Collective Communication

CS 406/531, Week 12

Group Communication

- **Motivation: accelerate interaction patterns among groups**
- **Approach: collective communication**
  - **—whole group works together *collectively* to realize a communication**
  - **—constructed from pairwise point-to-point communications**
- **Implementation strategy**
  - **—standard library of common collective operations**
  - **—leverage target architecture for efficient implementation**
- **Benefits of standard library implementations**
  - **—reduce development effort and cost for parallel codes**
  - **—improve performance through efficient implementations**
  - **—improve quality of scientific applications**

Topics for Today

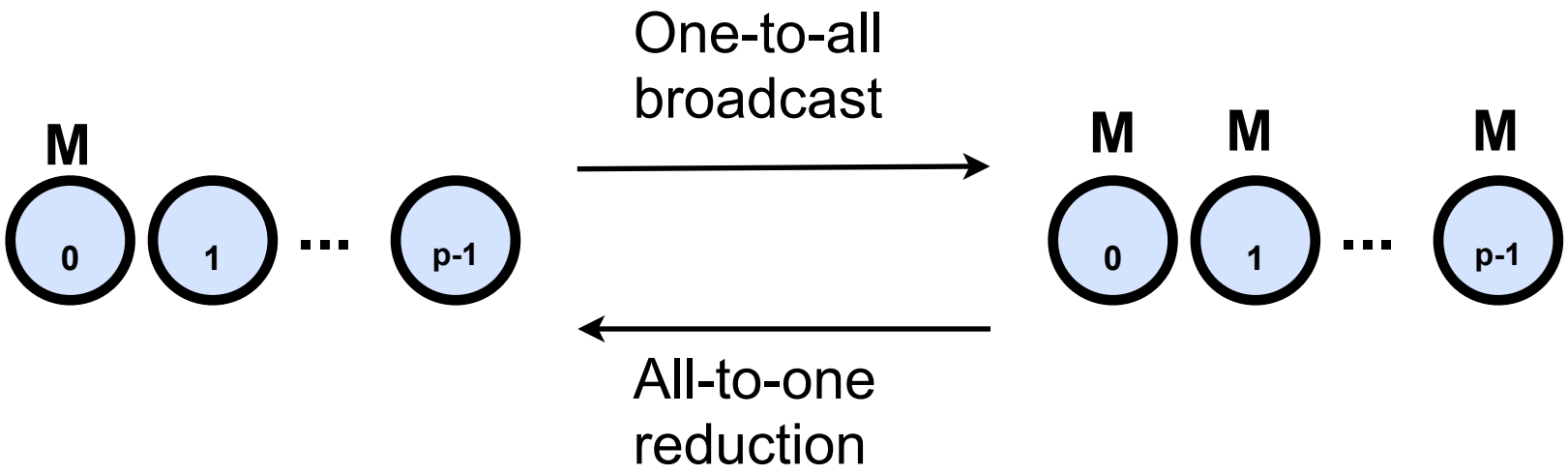
- One-to-all broadcast and all-to-one reduction
- All-to-all broadcast and reduction
- All-reduce and prefix-sum operations
- Scatter and gather
- All-to-all personalized communication
- Circular shift
- Optimizing collective patterns

Assumptions

- Network is bidirectional
- Communication is single-ported
  - node can receive only one message per step
- Communication cost model
  - message of size m, no congestion, time = $t_s + t_w m$
  - congestion: model by scaling t_w

One-to-All and All-to-One

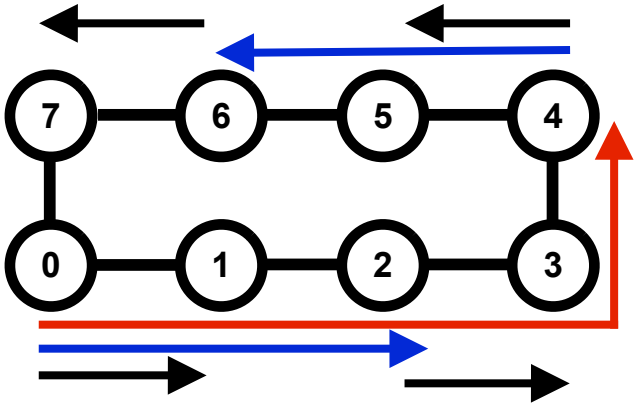
- **One-to-all broadcast**
  - —a processor has *m* units of data that it must send to everyone
- **All-to-one reduction**
  - —each processor has *m* units of data
  - —data items must be combined using some associative operator
    - e.g. addition, min, max
  - —result must be available at a target processor



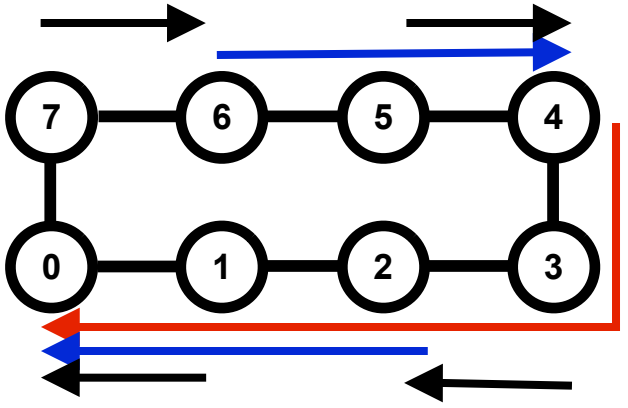
One-to-All and All-to-One on a Ring

- **Broadcast**
  - —naïve solution
    - source sends *p - 1* messages to the other *p - 1* processors
  - —use recursive doubling
    - source sends a message to a selected processor

      yields two independent problems over halves of the machine

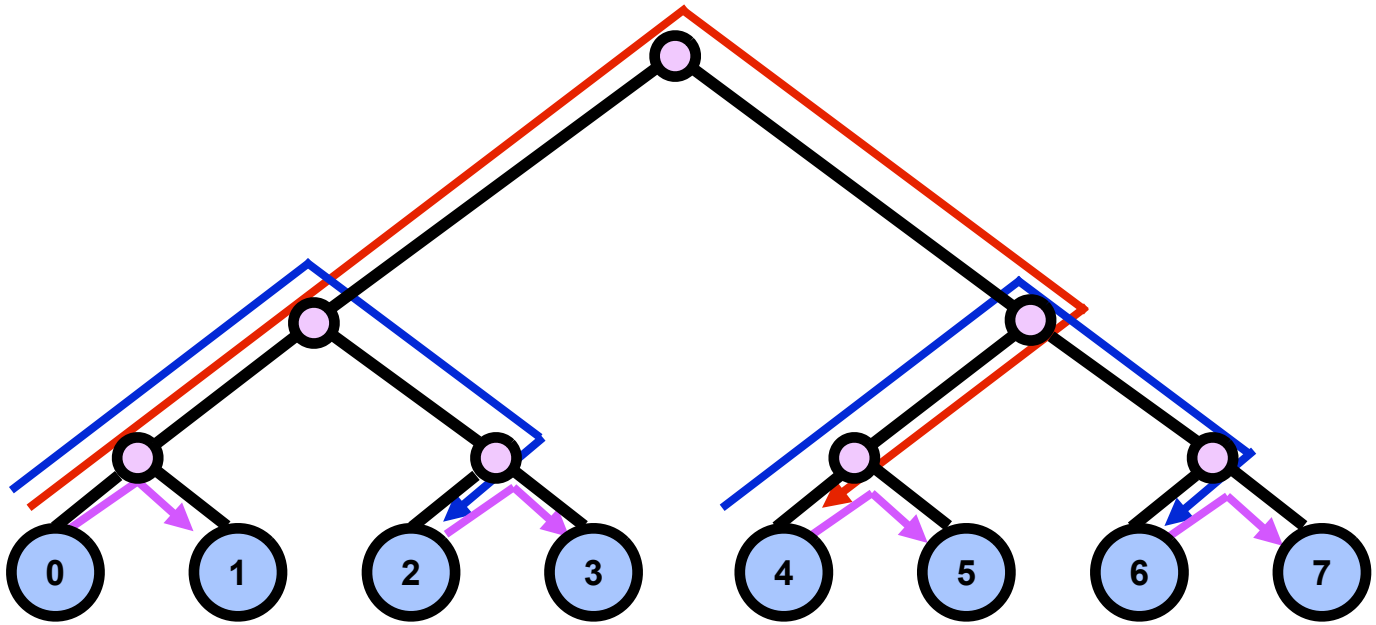


- **Reduction**
  - — invert the process



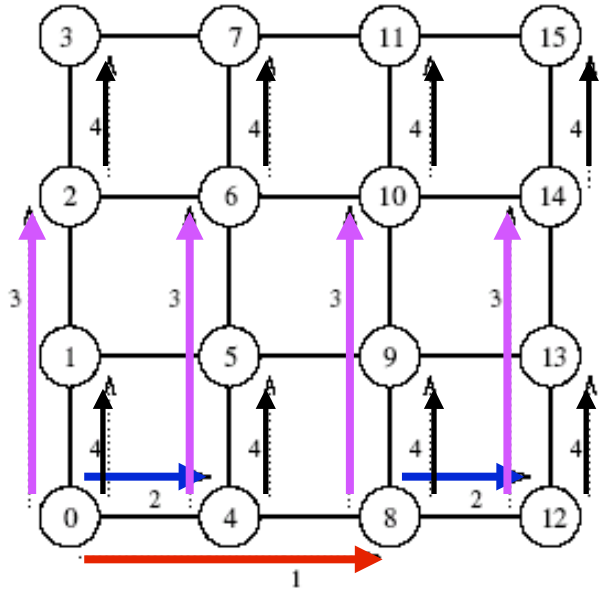
Broadcast on a Balanced Binary Tree

- Consider processors arranged in a dynamic binary tree
  - —processors are (logically) at the leaves
  - —internal nodes are routing nodes
- Assume leftmost processor is the root of the broadcast
- Use recursive doubling strategy: log *p* stages



