Introduction to Parallel Computing

George Karypis

Basic Communication Operations

Outline

- Importance of Collective Communication Operations
- One-to-All Broadcast
- All-to-One Reduction
- All-to-All Broadcast & Reduction
- All-Reduce & Prefix-Sum
- Scatter and Gather
- All-to-All Personalized

Collective Communication Operations

- They represent regular communication patterns that are performed by parallel algorithms.
  - Collective: Involve groups of processors
- Used extensively in most data-parallel algorithms.
- The parallel efficiency of these algorithms depends on efficient implementation of these operations.
- They are equally applicable to distributed and shared address space architectures
- Most parallel libraries provide functions to perform them
- They are extremely useful for “getting started” in parallel processing!

MPI Names

Table 4.2 MPI names of the various operations discussed in this chapter.

Operation	MPI Name
One-to-all broadcast	MPI_Bcast
All-to-one reduction	MPI_Reduce
All-to-all broadcast	MPI_Allgather
All-to-all reduction	MPI_Reduce_scatter
All-reduce	MPI_Allreduce
Gather	MPI_Gather
Scatter	MPI_Scatter
All-to-all personalized	MPI_Alltoall

One-to-All Broadcast & All-to-One Reduction

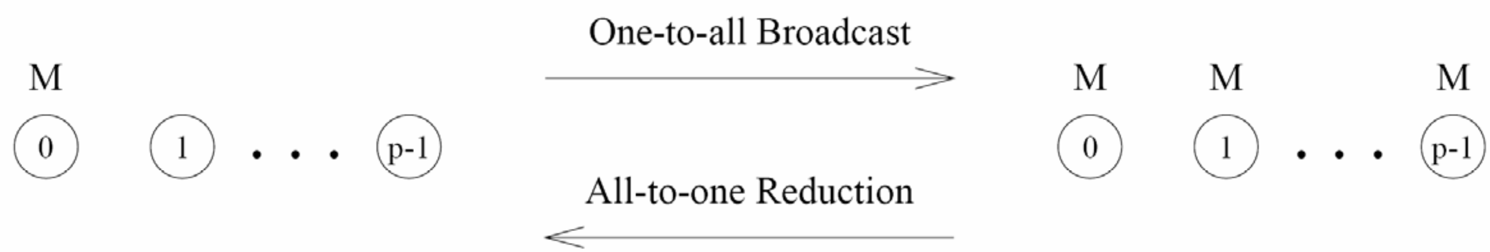


Figure 4.1 One-to-all broadcast and all-to-one reduction.

