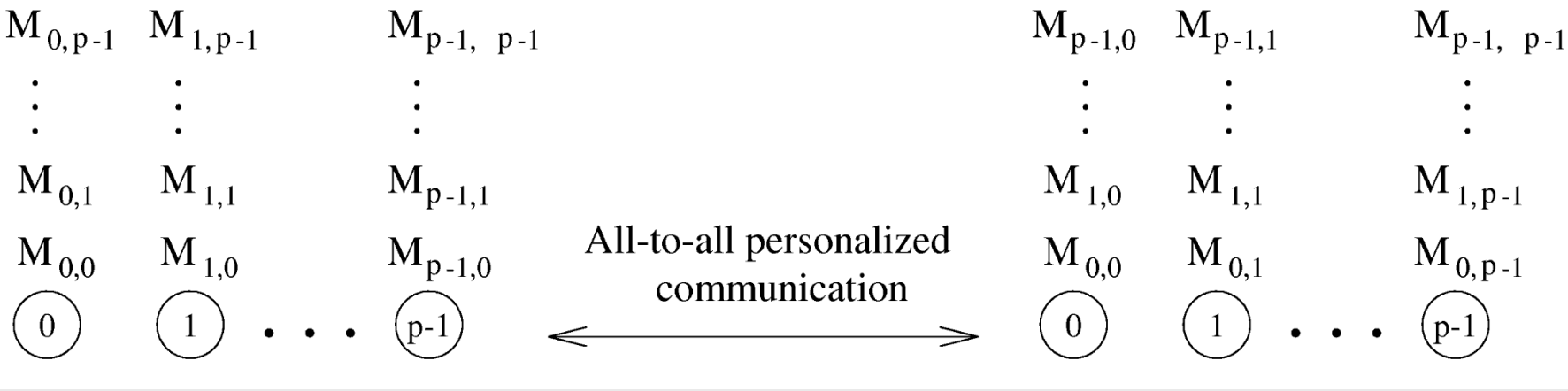
# CT Routing

- Cut-through routing does not provide any benefits over store-and-forward for singlenode scatter due to exclusive single link transfers

# All-to-all Scatter

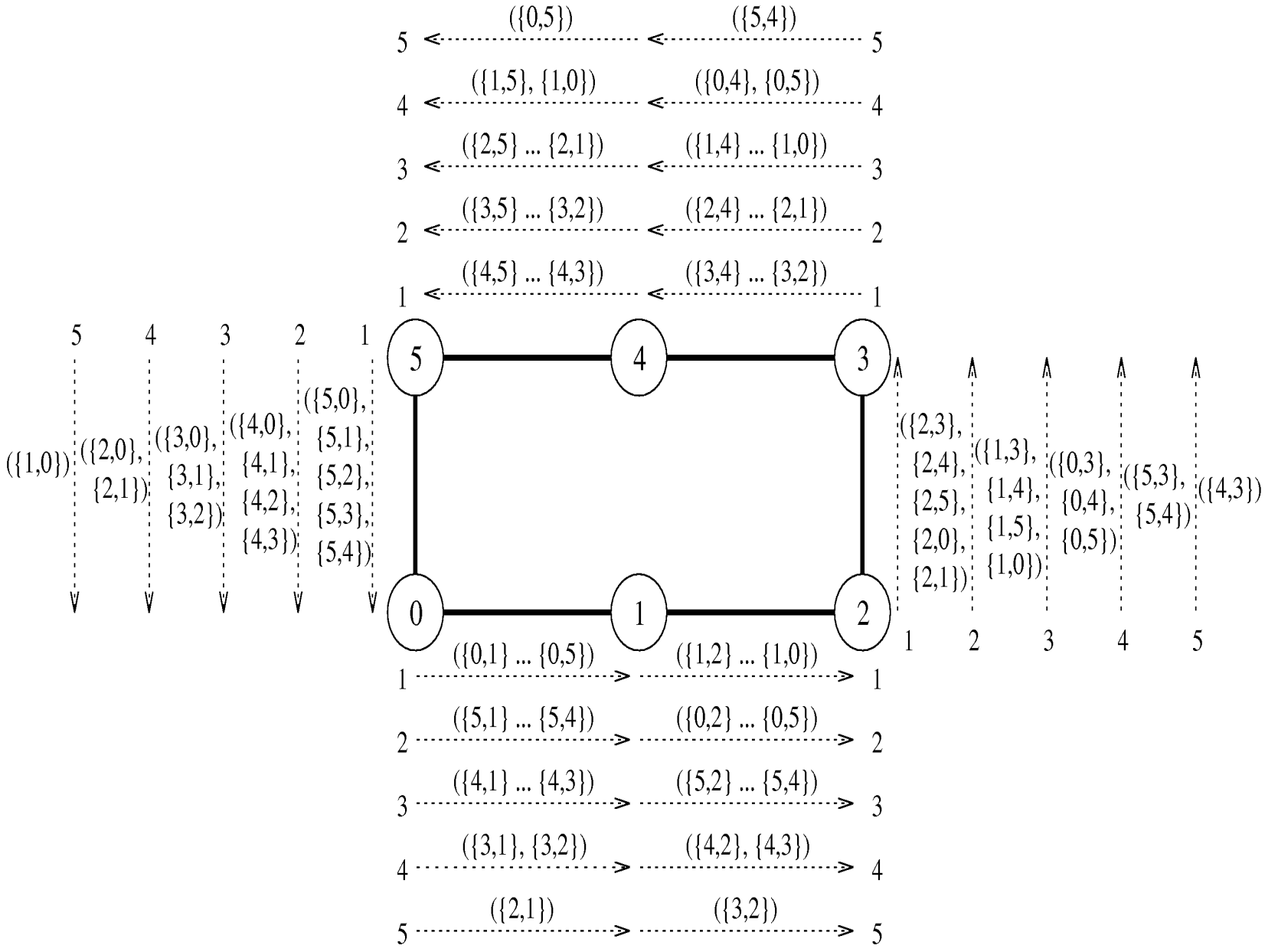
- Algorithms often require each processor to send different data to each of the other processors or each of a subset of processors
- This operation is called a all-to-all personalized communication or a multinode scatter
- At the start of a multinode scatter each processor has *(p-1)m* words of data; at the end each processor has a copy of the *m* words that originated at each of the other processors, *(p-1)m* words in all
- Naïve multinode scatter using *p* singlenode scatters

