Implementing Collective Communication

CS121 Parallel Computing

Spring 2020

Collective communication

- Important as the basis for many parallel algorithms.
- Implemented using multiple point to point communications, possibly in parallel.
  - Assume communicating m words (without contention) between a source and destination takes $t_s + m\ t_w$ time.
    - Ignore the the distance of the message, since per hop latency t_h is usually small.
  - With contention c, time becomes $t_s + c\ m\ t_w$.
- Implementations depend on communication hardware architecture.
- Operations
  - Broadcast and reduction.
  - All-to-all broadcast and reduction.
  - All-reduce, prefix sum.
  - Scatter and gather.
  - All-to-all scatter and reduce.
  - Circular shift.

Broadcast and reduction on ring

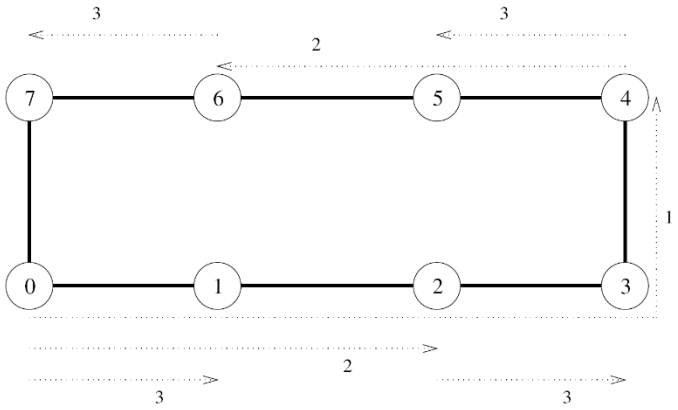


Figure 4.2 One-to-all broadcast on an eight-node ring. Node 0 is the source of the broadcast. Each message transfer step is shown by a numbered, dotted arrow from the source of the message to its destination. The number on an arrow indicates the time step during which the message is transferred.

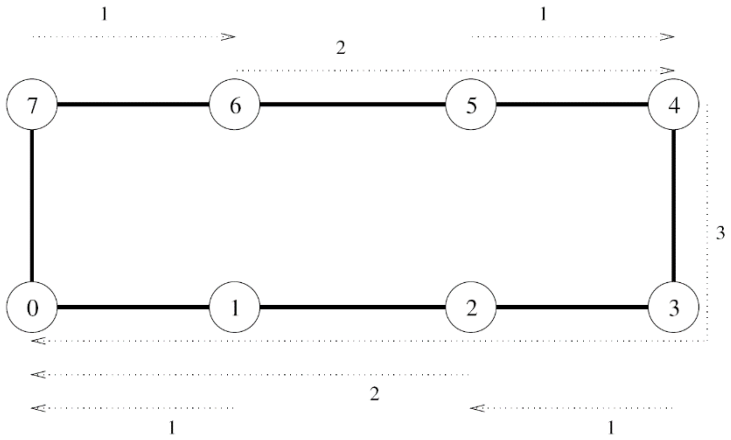


Figure 4.3 Reduction on an eight-node ring with node 0 as the destination of the reduction.

- Start with root sending message to distance p/2 node.
- In later stages all nodes with the message send it distance d/2, where d is distance of during last stage.
- Reduction has reverse communication pattern.

Cost

- With p processors, log p steps.
- In each step, a processor sends a size m message.
- Total time $(t_s + m\ t_w) \log p$.