Broadcast and Reduction on a 2D Mesh

- **Consider a mesh of p nodes**
  - **view each row as a linear array of $\sqrt{p}$ nodes**
  - **view each column as a linear array of $\sqrt{p}$ nodes**
- **Two step broadcast and reduction operations**
  1. **perform the operation along a row**
  2. **perform the operation along each column concurrently**

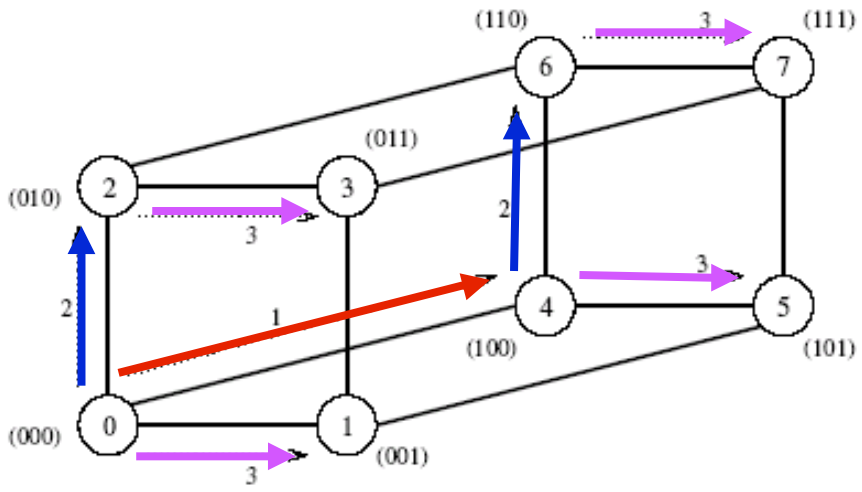


broadcast on 4 x 4 mesh

- **Generalizes to higher dimensional meshes**

Broadcast and Reduction on a Hypercube

- **Consider hypercube with 2^d nodes**
  - **—view as *d*-dimensional mesh with two nodes in each dimension**
- **Apply mesh algorithm to a hypercube**
  - **—*d* (= *log p*) steps**



Broadcast and Reduction Algorithms

- **Each of aforementioned broadcast/reduction algorithms**
  - **—adaptation of the same algorithmic template**
- **Next slide: a broadcast algorithm for a hypercube of 2^d nodes**
  - **—can be adapted to other architectures**
  - **—in the following algorithm**
    - ***my_id* is the label for a node**
    - ***X* is the message to be broadcast**

One-to-All Broadcast Algorithm

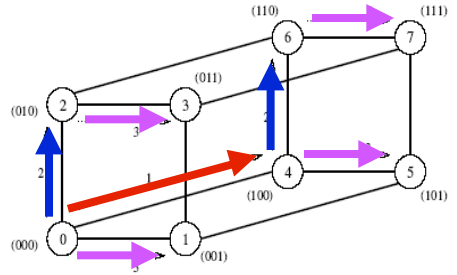
```
1.   procedure GENERAL_ONE_TO_ALL_BC(d, my_id, source, X)
2.   begin
3.       my_virtual_id := my_id XOR source;
4.       mask := 2^d − 1;
         for i := d − 1 downto 0 do      /* Outer loop */
             mask := mask XOR 2^i;       /* Set bit i of mask to 0 */
7.           if (my_virtual_id AND mask) = 0 then
8.               if (my_virtual_id AND 2^i) = 0 then
9.                   virtual_dest := my_virtual_id XOR 2^i;
10.                  send X to (virtual_dest XOR source);
             /* Convert virtual_dest to the label of the physical destination */
11.              else
12.                  virtual_source := my_virtual_id XOR 2^i;
13.                  receive X from (virtual_source XOR source);
             /* Convert virtual_source to the label of the physical source */
14.              endelse;
15.      endfor;
16.  end GENERAL_ONE_TO_ALL_BC
```

position relative to source

I am the master of a 2^i subcube

// even

// odd



One-to-all broadcast of a message *X* from *source* on a hypercube

All-to-One Reduction Algorithm

```
1.    procedure ALL_TO_ONE_REDUCE(d, my_id, m, X, sum)
2.    begin
3.        for j := 0 to m − 1 do sum[j] := X[j];
4.        mask := 0;
5.        for i := 0 to d − 1 do
              /* Select nodes whose lower i bits are 0 */
6.            if (my_id AND mask) = 0 then
7.                if (my_id AND 2^i) ≠ 0 then // odd
8.                    msg_destination := my_id XOR 2^i;
9.                    send sum to msg_destination;
10.               else                                  // even
11.                   msg_source := my_id XOR 2^i;
12.                   receive X from msg_source;
13.                   for j := 0 to m − 1 do
14.                       sum[j] := sum[j] + X[j];
15.               endelse;
16.           mask := mask XOR 2^i;   /* Set bit i of mask to 1 */
17.       endfor;
18.   end ALL_TO_ONE_REDUCE
```

I am the master of a 2^i subcube

All-to-One sum reduction on a *d*-dimensional hypercube

Each node contributes msg *X* containing *m* words, and node 0 is the destination

Broadcast/Reduction Cost Analysis

Hypercube

- **Log p point-to-point simple message transfers**
  - **each message transfer time: $t_s + t_w m$**
- **Total time**

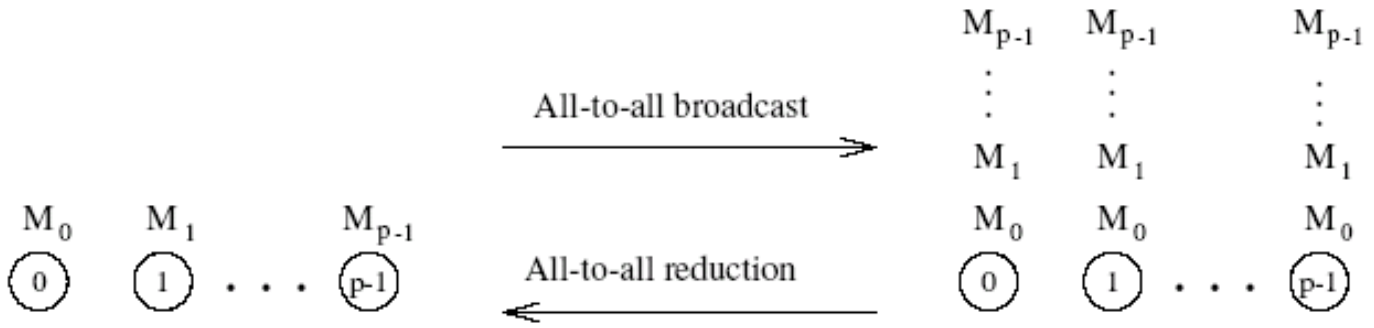
$$T = (t_s + t_w m) \log p.$$

All-to-All Broadcast and Reduction

Each processor is the source as well as destination

- **Broadcast**
  - **—each process broadcasts its own *m*-word message all others**
- **Reduction** ← reduce + scatter (if kp words are at each process, each process will have k words, different words)
  - **—each process gets a copy of the result**