# All-to-all Scatter



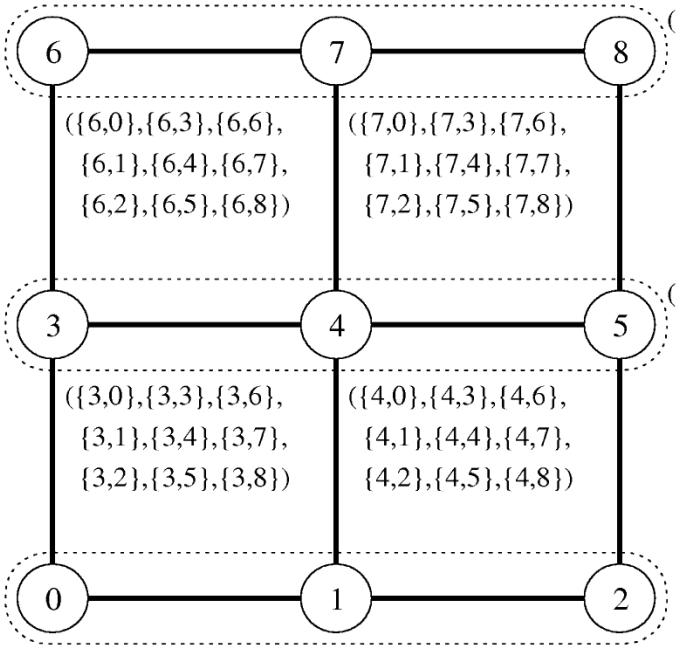
# SF Routing on Ring

- Every processor consolidates all the data to be sent and sends a single message to the next processor on the ring in the first step
- In every subsequent step, processors retain part of the message received for themselves and send the rest to the next processor on the ring
- $(p-1)$ steps
- $(t_s + \frac{1}{2} t_w mp)(p-1)$ cost