Source: Introduction to Parallel Computing, Grama et al

Broadcast on mesh

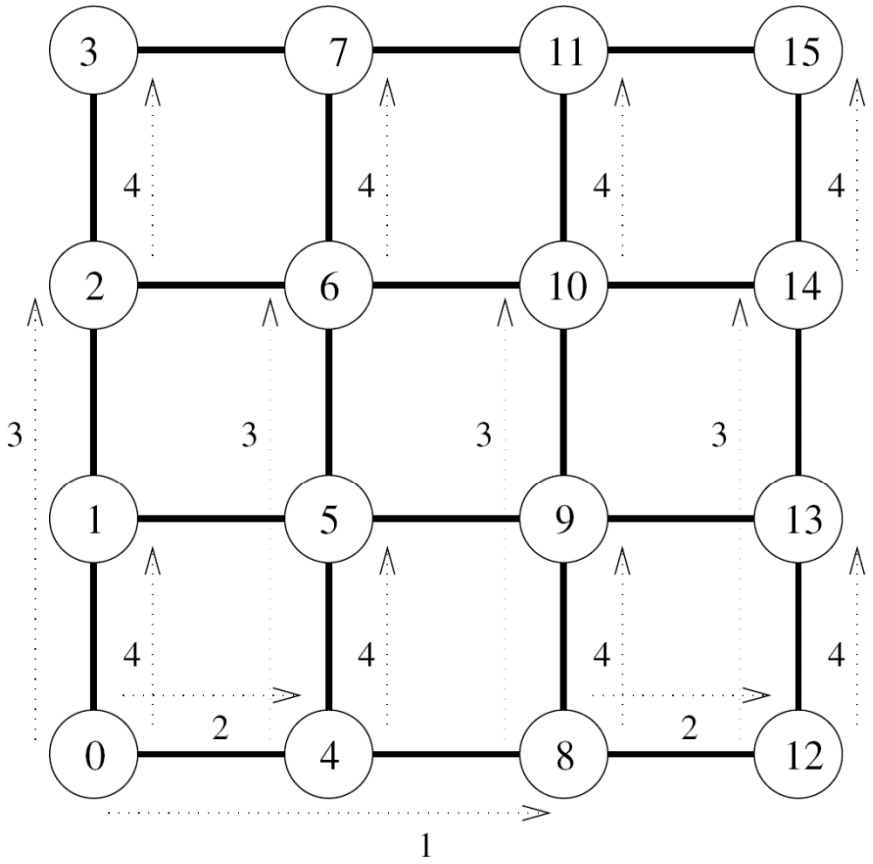


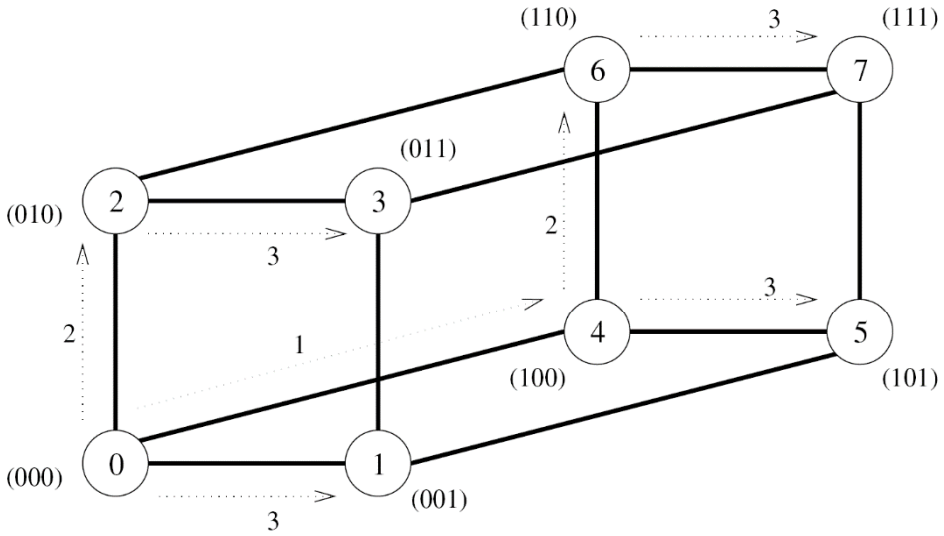
Figure 4.5 One-to-all broadcast on a 16-node mesh.

- Root first broadcasts along its row using ring algorithm.
- The nodes of root's row then broadcast along their columns using ring algorithm.

Cost

- log $\sqrt{p}$ = (log p) / 2 steps along row, same along columns.
- Total time $(t_s + m\, t_w) \log p$.

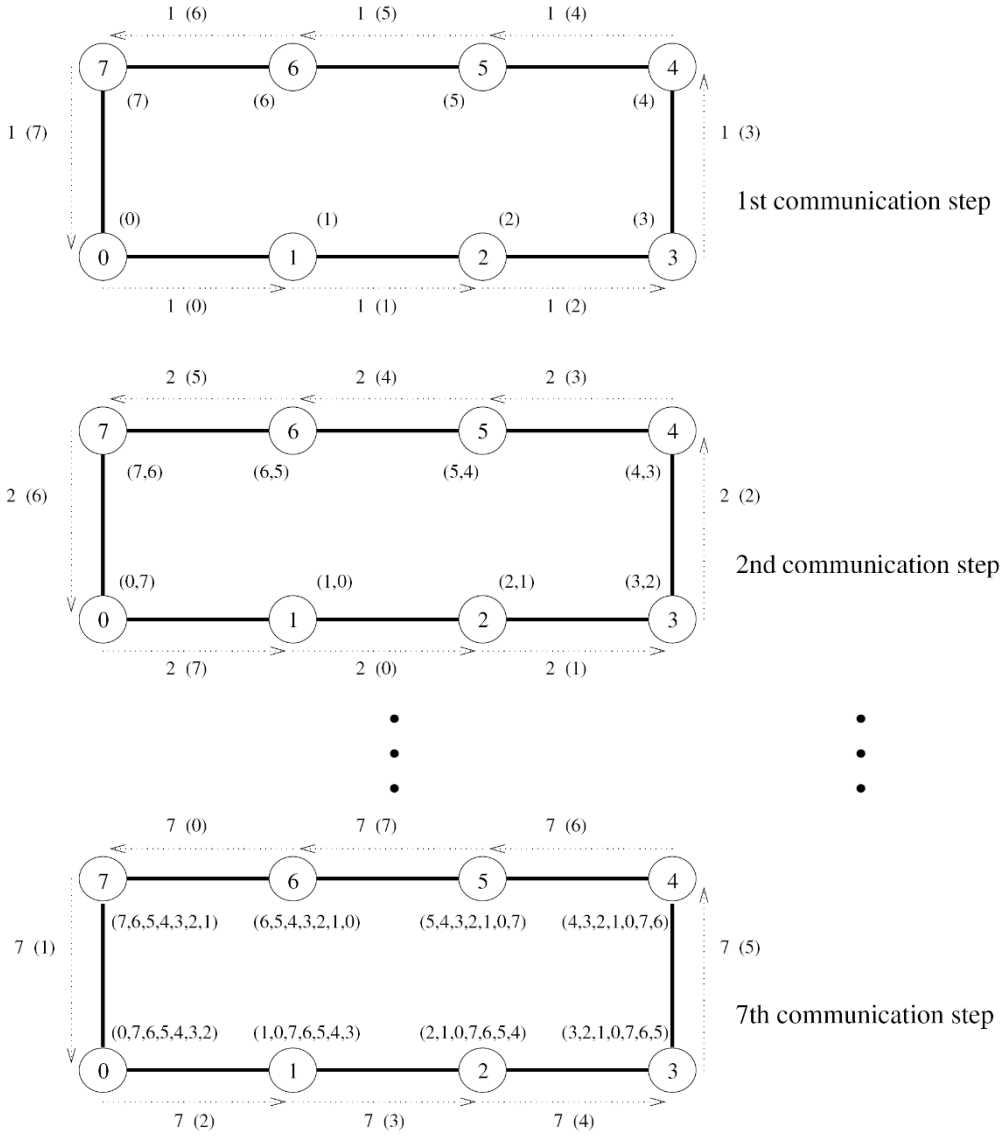
Broadcast on hypercube



- All nodes with message send.
- Send in parallel along the dimensions in order.
- Again, log p steps, so total time $(t_s + m\ t_w) \log p$.