every node is the destination of an all-to-one reduction



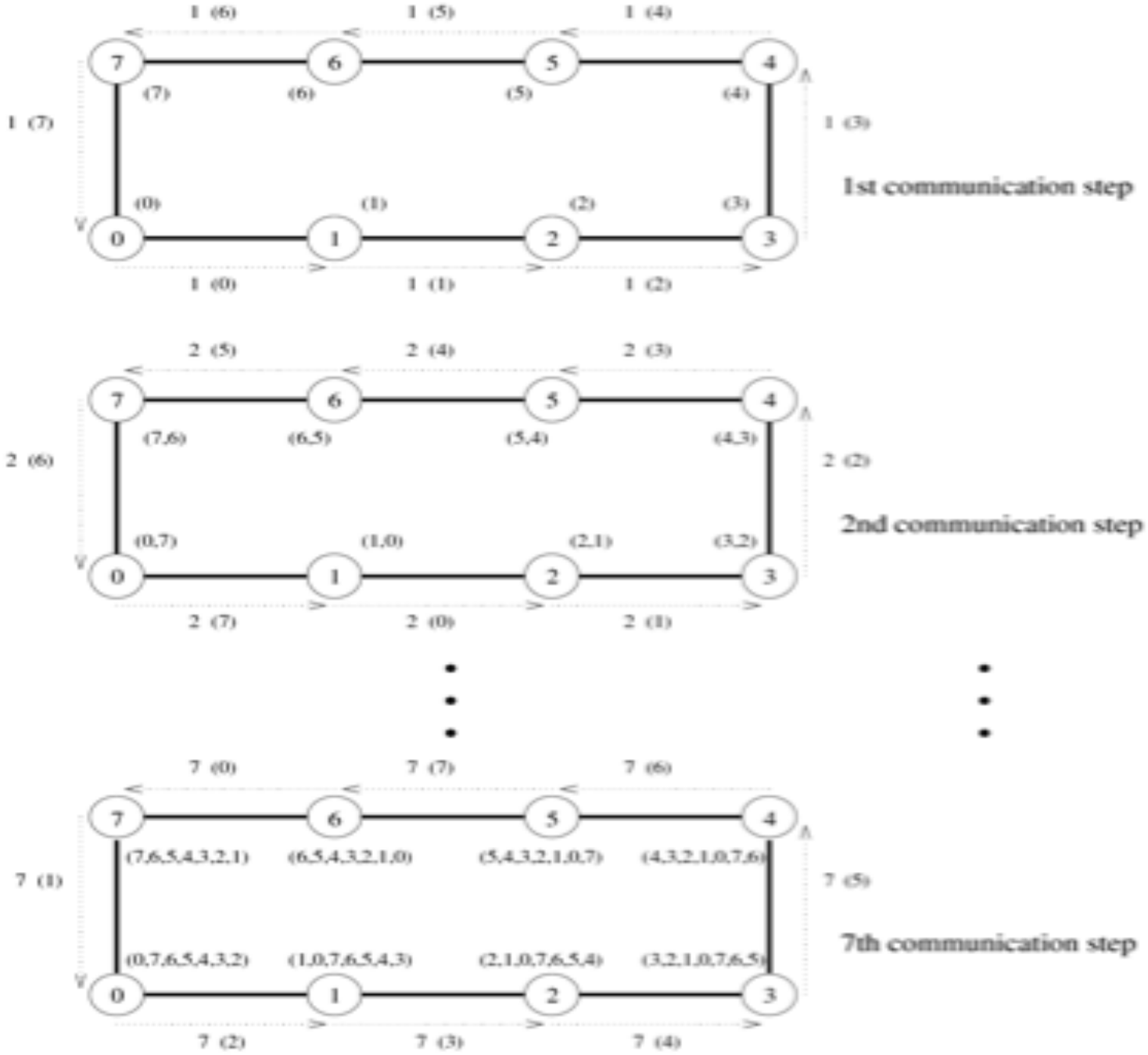


Figure 4.9 All-to-all broadcast on an eight-node ring. The label of each arrow shows the time step and, within parentheses, the label of the node that owned the current message being transferred before the beginning of the broadcast. The number(s) in parentheses next to each node are the labels of nodes from which data has been received prior to the current communication step. Only the first, second, and last communication steps are shown.

All-to-All Broadcast/Reduction on a Ring

```
1.    procedure ALL_TO_ALL_BC_RING(my_id, my_msg, p, result)
2.    begin
3.        left := (my_id − 1) mod p;
4.        right := (my_id + 1) mod p;
5.        result := my_msg;
6.        msg := result;
7.        for i := 1 to p − 1 do
8.            send msg to right;
9.            receive msg from left;
10.           result := result ∪ msg;
11.       endfor;
12.   end ALL_TO_ALL_BC_RING
```

Also works for linear array with bi-directional links

All-to-All Broadcast on a Ring

```
1.  procedure ALL_TO_ALL_RED_RING(my_id, my_msg, p, result)
2.  begin
3.     left := (my_id - 1) mod p;
4.     right := (my_id + 1) mod p;
5.     recv := 0;
6.     for i := 1 to p - 1 do
7.        j := (my_id + i) mod p;
8.        temp := msg[j] + recv;
9.        send temp to left;
10.        receive recv from right;
11.    endfor;
12.    result := msg[my_id] + recv;
13. end ALL_TO_ALL_RED_RING
```

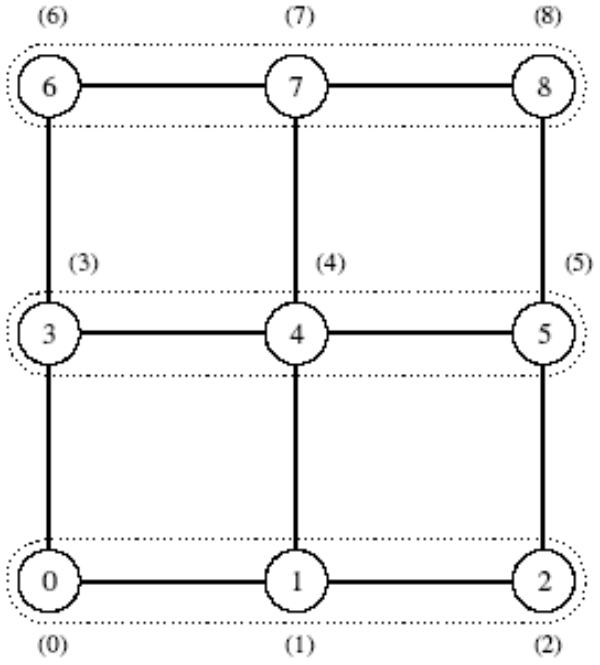
For an all-to-all reduction,

- combine (rather than append) each incoming message into your local result
- forward to your successor what you have received

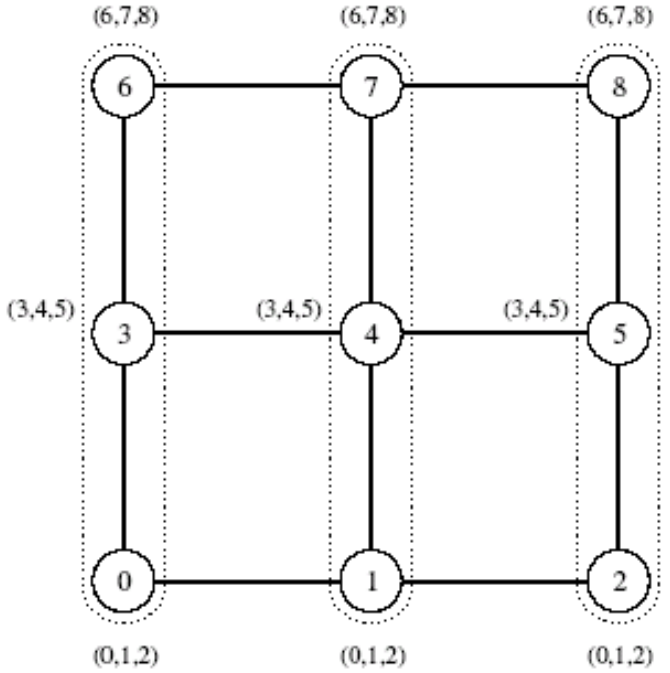
All-to-all Broadcast on a Mesh

Two phases

- **Each row performs all-to-all broadcast as for linear array/ring**
  - **—each node collects $\sqrt{p}$ messages for nodes in its own rows**
  - **—consolidates into a single message of size $m\sqrt{p}$.**
- **Perform column-wise all-to-all broadcast of merged messages**



(a) Initial data distribution

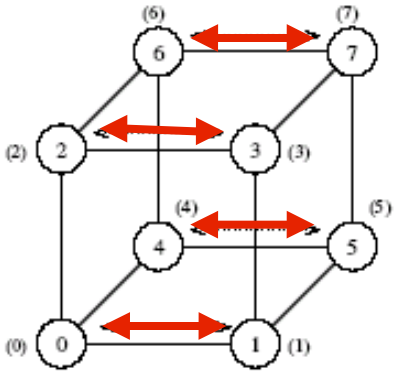


(b) Data distribution after rowwise broadcast

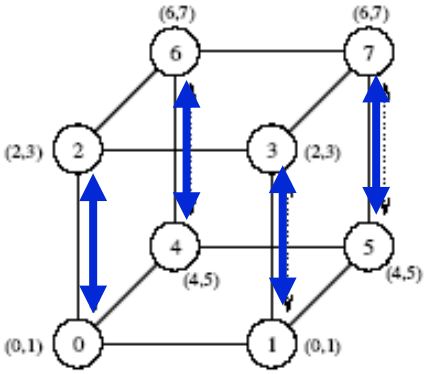
All-to-all Broadcast on a Hypercube

- **Generalization of the mesh algorithm to *log p* dimensions**
- **Message size doubles in each of *log p* steps**