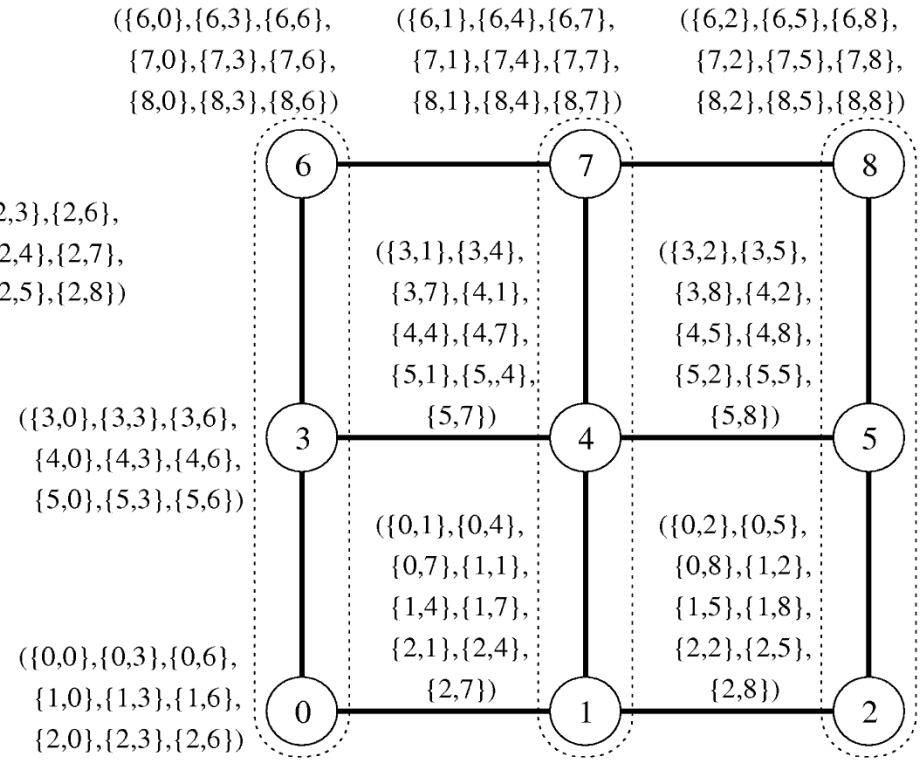

({0,5})
({5,4})
5
5
({1,5}, {1,0})
({0,4}, {0,5})
4
4
({2,5} ... {2,1})
({1,4} ... {1,0})
3
3
({3,5} ... {3,2})
({2,4} ... {2,1})
2
2
({4,5} ... {4,3})
({3,4} ... {3,2})
1
1
5
4
3
2
1
({1,0})
({2,0}, {2,1})
({3,0}, {3,1}, {3,2})
({4,0}, {4,1}, {4,2}, {4,3})
({5,0}, {5,1}, {5,2}, {5,3}, {5,4})
5
4
3
0
1
2
({2,3}, {2,4}, {2,5}, {2,0}, {2,1})
({1,3}, {1,4}, {1,5}, {1,0})
({0,3}, {0,4}, {0,5})
({5,3}, {5,4})
({4,3})
1
2
3
4
5
({0,1} ... {0,5})
({1,2} ... {1,0})
1
1
({5,1} ... {5,4})
({0,2} ... {0,5})
2
2
({4,1} ... {4,3})
({5,2} ... {5,4})
3
3
({3,1}, {3,2})
({4,2}, {4,3})
4
4
({2,1})
({3,2})
5
5

# SF Routing on 2d Torus

- Processors consolidate messages into groups meant for each processor column
- The ring method is applied to each processor row independently
- Now the messages in each processor are sorted into groups meant for each processor row
- $2(\sqrt{p}-1)$ steps
- $(2t_s + t_w mp)(\sqrt{p}-1)$ cost