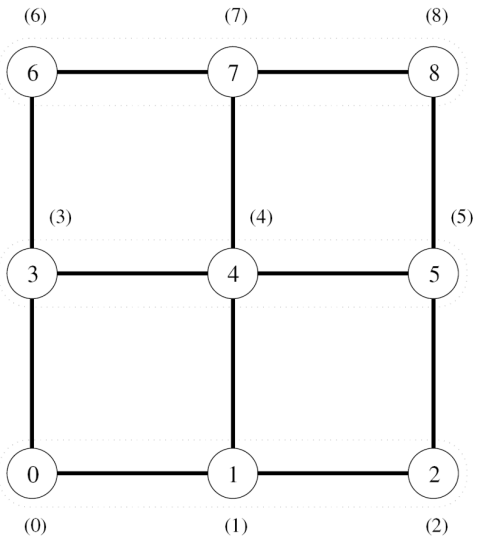
Figure 4.6 One-to-all broadcast on a three-dimensional hypercube. The binary representations of node labels are shown in parentheses.

All-to-all broadcast on ring

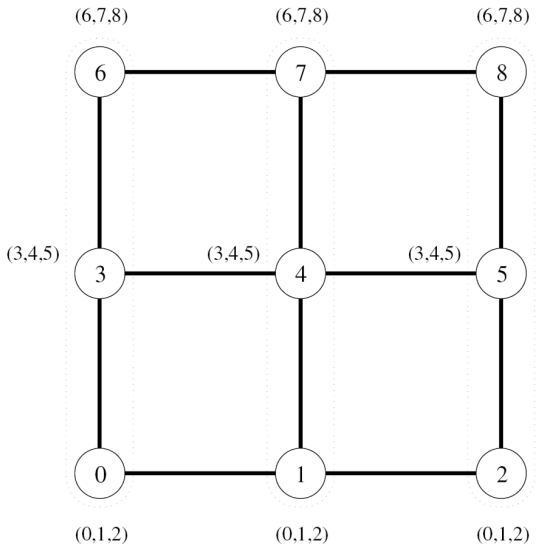


- For a size p ring, run in p-1 steps.
- Keep passing data down the ring. Processes store any new data they receive.
- Numbers in parentheses show the data a process has received by a certain time.
- Cost is $(t_s + m\, t_w)\,(p-1)$.

All-to-all broadcast on torus



(a) Initial data distribution



(b) Data distribution after rowwise broadcast

- First do all-to-all broadcast along each row using ring algorithm.
- After this, each process has a $\sqrt{p}$ pieces of data.
- Then do all-to-all broadcast along each column using ring algorithm.
  - Each node sends data with size $\sqrt{p}$.

Cost

- First stage has $\sqrt{p}-1$ steps and $(t_s + m\, t_w)(\sqrt{p} -1)$ cost.
- Second stage has $\sqrt{p}-1$ steps and $(t_s + m\sqrt{p}\, t_w)(\sqrt{p} -1)$ cost because each message has $m\sqrt{p}$ size.
- Total cost $2\, t_s (\sqrt{p} -1) + m\, t_w (p -1)$

All-to-all broadcast and reduce on hypercube

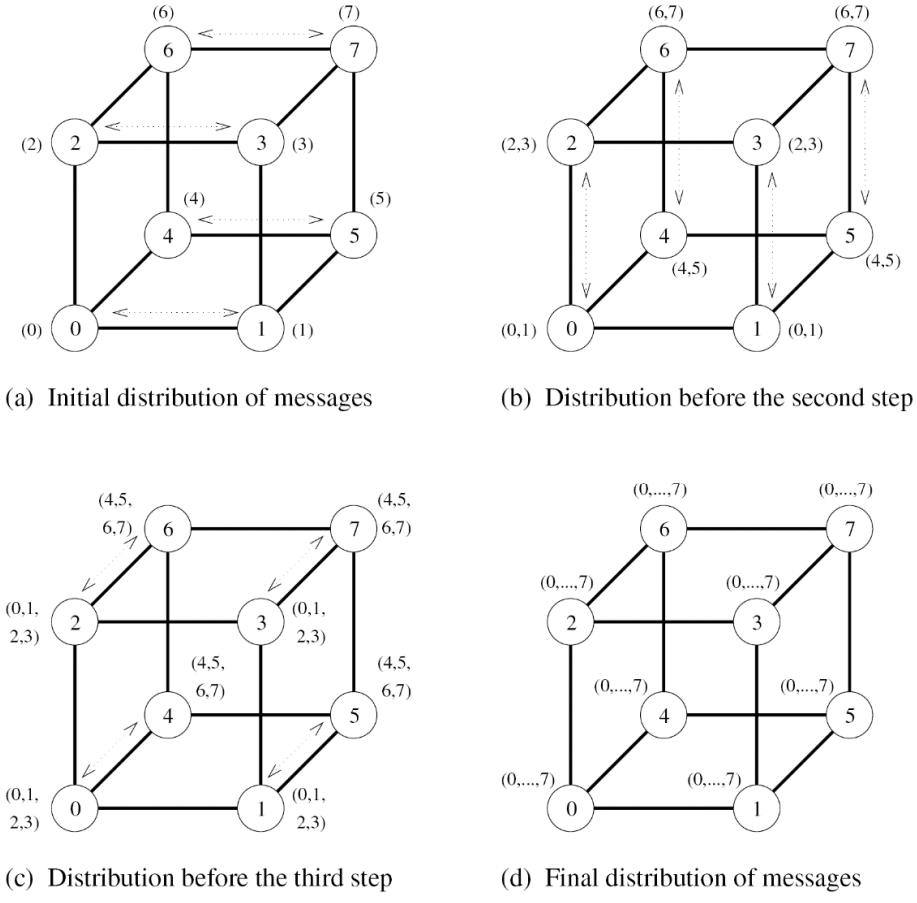


Figure 4.11 All-to-all broadcast on an eight-node hypercube.