1 value @ each



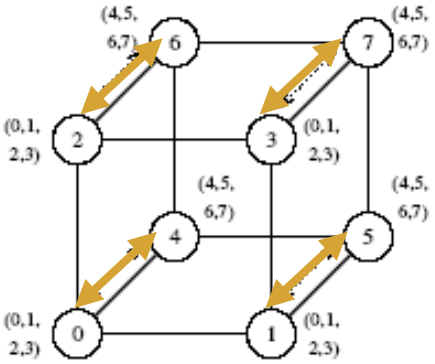
(a) Initial distribution of messages



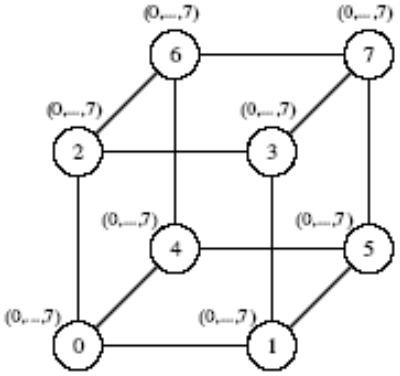
(b) Distribution before the second step

2 values @ each

4 values @ each



(c) Distribution before the third step



(d) Final distribution of messages

8 values @ each

All-to-all Broadcast on a Hypercube

```
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 − 1 do
5.            partner := my_id XOR 2^i;
6.            send result to partner;
7.            receive msg from partner;
8.            result := result ∪ msg;
9.        endfor;
10.   end ALL_TO_ALL_BC_HCUBE
```

All-to-all Reduction

- Similar to all-to-all broadcast, except for the merge
- Algorithm sketch

```
my_result = local_value

for each round
   send my_result to partner
   receive msg
   my_result = my_result ⊕ msg

post condition: each my_result now contains global result
```

Cost Analysis for All-to-All Broadcast

- **Ring**
  - *$(t_s + t_w m)(p-1)$*
- **Mesh**
  - *phase 1: $(t_s + t_w m)(\sqrt{p} - 1)$*
  - *phase 2: $(t_s + t_w m\sqrt{p})(\sqrt{p} - 1)$*
  - *total: $2t_s(\sqrt{p} - 1) + t_w m(p-1)$*
- **Hypercube**

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

$$= t_s \log p + t_w m(p-1).$$

Above algorithms are asymptotically optimal in msg size

Contention on Mapping

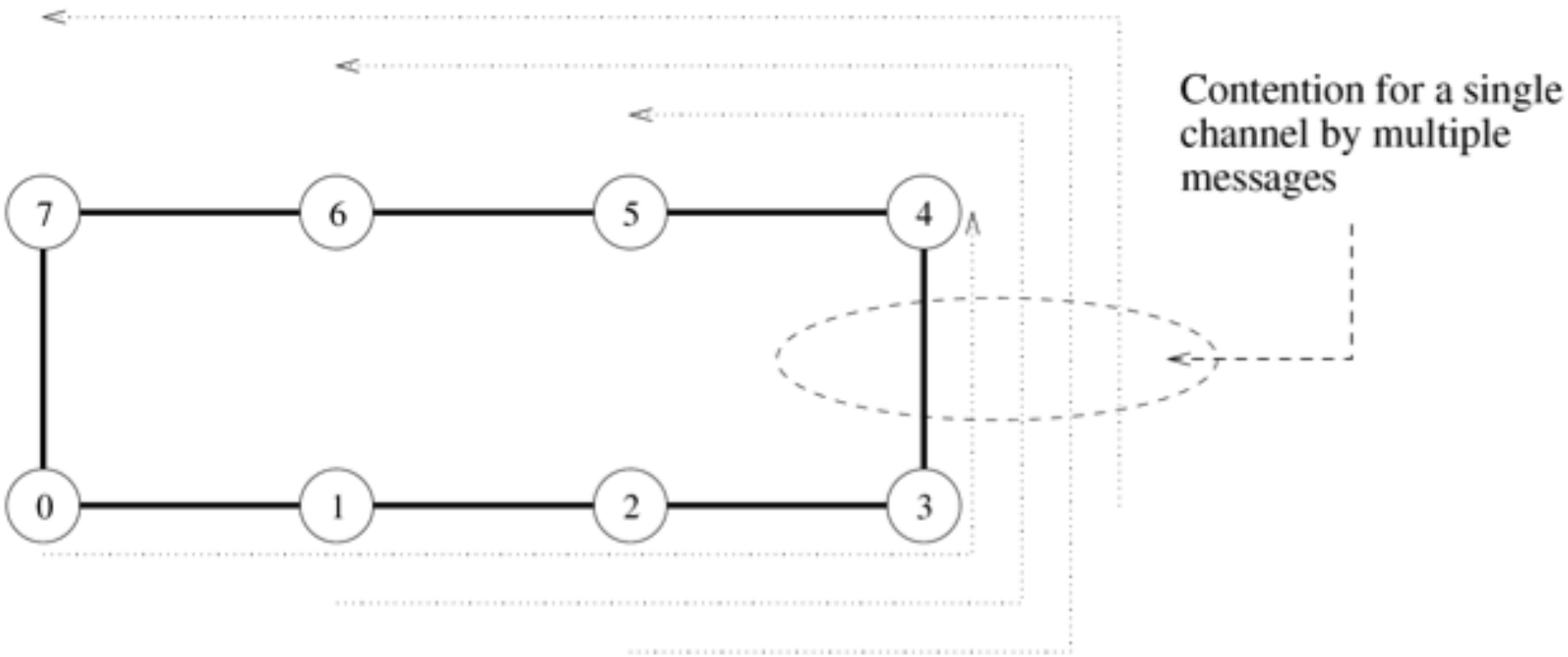
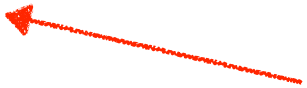


Figure 4.12 Contention for a channel when the communication step of Figure 4.11(c) for the hypercube is mapped onto a ring.

All-Reduce

- **Before: Each node has an element,**
  After: All the nodes have the reduction of these elements.



reduce + broadcast (if 10 numbers at each processor then the same 10 numbers will be at all processors)

- **Naive**
  - ***all to one reduction + one to all broadcast***
- **Strategy**
  - ***Similar to all to one reduction. Use all the links***

Prefix Sum

- Can implement using all-to-all reduction kernel
- Constraint
  - prefix sums on node *k*: values from *k*-node subset with labels ≤ *k*
- Strategy
  - implemented using an additional result buffer
  - add incoming value to result buffer on node *k*
    - only if the msg from a node ≤ *k*