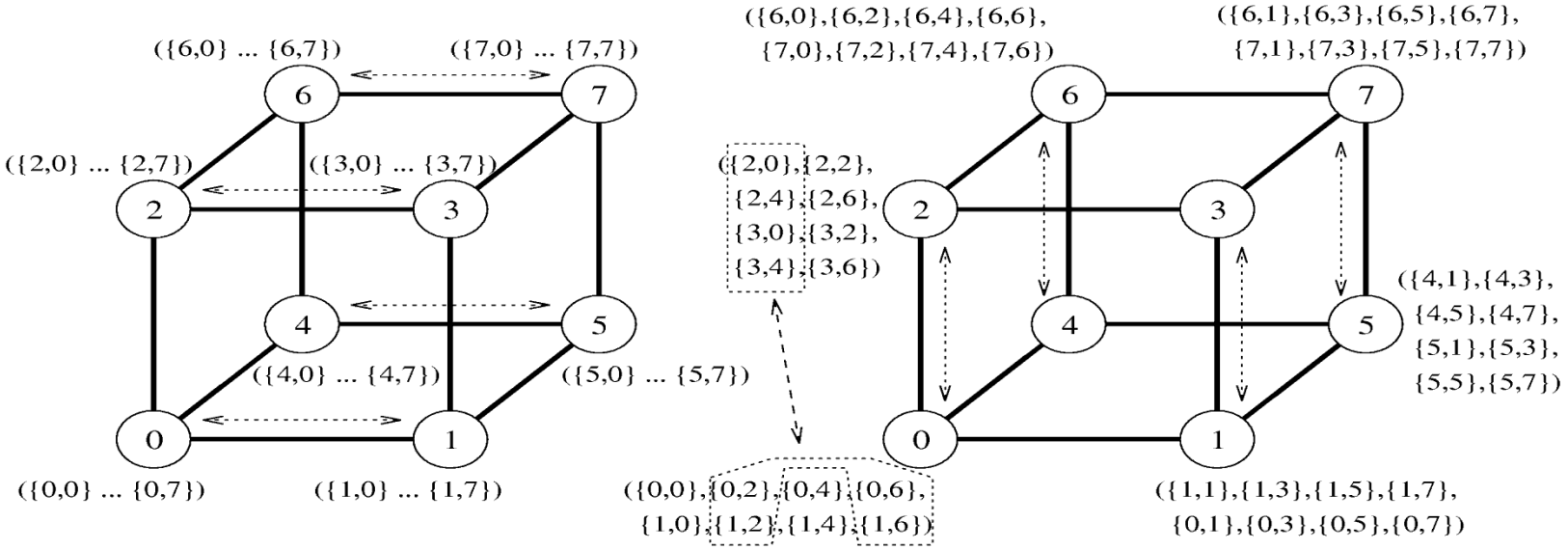

(a) Data distribution at the beginning of first phase




(b) Data distribution at the beginning of second phase

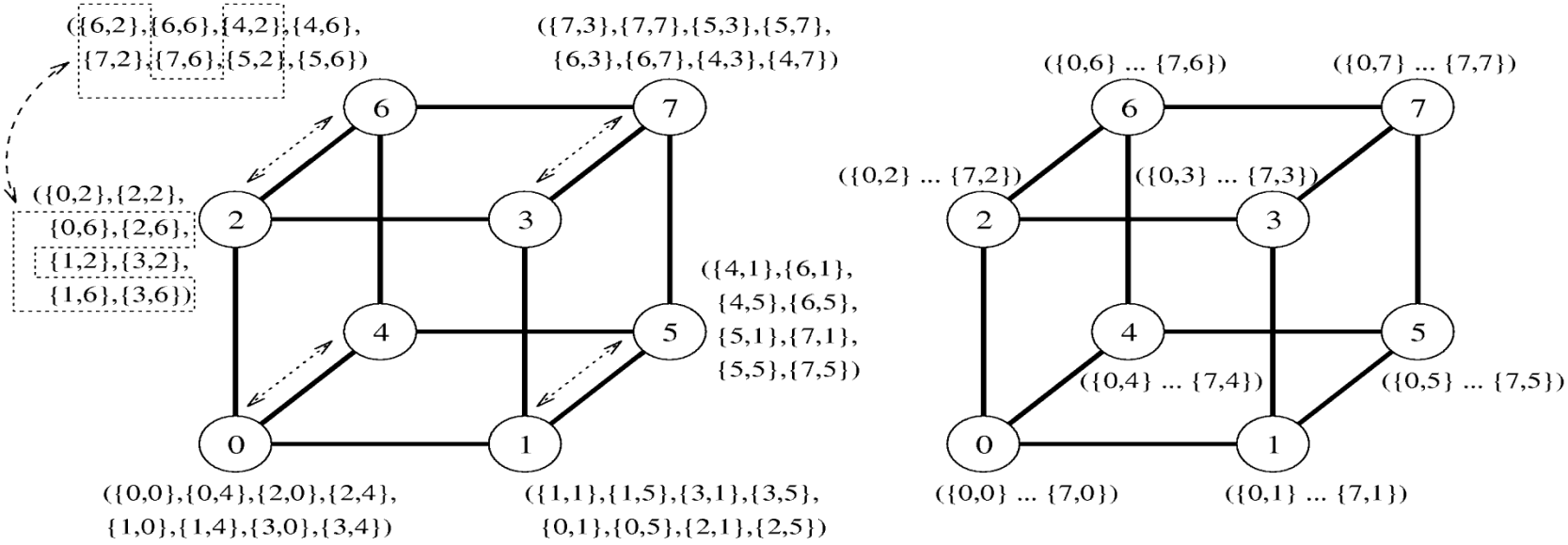
# SF Routing on Hypercube

- Take $\log(p)$ steps for a *p* processor hypercube
- In the ith step, every processor exchanges messages with the neighboring processor that differs in the ith least significant bit
- At each stage, every processor holds p messages, p/2 of these are consolidated into a single message for exchange with a neighboring processor
- $(t_s + \frac{1}{2} t_w mp)\log(p)$ cost