- In each step, each process sends all the data it has received along a dimension.
- log p steps total.
- In step i, send messages of size 2^i.
- Total cost

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

- All-to-all reduce has same communication pattern, except processes add all the data they receive.
- So each message only has size m, and total cost is (t_s + m t_w) log p.

Prefix sum on hypercube

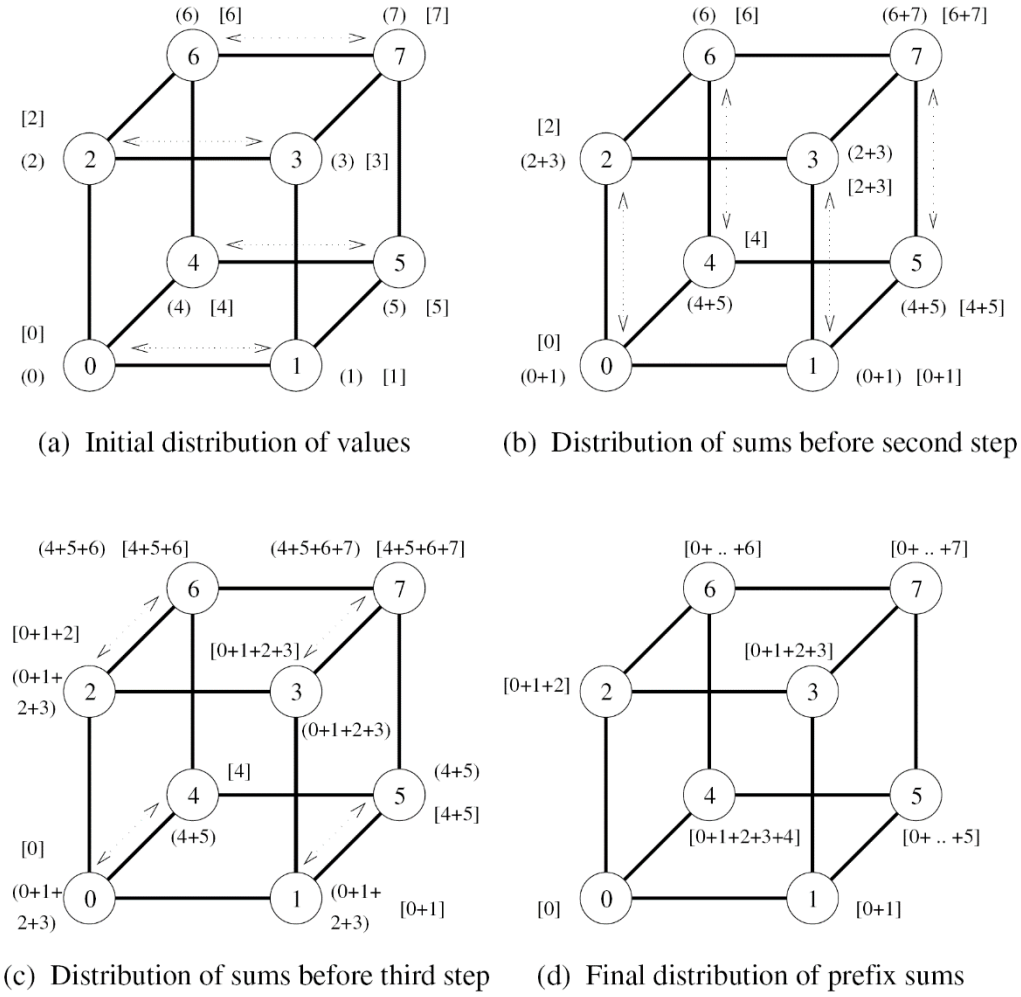


Figure 4.13 Computing prefix sums on an eight-node hypercube. At each node, square brackets show the local prefix sum accumulated in the result buffer and parentheses enclose the contents of the outgoing message buffer for the next step.

- Each process keeps two values, its prefix sum p (shown in brackets), and the sum s of all the values it's received (shown in parentheses).
  - There are some mistakes in the s values, e.g. in figure (b) the (6) value should be (6+7), and in figure (c), (4+5+6) should be (4+5+6+7)
- In each step, each process
  - Sends its s value along a new dimension, in order of least to most significant dimension.
  - Adds s it receives into its own s.
  - If it received s from a lower ordered process, it adds the new s into its current p.
- This works correctly because by the time a process is ready to receive data along a dimension, all values of processes in lower dimensions have been summed in s and will be sent.
- There are log p steps, and the total cost is $(t_s + m\, t_w) \log p$.