Prefix Sum

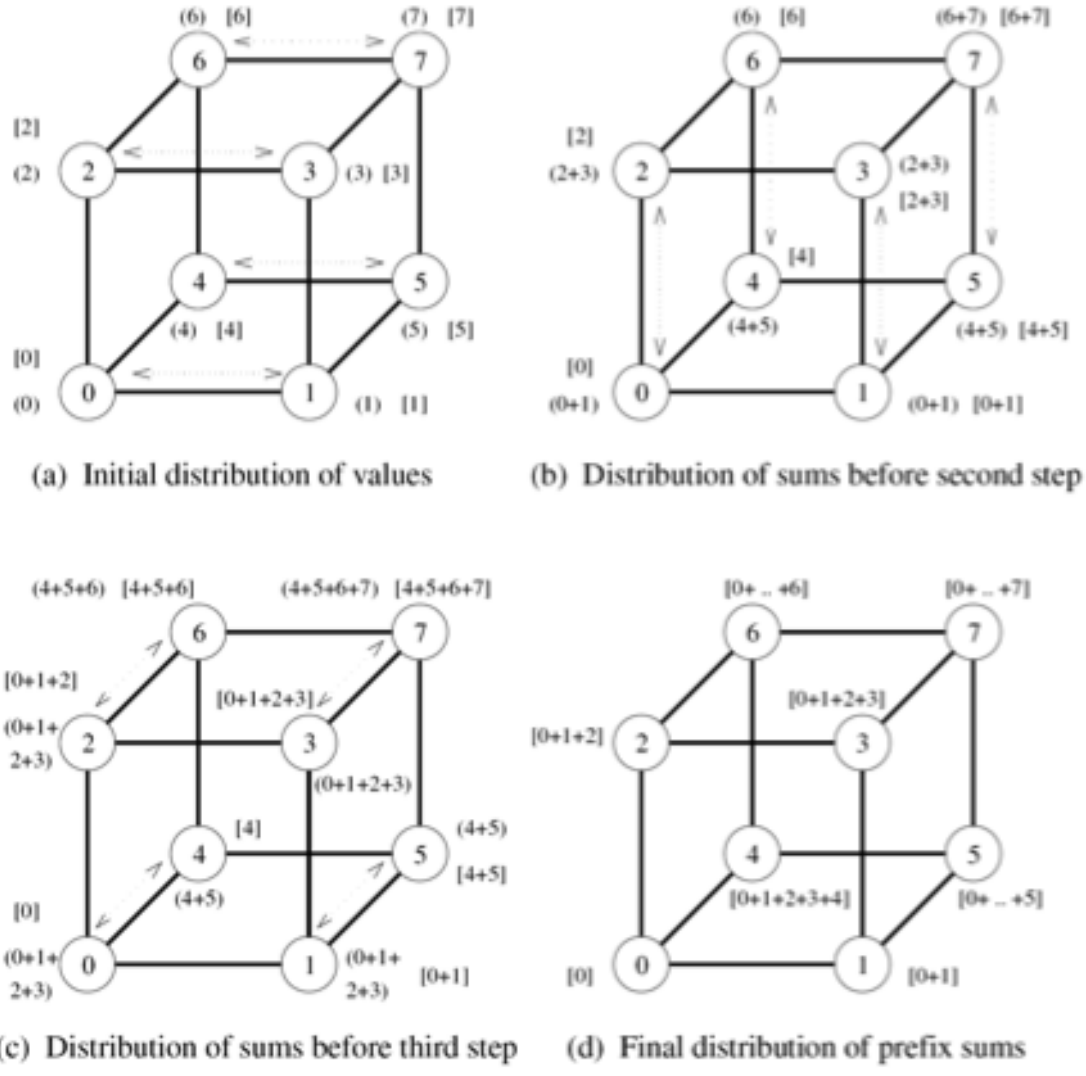


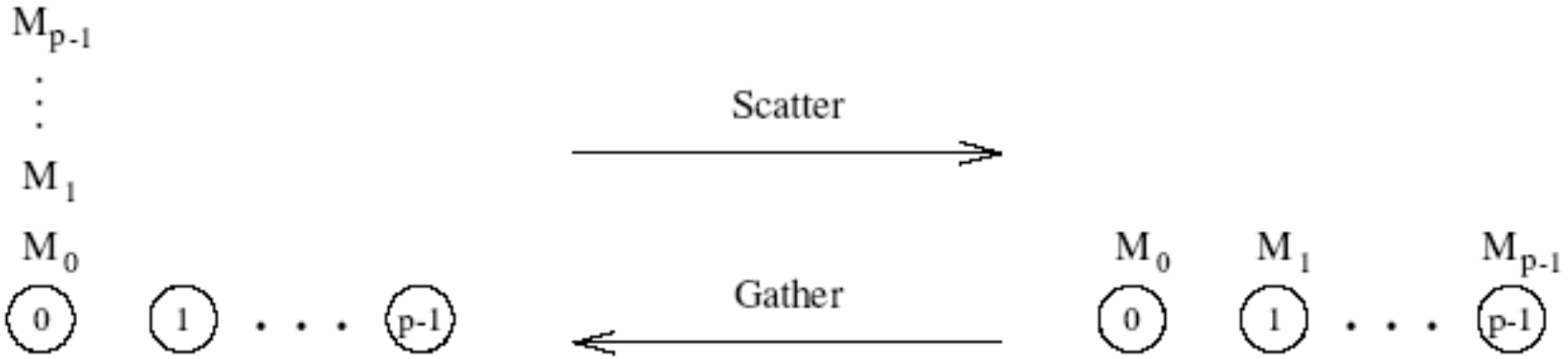
Figure 4.13 Computing prefix sums on an eight-node hypercube. At each node, square brackets show the local prefix sum accumulated in the result buffer and parentheses enclose the contents of the outgoing message buffer for the next step.

Prefix Sum on a Hypercube

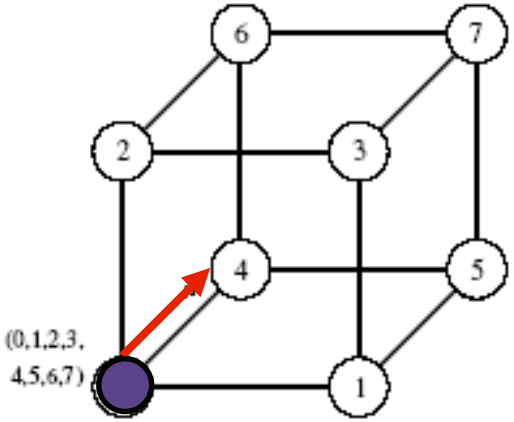
```
1.   procedure PREFIX_SUMS_HCUBE(my_id, my_number, d, result)
2.   begin
3.       result := my_number;
4.       msg := result;
5.       for i := 0 to d − 1 do
6.           partner := my_id XOR 2^i;
7.           send msg to partner;
8.           receive number from partner;
9.           msg := msg + number;
10.          if (partner < my_id) then result := result + number;
11.      endfor;
12.  end PREFIX_SUMS_HCUBE
```

Scatter and Gather

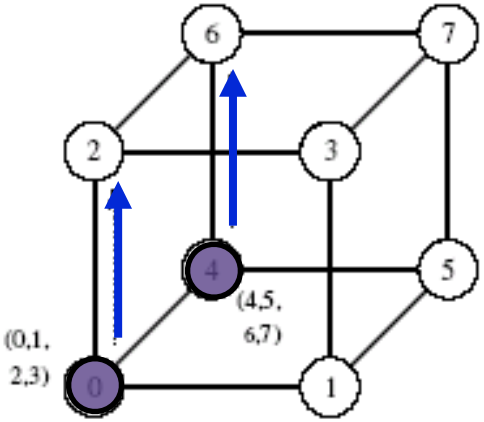
- **Scatter**
  - —a node sends a unique message of size *m* to every other node
    - AKA one-to-all personalized communication
  - —algorithmic structure is similar to broadcast
    - scatter: message size get smaller at each step
    - broadcast: message size stay constant
- **Gather**
  - —single node collects a unique message from each node
  - —inverse of the scatter operation; can be executed as such



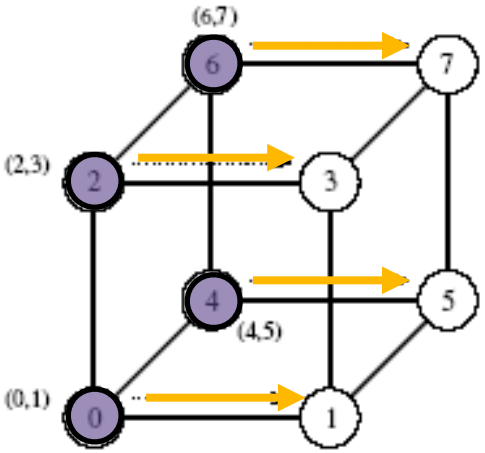
Scatter on a Hypercube



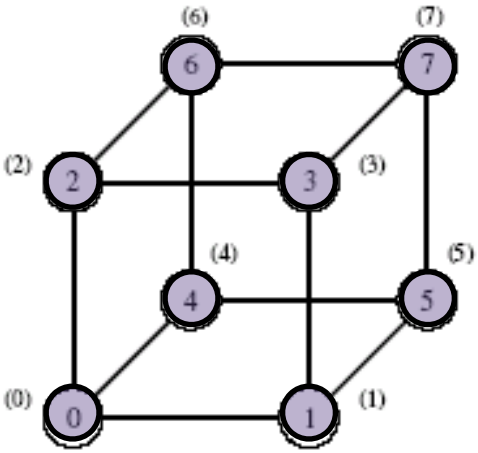
(a) Initial distribution of messages



(b) Distribution before the second step



(c) Distribution before the third step



(d) Final distribution of messages

Cost of Scatter and Gather

- Log *p* steps
  - —in each step
    - the message is sent to partner
    - message size halves
- Time