(a) Initial distribution of messages

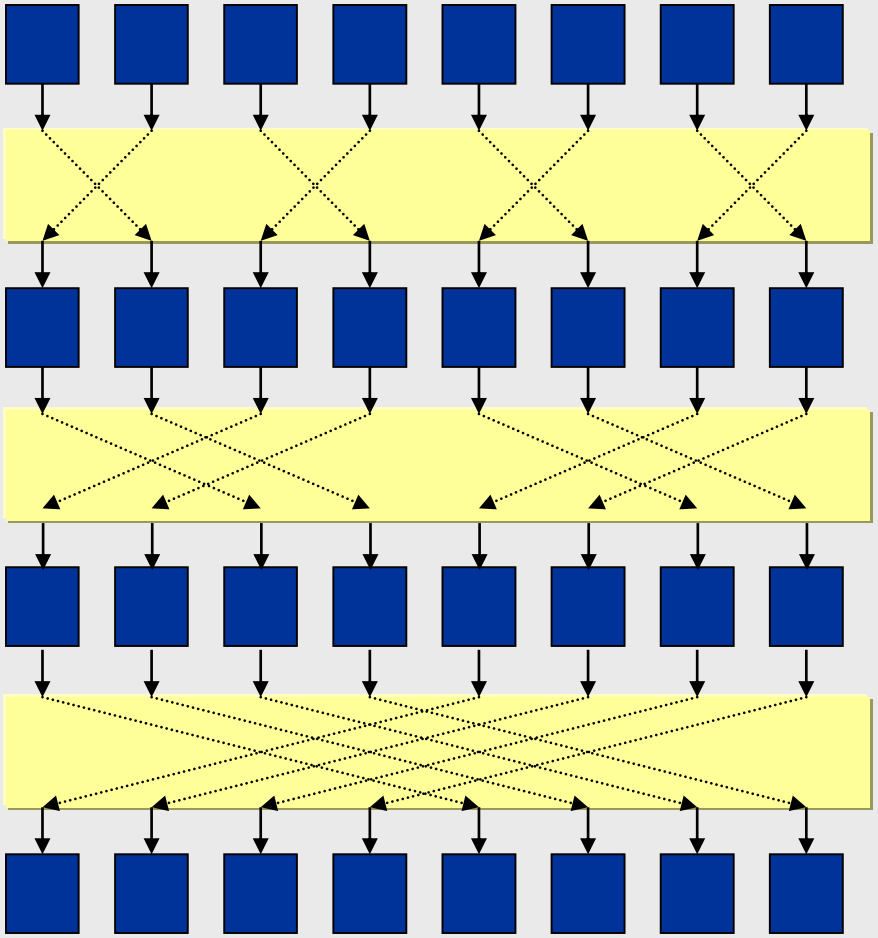
(b) Distribution before the second step




(c) Distribution before the third step

(d) Final distribution of messages

# Personalized All-to-All Total Data Exchange



# CT Routing

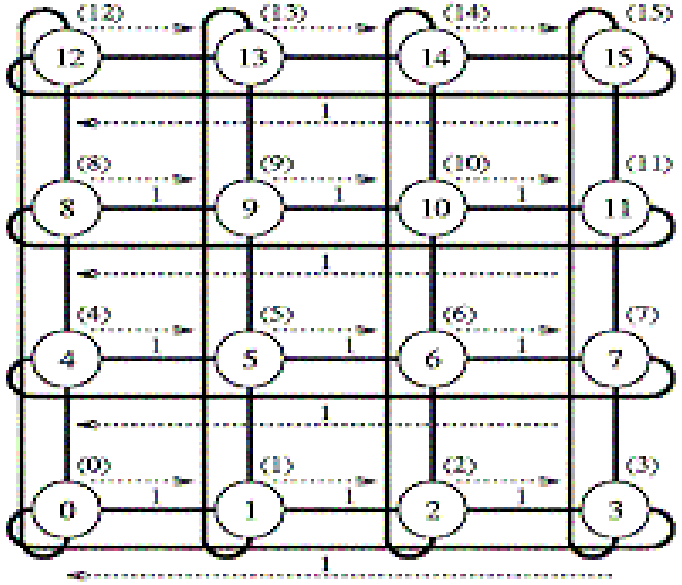
- Cut-through routing does not provide any benefits over store-and-forward for multinode scatters on rings and meshes.

# Circular Shift

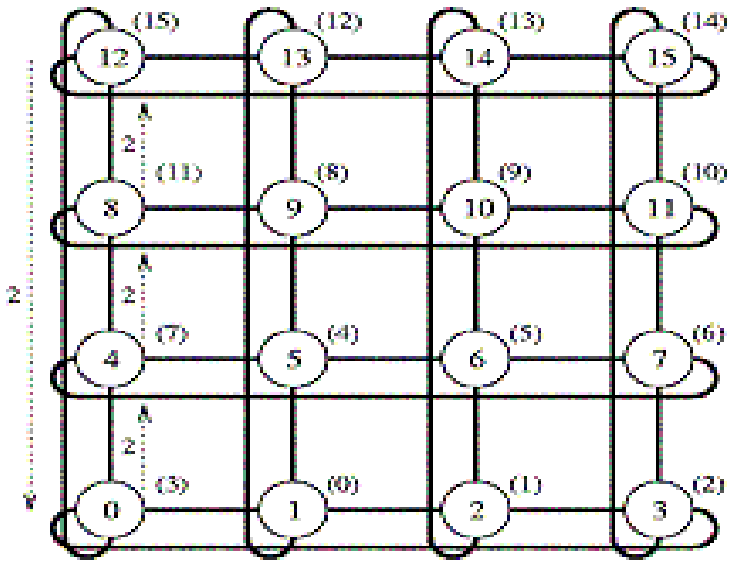
- A permutation is a one-to-one message passing operation in which every processor sends a message to another unique processor
- A circular q-shift is a permutation in which proc i sends a message to proc (i+q) mod p
- Many matrix computations and pattern matching algorithms need such a shift operation
- Shift is an intuitive operation for a ring network, it takes min(q,p-q) steps involving single link communication

# SF Routing on Torus

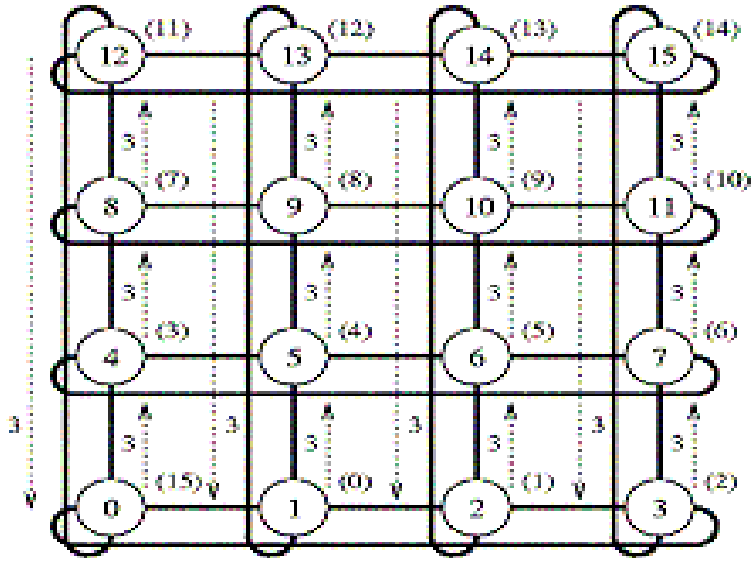
- Entire data is first shifted simultaneously by $q \bmod \sqrt{p}$ steps along the rows
- Need a compensatory column shift for messages that need to actually go to the next row
- Now shift data along columns using $\left\lfloor \frac{q}{\sqrt{p}} \right\rfloor$ steps
- Can select direction of shift in the row and column phases to minimize the total number of steps
- $(t_s + t_w m)(2\left\lfloor \frac{\sqrt{p}}{2} \right\rfloor + 1)$ is the maximum cost
- Example 5-shift on a 4 by 4 mesh, first shift 3 steps by the rows (backward 1), and then shift 1 step along the columns (figure 22 text)