Scatter and gather

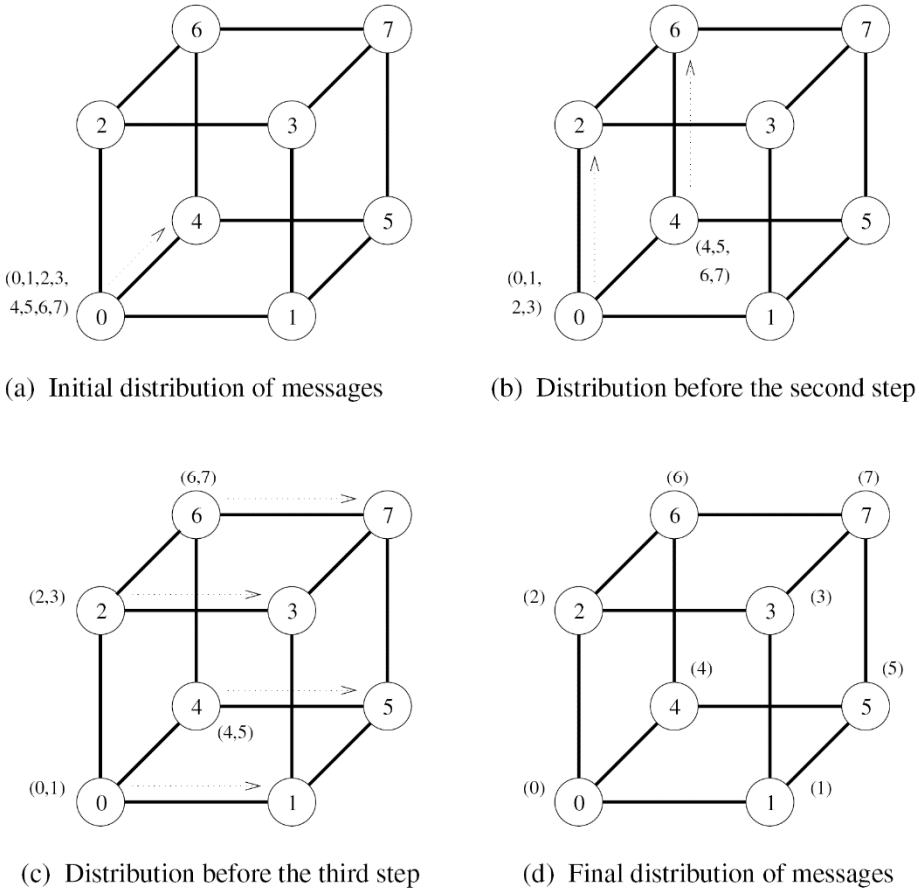


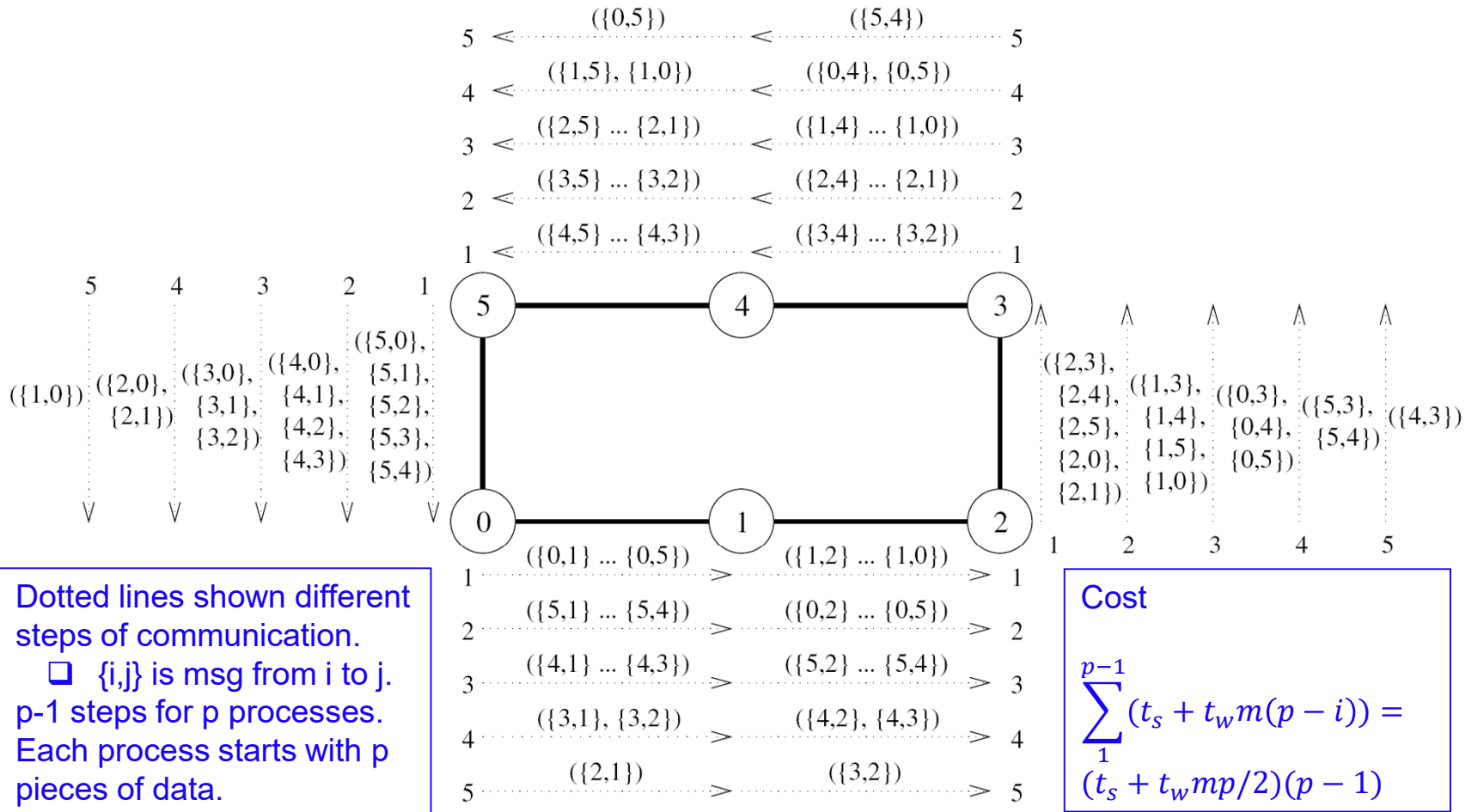
Figure 4.15 The scatter operation on an eight-node hypercube.

- Initially root has p pieces of data.
- In each step, a process sends half the data it's received along a new dimension.
- Total cost

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

- Gather reverses the communication, and has the same cost.