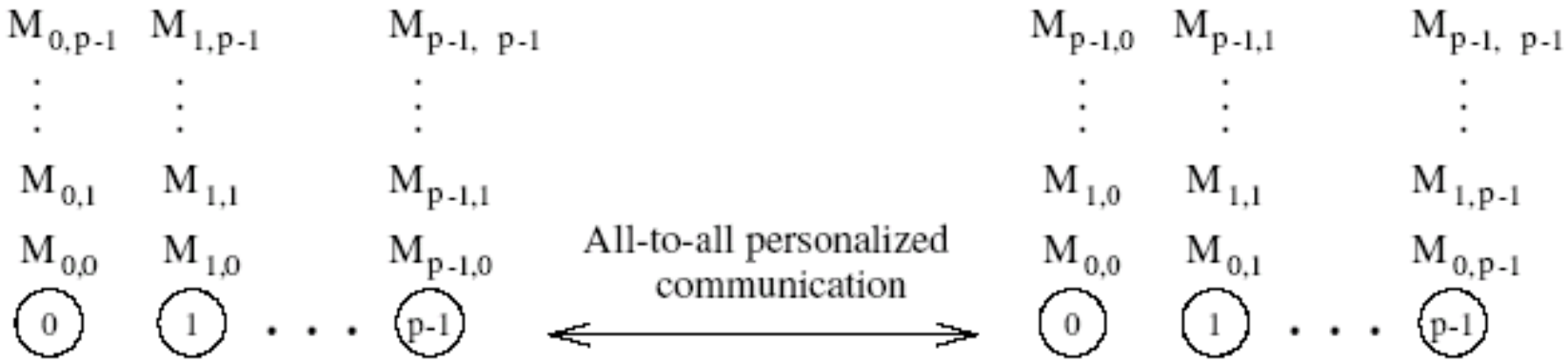
$$T = t_s \log p + t_w m(p-1). \qquad \text{(for all networks)}$$

- Note: time is asymptotically optimal in message size

All-to-All Personalized Communication

Total exchange

- **Each node: distinct message of size *m* for every other node**



All-to-All Personalized Communication

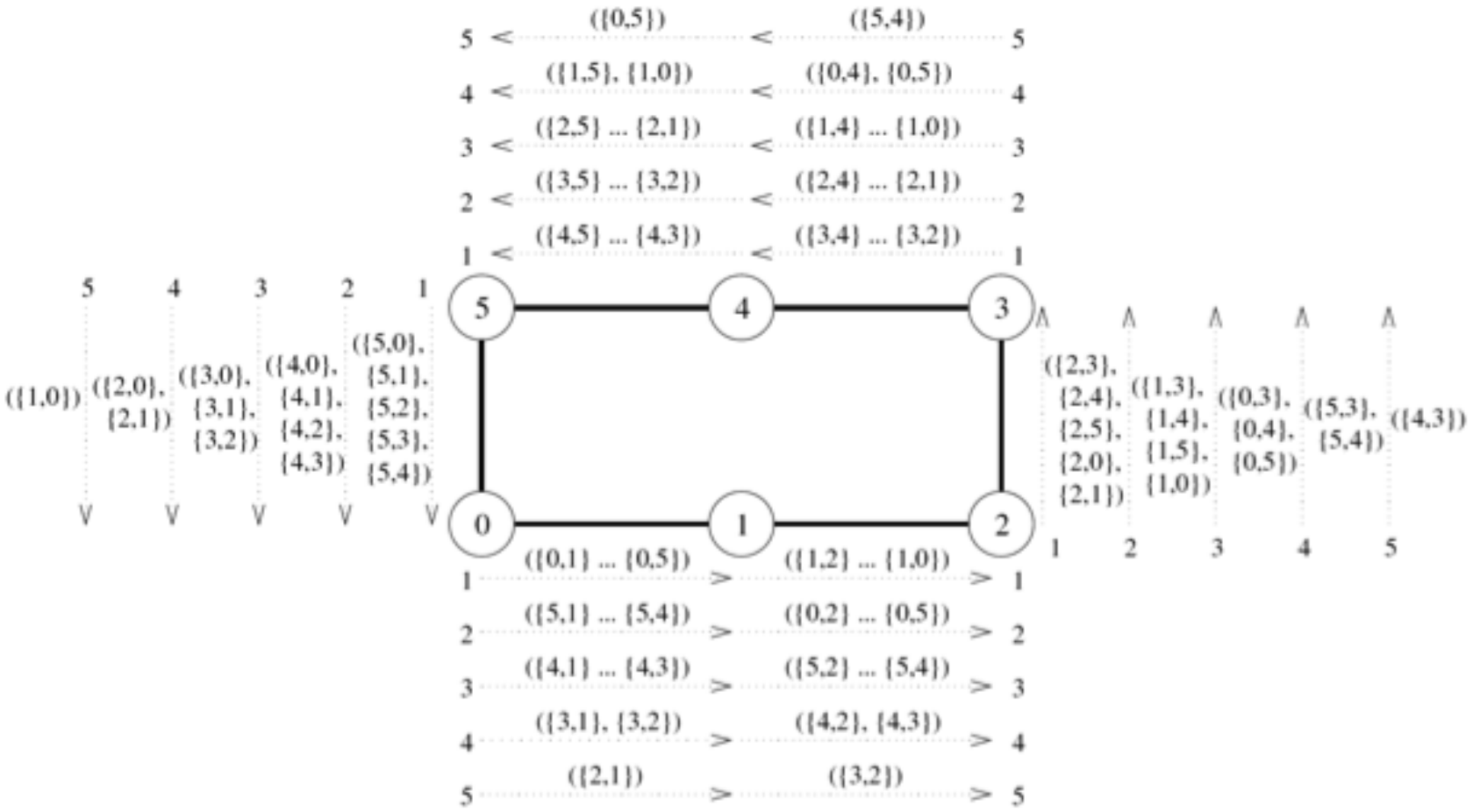


Figure 4.18 All-to-all personalized communication on a six-node ring. The label of each message is of the form $\{x, y\}$, where x is the label of the node that originally owned the message, and y is the label of the node that is the final destination of the message. The label $(\{x_1, y_1\}, \{x_2, y_2\}, \ldots, \{x_n, y_n\})$ indicates a message that is formed by concatenating n individual messages.

All-to-All Personalized Communication

- Every node has p pieces of data, each of size m
- Algorithm sketch for a ring

```
for k = 1 to p - 1
  send message of size m(p - k) to neighbor
  select piece of size m out of message for self
```

- Cost analysis

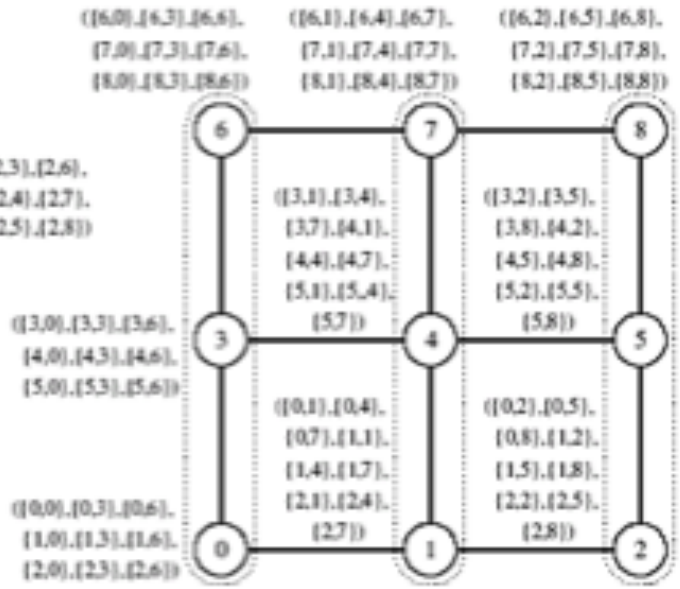
$$
\begin{aligned}
T &= \sum_{i=1}^{p-1} (t_s + t_w m(p-i)) \\
&= t_s(p-1) + t_w m \sum_{i=1}^{p-1} i \\
&= (t_s + t_w mp/2)(p-1)
\end{aligned}
$$

Optimality?

All-to-All Personalized Communication

- Mesh: apply the ring pattern twice

(a) Data distribution at the beginning of first phase

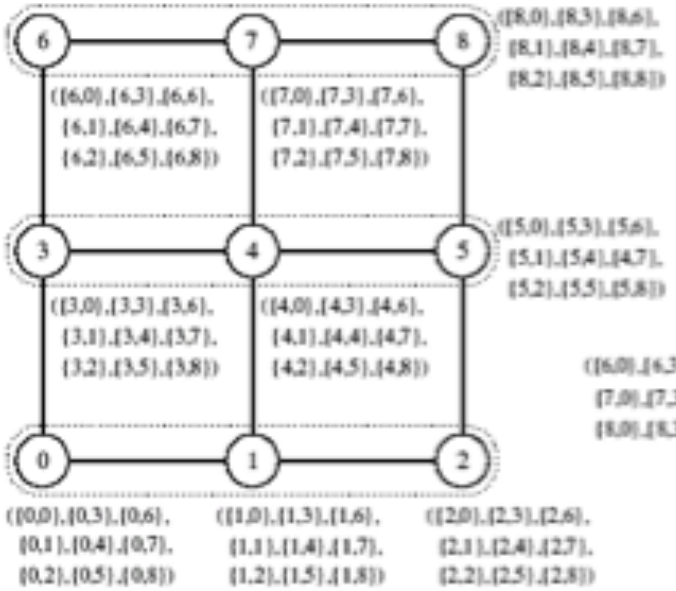


(b) Data distribution at the beginning of second phase

-Cost?