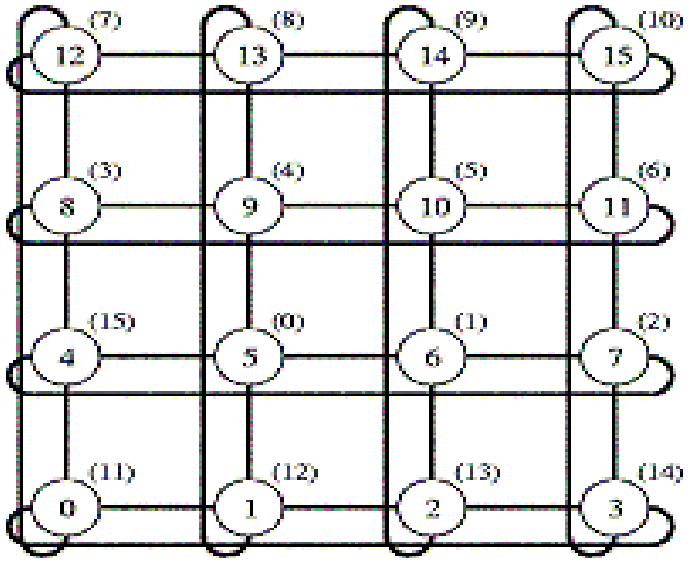

(a) Initial data distribution and the first communication step




(b) Step to compensate for backward row shifts




(c) Column shifts in the third communication step




(d) Final distribution of the data

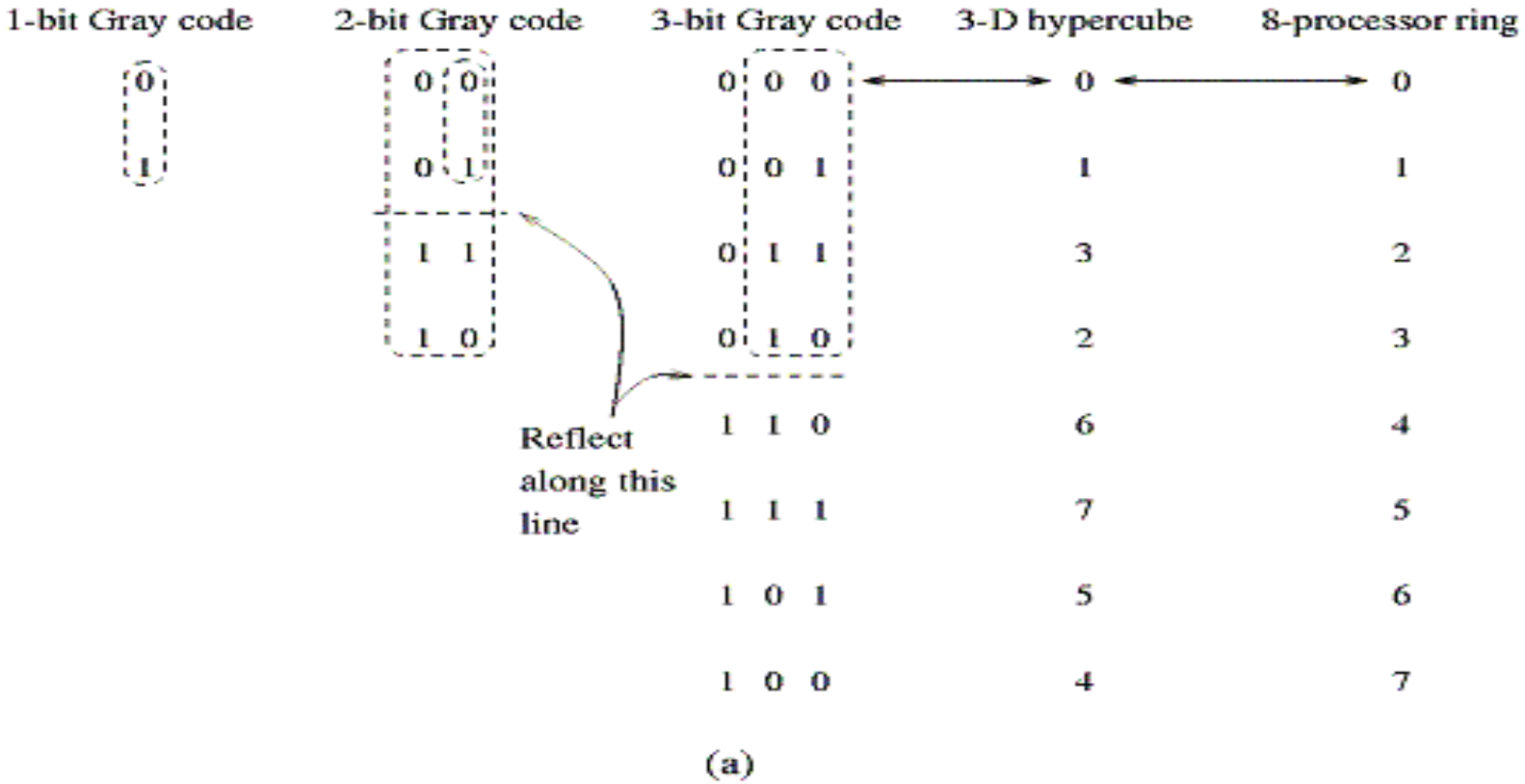
# SF Routing on Hypercube

- We embed a ring into the hypercube using a Gray code mapping
- Proc i on the ring is mapped onto proc G(i,d) where d=log(p) is the number of dimensions in the hypercube and G is defined by

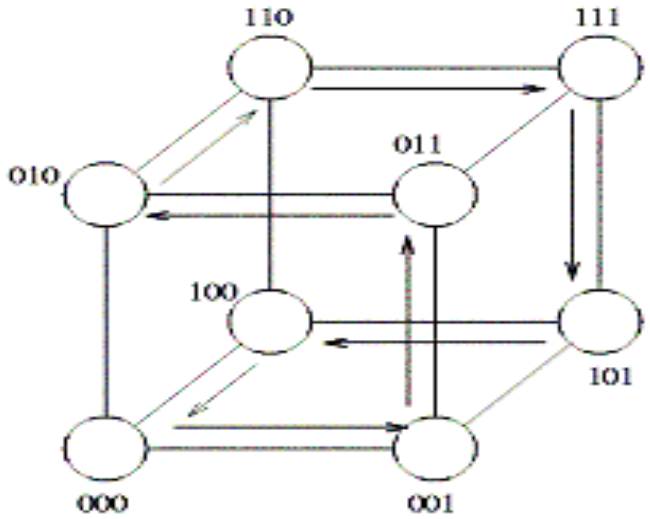
$$G(0,1) = 0$$

$$G(1,1) = 1$$

$$G(i, x+1) = \begin{cases} G(i,x) & i < 2^x \\ 2^x + G(2^{x+1} - 1 - i, x) & i \geq 2^x \end{cases}$$

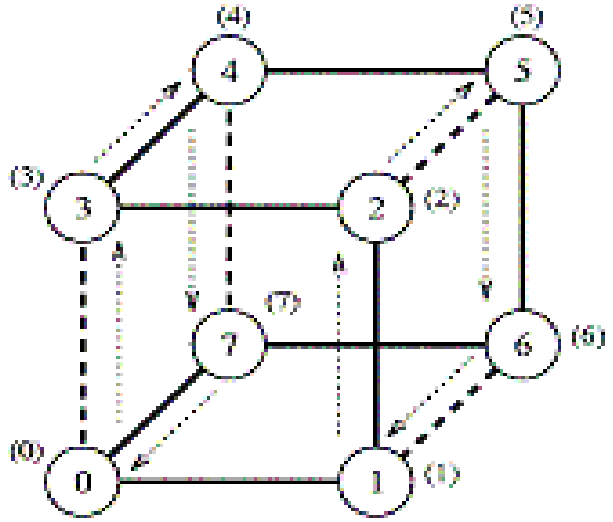

1-bit Gray code
2-bit Gray code
3-bit Gray code
3-D hypercube