All-to-all personalized on ring



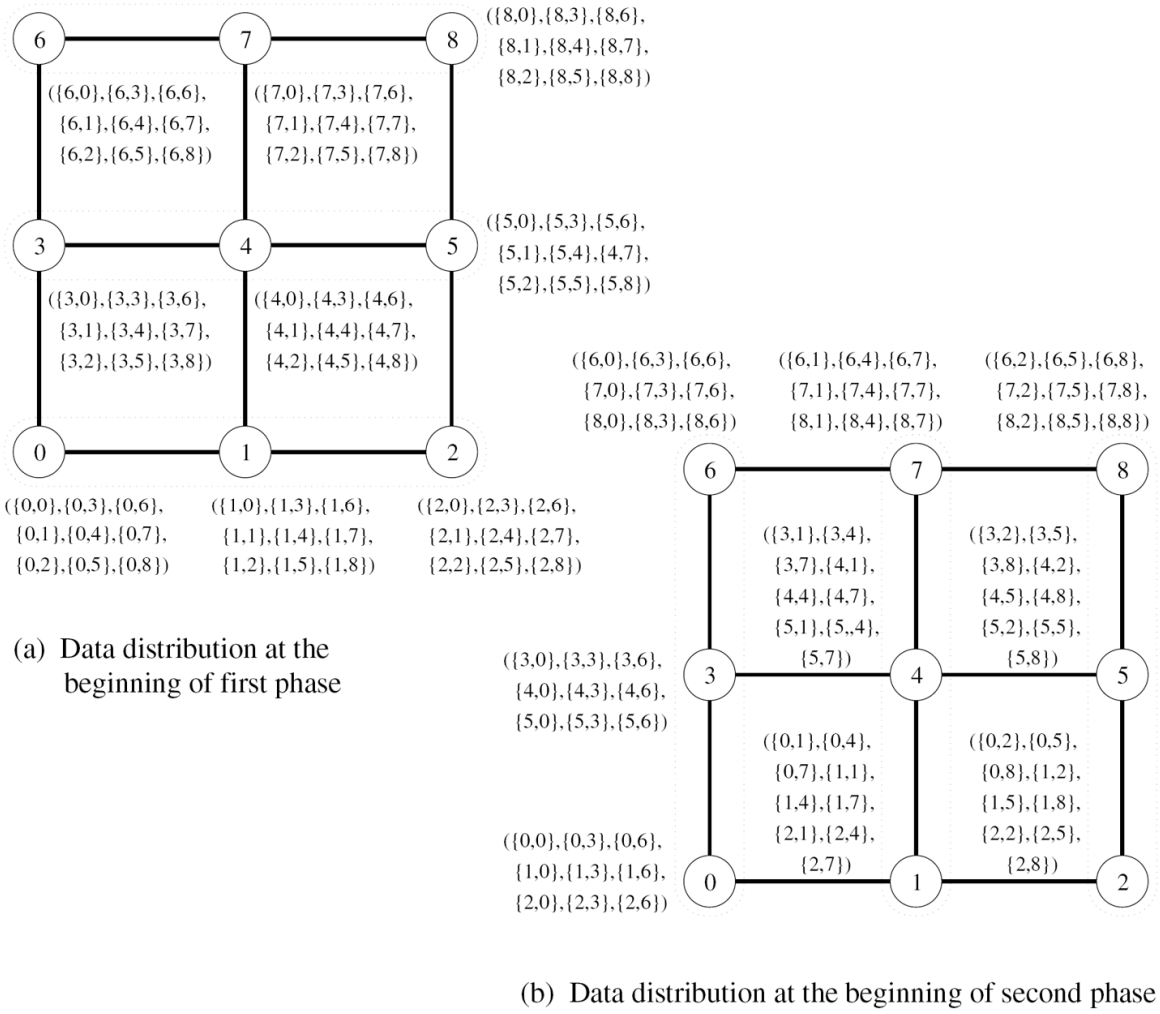
- Dotted lines shown different steps of communication.
  - {i,j} is msg from i to j.
- p-1 steps for p processes.
- Each process starts with p pieces of data.
- Each time it receives data, it keeps the piece intended for it, and passes on the rest.

Cost

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

$$(t_s + t_w mp/2)(p-1)$$

All-to-all personalized on torus



- Each row first breaks its data into √p blocks, each block containing all its messages for nodes within a different column.
- Then it does all-to-all personalized with the blocks to its row.
- Next, do all to all personalized along each column.
- Each phase involves all-to-all personalized sending of data of mp size to $\sqrt{p}$ processors.
- Using previous equation, total cost is $(2\ t_s + mpt_w)(\sqrt{p} - 1)$.