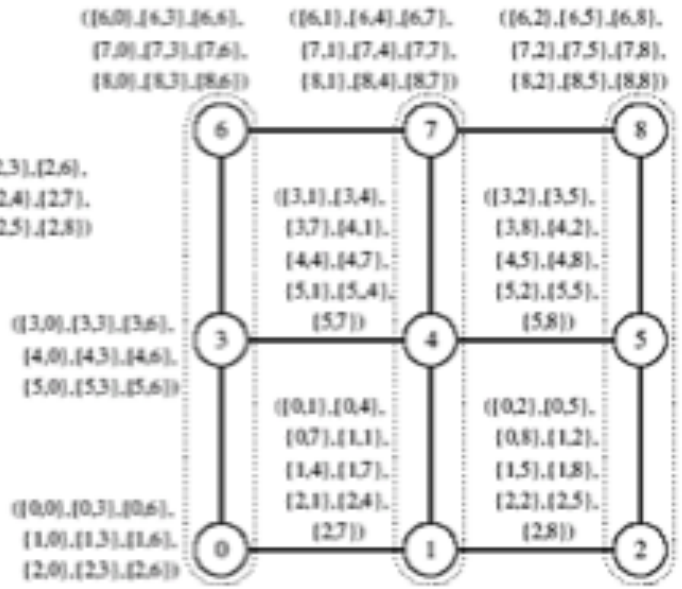
All-to-All Personalized Communication

- Mesh: apply the ring pattern twice



(a) Data distribution at the beginning of first phase

(2ts + twmp)(sqrt(p) - 1)

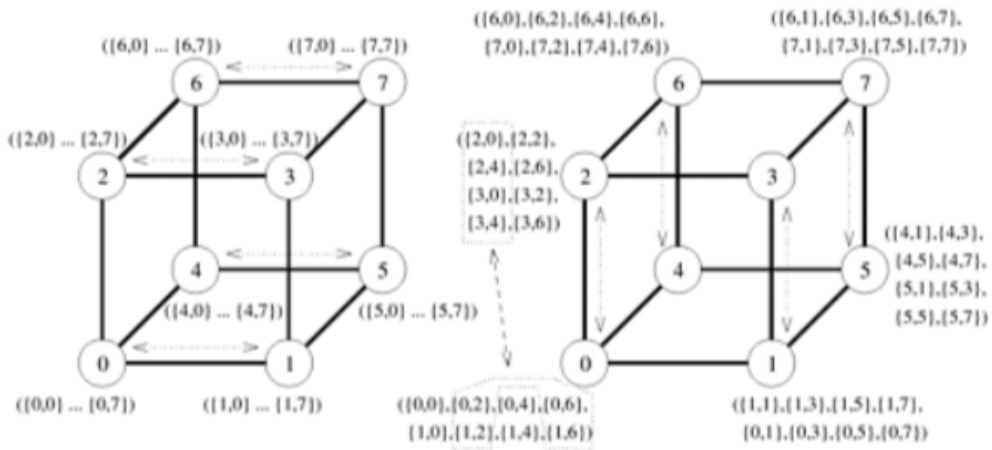


(b) Data distribution at the beginning of second phase

All-to-All Personalized Communication

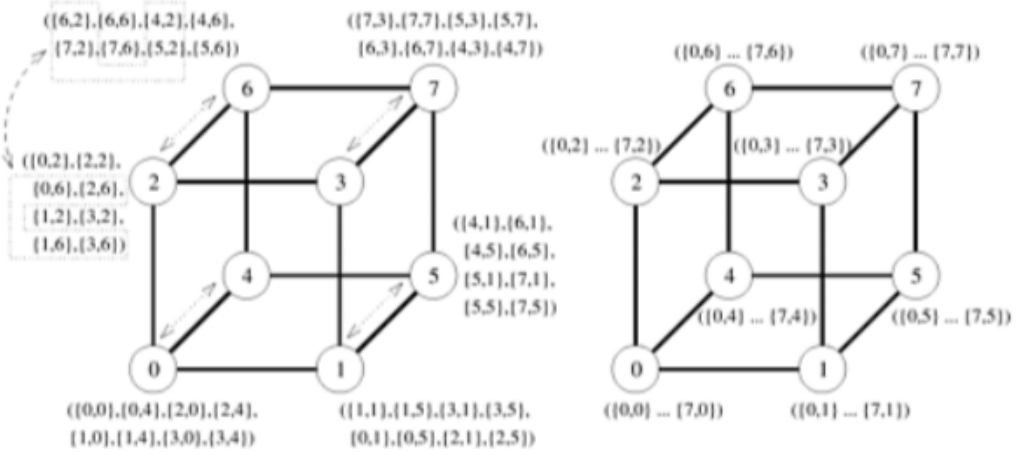
Hypercube:
Can we apply the mesh idea in multiple dimensions?

Cost?
Optimality?



(a) Initial distribution of messages

(b) Distribution before the second step



(c) Distribution before the third step

(d) Final distribution of messages

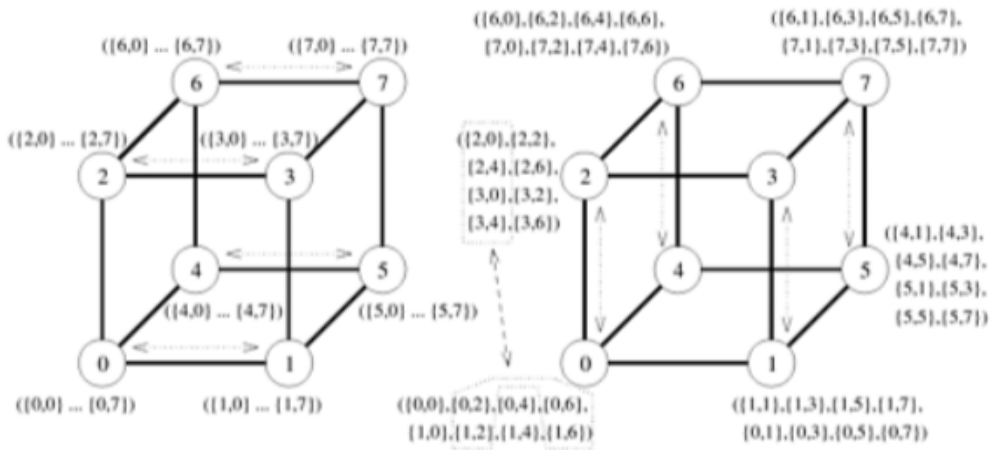
All-to-All Personalized Communication

Hypercube:
Can we apply the mesh idea in multiple dimensions?

Cost?
Optimality?

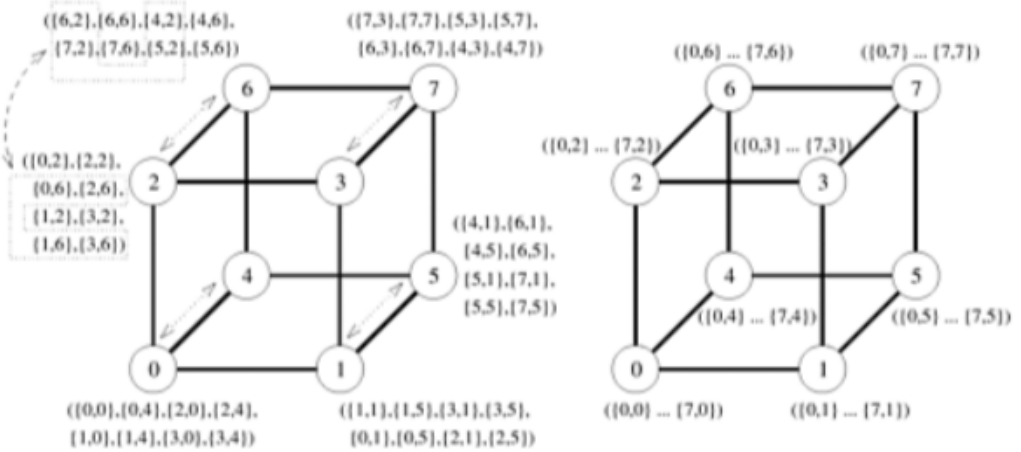
(ts + twmp/2)(log p)

not optimal



(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 Personalized Communication

Hypercube:
Another idea?