8-processor ring
0
1
0 0
0 1
1 1
1 0
0 0 0
0 0 1
0 1 1
0 1 0
1 1 0
1 1 1
1 0 1
1 0 0
0
1
3
2
6
7
5
4
0
1
2
3
4
5
6
7
Reflect along this line

(a)

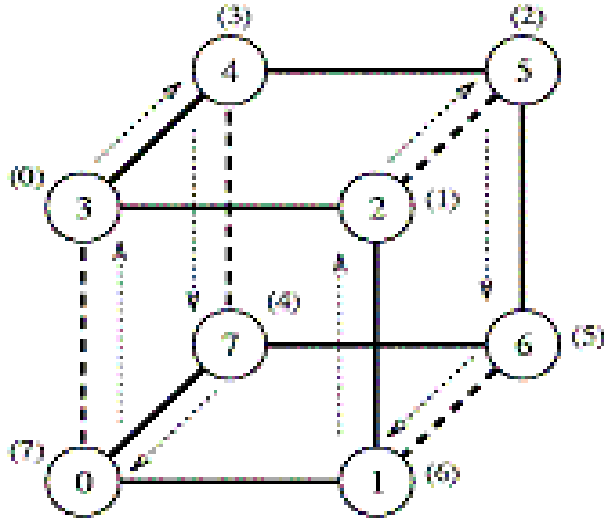

110
111
011
010
100
101
000
001

# SF Routing on Hypercube

- A property of the Gray code mapping is that processors that are at a distance $2^i$ on the ring are exactly two links apart on the hypercube (except for i=0 where it is one link)
- To perform a q shift, we decompose q as the sum of powers of 2. if s is the number of powers of 2 that are contained in q, we need s phases to perform a q shift
- In each phase except for a 1 shift, we perform two communication steps
- $(t_s + t_w m)(2\log p - 1)$ is the maximum cost
- Example a 5-shift is performed by a 4-shift + a 1-shift ($2^{2+}$ $2^{0)}$