All-all personalized on hypercube

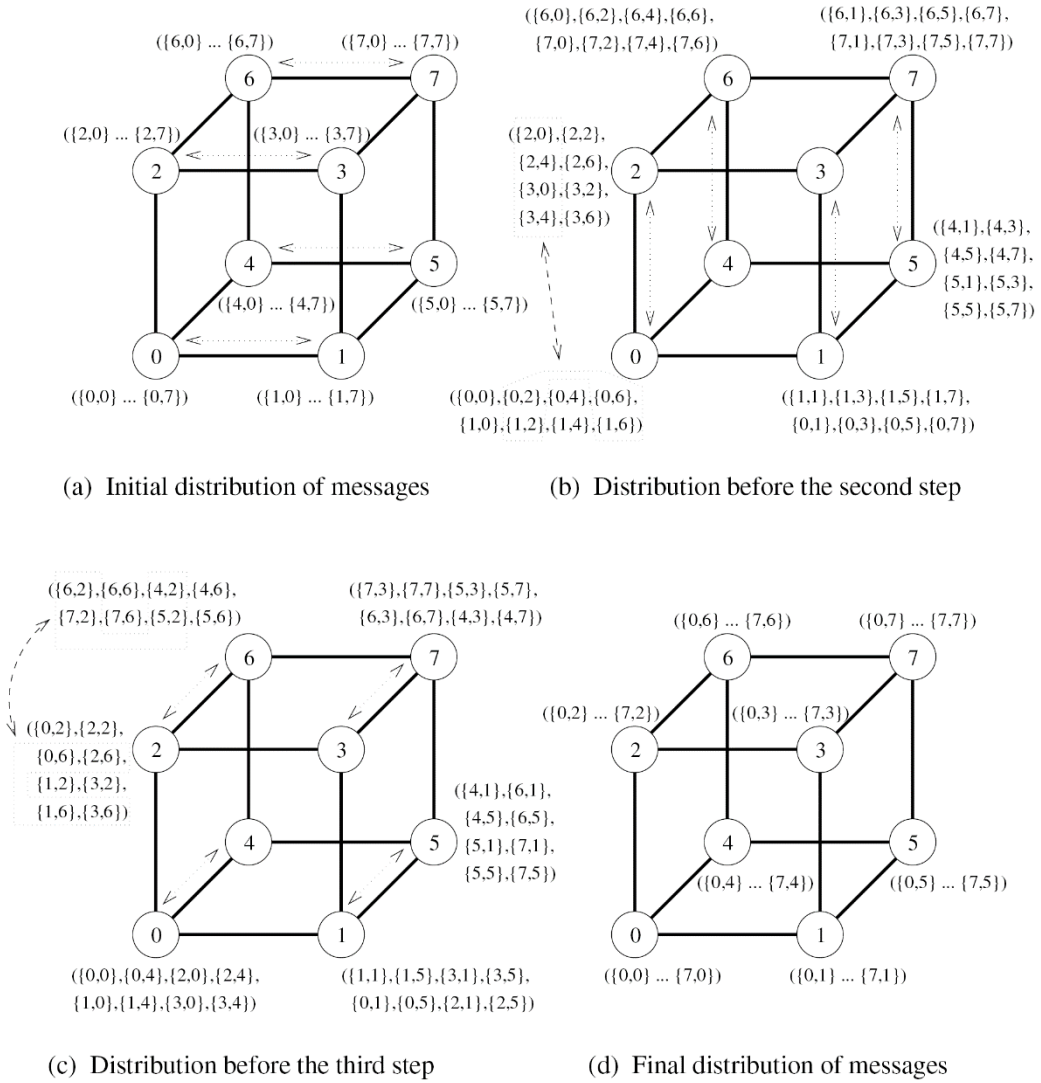


Figure 4.20 An all-to-all personalized communication algorithm on a three-dimensional hypercube.

- Send in order of dimension from least to most significant.
  - For dimension i, send all data with destination differing in i'th least significant bit.
  - Each step confines broadcast problem to one less dimension.
- Total of log p steps, and send p/2 amount of data in each step.
- Total time $(t_s + m\ p\ t_w / 2) \log p$.

- Each process receives m (p-1) pieces of data.
- Average distance between processes (log p)/2.
- Total communication volume for p processes is mp(p-1)(log p) / 2.
- Number of links in hypercube is p log p / 2.
- So lower bound on communication is volume divided by number of links, i.e. $t_w\ m\ (p-1)$.
- This algorithm isn't optimal.

All-all personalized on hypercube

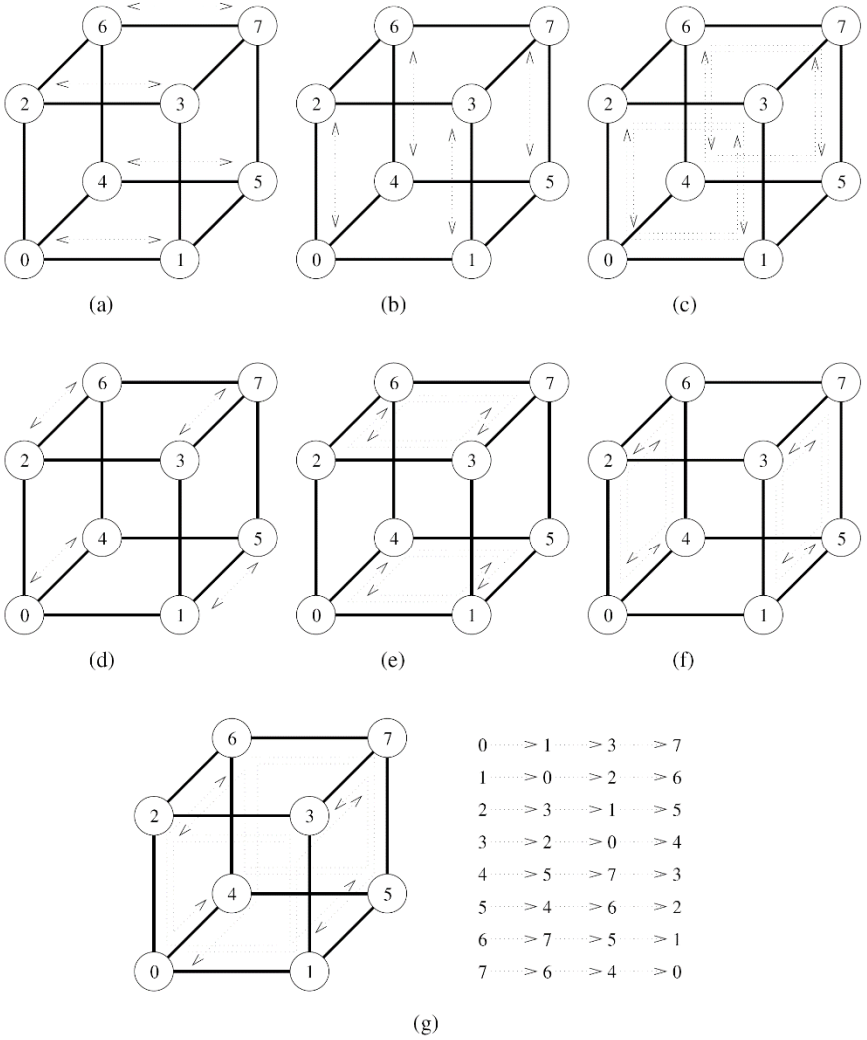
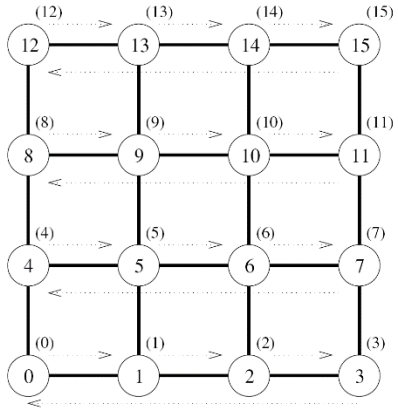


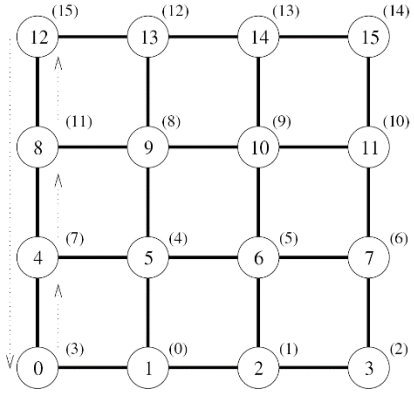
Figure 4.21 Seven steps in all-to-all personalized communication on an eight-node hypercube.