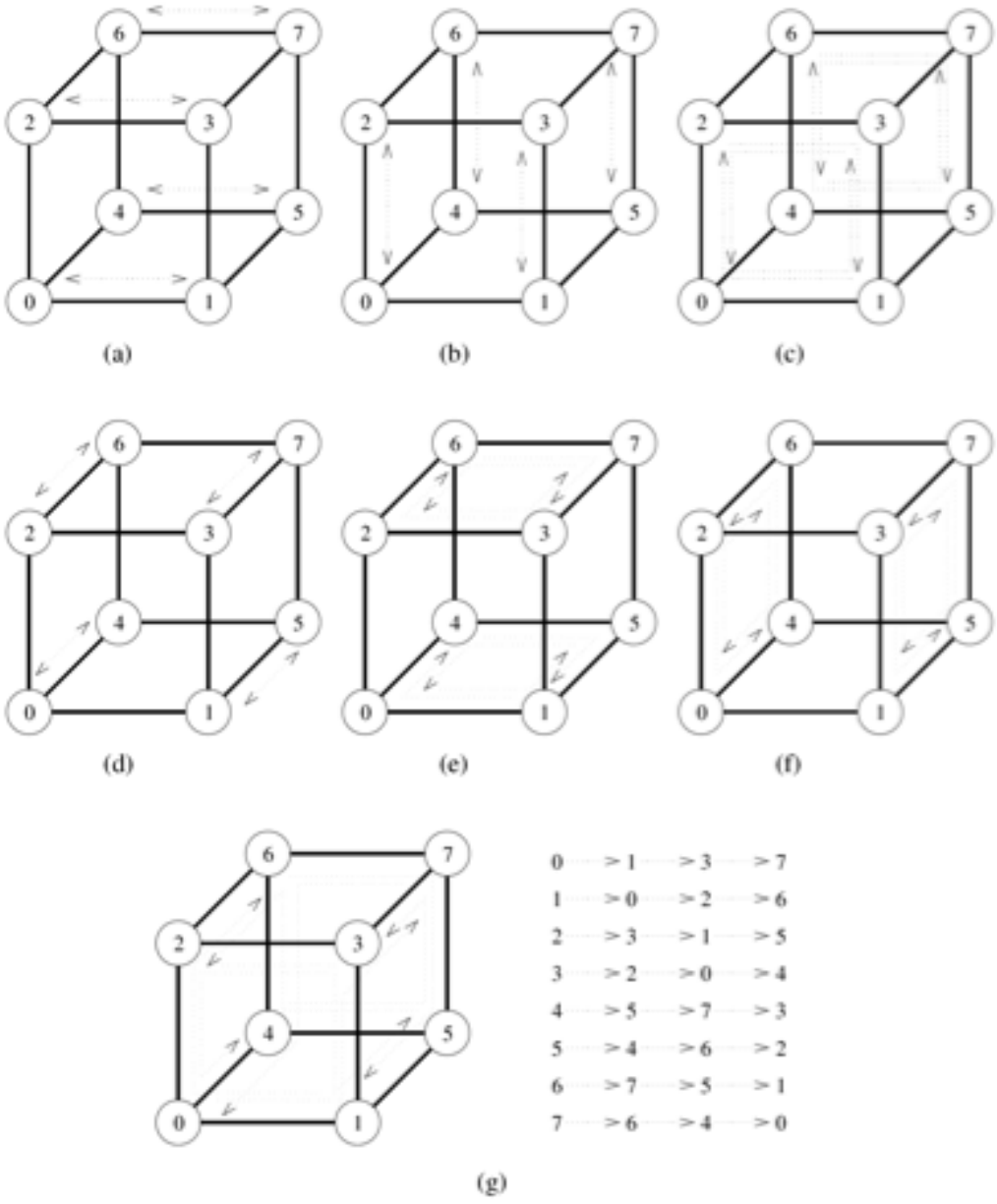


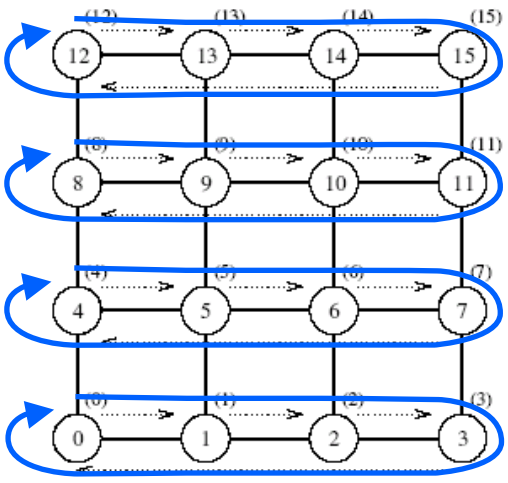
Figure 4.21 Seven steps in all-to-all personalized communication on an eight-node hypercube.

Circular Shift

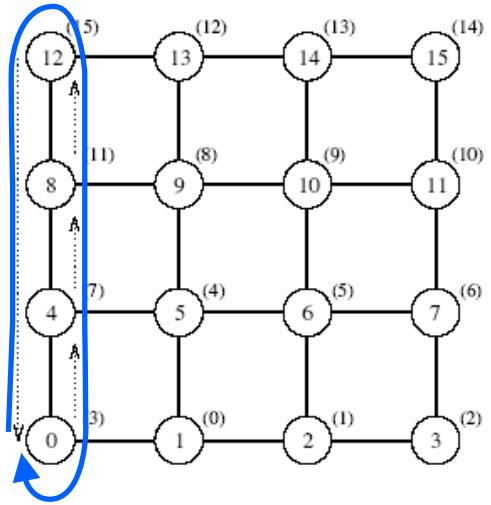
- **Permutation on p nodes**
  - — node *i* sends a data packet to node *(i + q)* mod *p, (0 ≤ q ≤ p)*
- **Ring**
  - —requires min(*q*,*p* – *q*) neighbor communications
- **Mesh**
  - —collectively shift along each direction in phases
  - —time upper bound

$$T = (t_s + t_w m)(\sqrt{p} + 1).$$

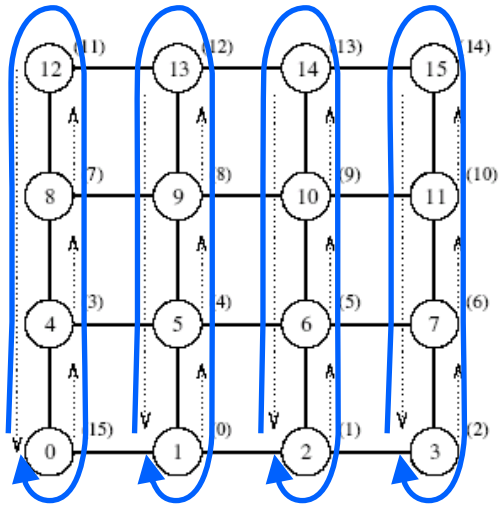
Circular Shift on a Ring Embedded in a Mesh



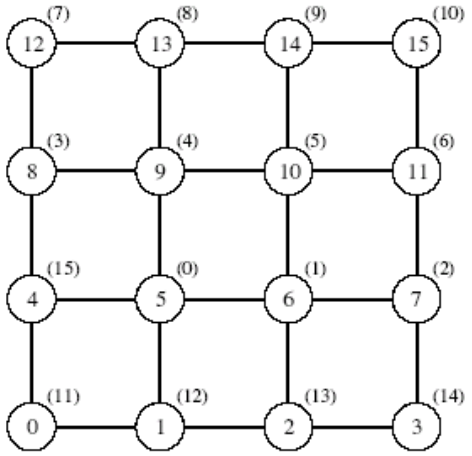
(a) Initial data distribution and the first communication step



(b) Step to compensate for backward row shifts



(c) Column shifts in the third communication step



(d) Final distribution of the data

Circular 5-shift on a *4 x 4* Mesh

Optimizing Collective Patterns

Example: one-to-all broadcast of large messages on a hypercube

- Consider broadcast of message *M* of size *m*, where *m* is large
- Cost of straightforward strategy $T = (t_s + t_w m)\log p$
- Optimized strategy
  - split *M* into *p* parts $M_0, M_1, \ldots M_p$ of size *m/p* each
    - want to place $M_0 \cup M_1 \cup \ldots \cup M_p$ on all nodes
  - scatter M_i to node *i*
  - have nodes collectively perform all-to-all broadcast
    - each node *k* broadcasts its M_k
- Cost analysis
  - scatter time = $t_s \log p + t_w(m/p)(p-1)$ *(slide 29)*
  - all-to-all broadcast time = $t_s \log p + t_w(m/p)(p-1)$ *(slide 22)*
  - total time = $2(t_s \log p + t_w(m/p)(p-1)) \approx 2(t_s \log p + t_w m)$ *(faster than slide 13)*

References

- Adapted from slides “Principles of Parallel Algorithm Design” by Ananth Grama
- Based on Chapter 4 of “Introduction to Parallel Computing” by Ananth Grama, Anshul Gupta, George Karypis, and Vipin Kumar. Addison Wesley, 2003
- Adapted from the slides of Prof. John Mellor-Crummey from Rice University.