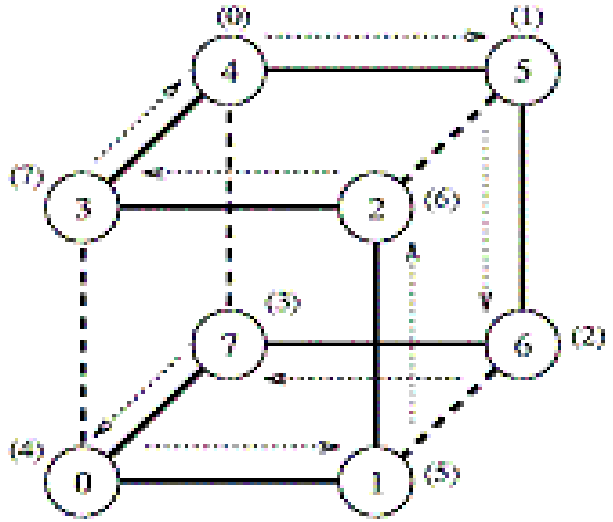



First communication step of the 4-shift

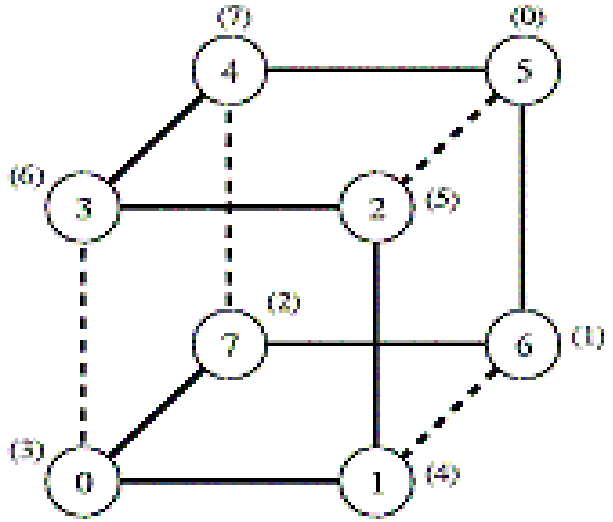



Second communication step of the 4-shift

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

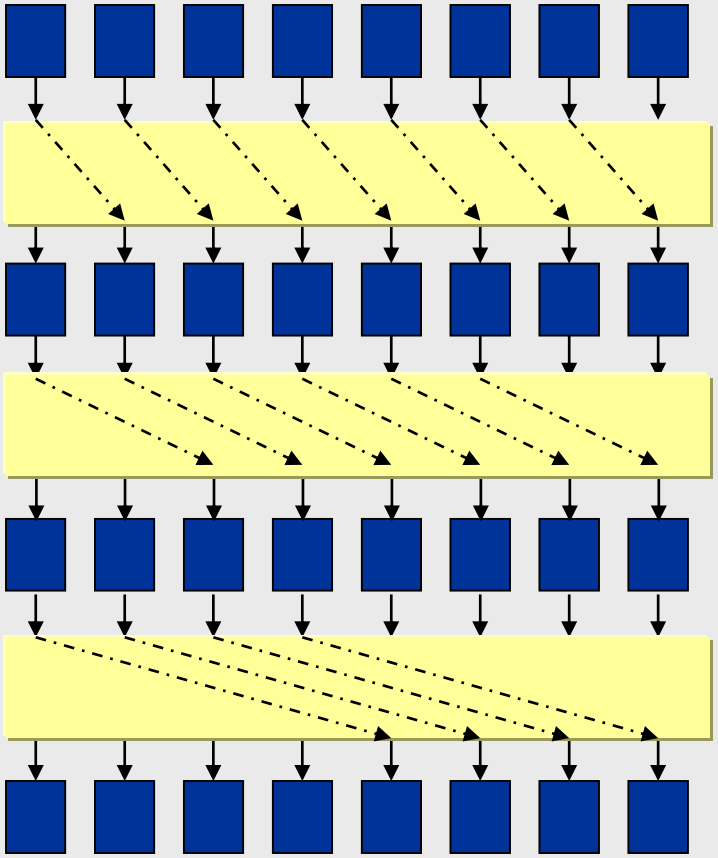



(b) The second phase (a 1-shift)




(c) Final data distribution after the 5-shift

# Partial Sum Gathering



# Summary

- One-to-all broadcast
- All-to-all broadcast
- One-to-all scatter
- All-to-all scatter
- Circular shift
- Partial sum (homework)
- Reading:
  - Chpt 4 of Kumar's book