- Another method for all-to-all personalized communication is for each process to send one message to each other process.
- Run for p-1 steps. In step j, process i sends data to process (i XOR j).
  - Message routed using E-cube routing (i.e. send in order of increasing dimension.)
  - A hypercube is so well connected that this communication pattern can be routed with no congestion, assuming bidirectional links.
- So each message sent in $t_s + m\,t_w$ time, and total time is $(t_s + m\,t_w)\,(p-1)$, which is optimal.

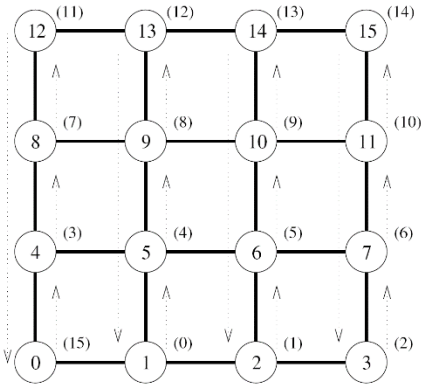
Circular shift on mesh



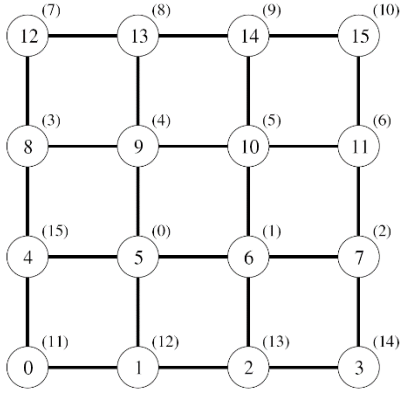
(a) Initial data distribution and the first communication step



(b) Step to compensate for backward row shifts



(c) Column shifts in the third communication step

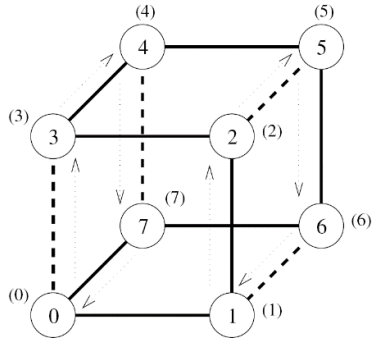


(d) Final distribution of the data

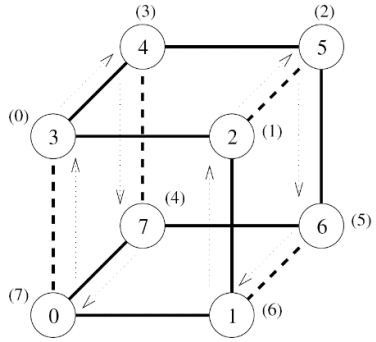
Figure 4.22 The communication steps in a circular 5-shift on a 4 × 4 mesh.

- Given a shift of k, process p sends to (p+k) mod p.
- Do k mod √p shifts along x dimension, ⌊k / √p⌋ shifts along y dimension.
- First, shift along the x dimension.
- For each x shift, need to do a compensatory shift in the first column.
  - E.g. in figure (b), 3's message should be at node 4, not 0.
- Then shift along y dimension.
- The movement in either dimension is at most √p.
- Each shift costs $t_s + m\ t_w$, so total cost $2\ (t_s + m\ t_w)\ \sqrt{p}$.

Circular shift on hypercube

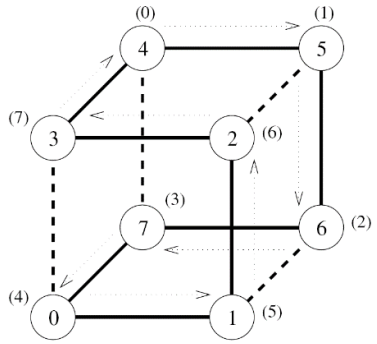


First communication step of the 4-shift

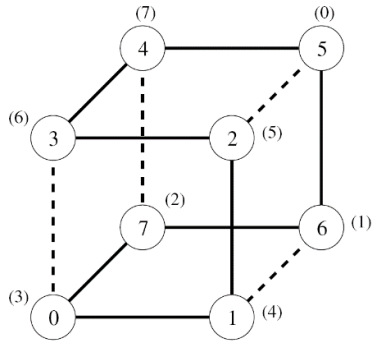


Second communication step of the 4-shift

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



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



(c) Final data distribution after the 5-shift

Figure 4.23 The mapping of an eight-node linear array onto a three-dimensional hypercube to perform a circular 5-shift as a combination of a 4-shift and a 1-shift.

- To shift in hypercube, first map the ring into the hypercube using the Gray code construction.
  - For any i > 0, any two nodes differing by 2^i in ring are mapped to nodes distance 2 apart in the hypercube.
  - Ex Nodes 2 and 6 get mapped to $(011)_2$ and $(101)_2$, resp.
  - Neighbors in the ring are mapped to neighbors in the hypercube.
- To shift by k, write k in binary, then shift along dimensions with 1 digits.
  - Ex To shift by $5=(101)_2$, each node shifts along dimension 0, then 2.
  - For each dimension $i \neq 0$, shifting by 2^i takes 2 steps.
  - Shifting along dimension 0 takes 1 step.
- Shift along at most log p dimensions, so cost is $2\ (t_s + m\ t_w) \log p$.