Broadcast on a Ring Algorithm

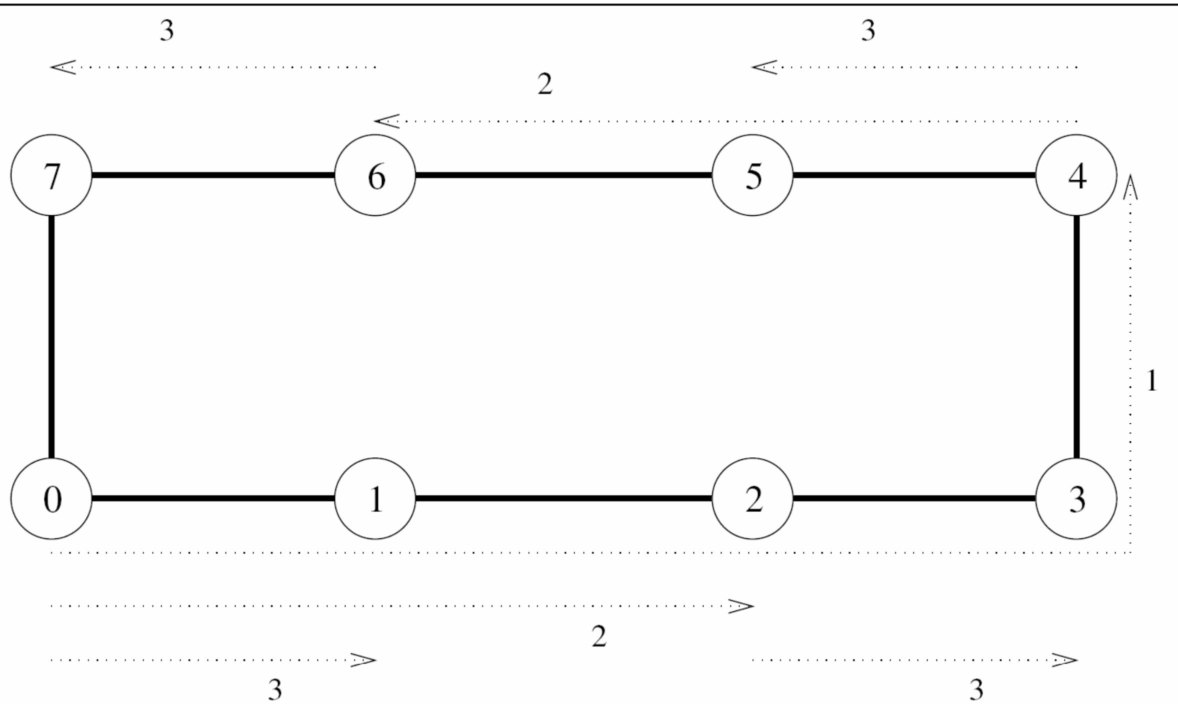


Figure 4.2 One-to-all broadcast on an eight-node ring. Node 0 is the source of the broadcast. Each message transfer step is shown by a numbered, dotted arrow from the source of the message to its destination. The number on an arrow indicates the time step during which the message is transferred.

Reduction on a Ring Algorithm

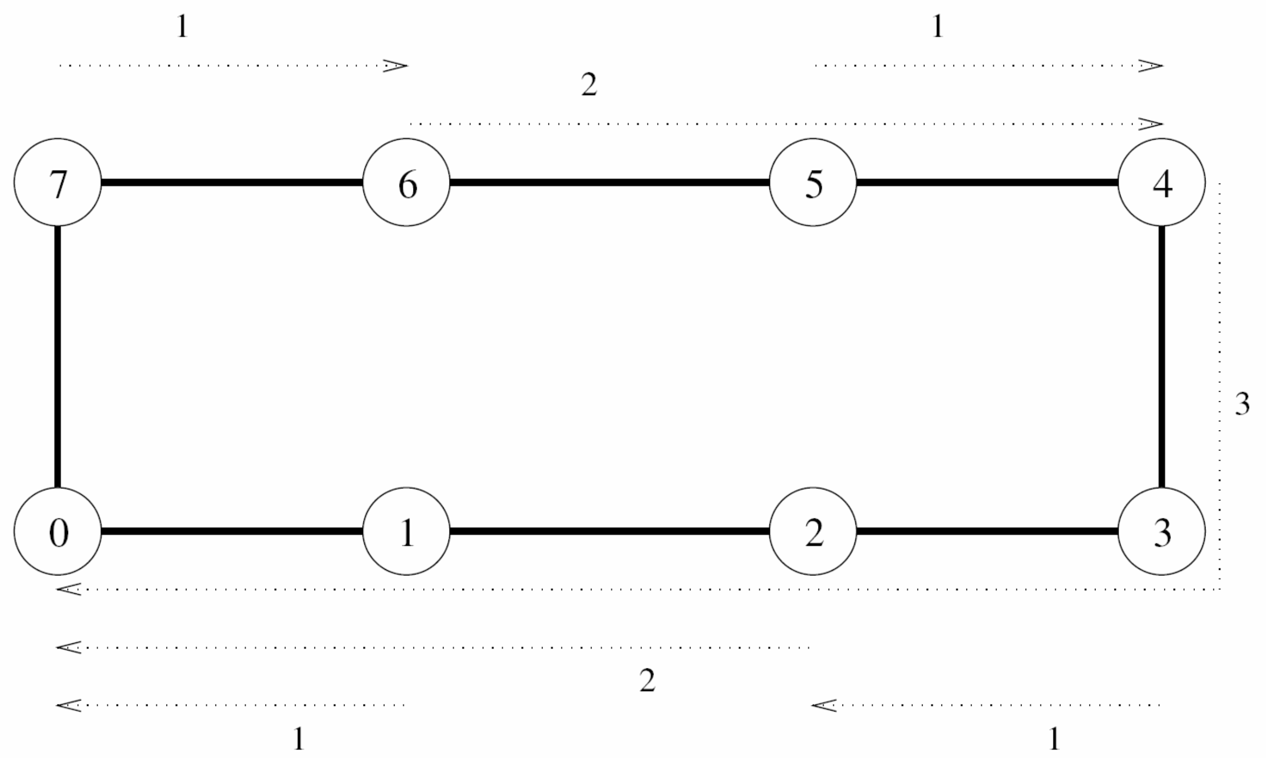


Figure 4.3 Reduction on an eight-node ring with node 0 as the destination of the reduction.

Broadcast on a Mesh

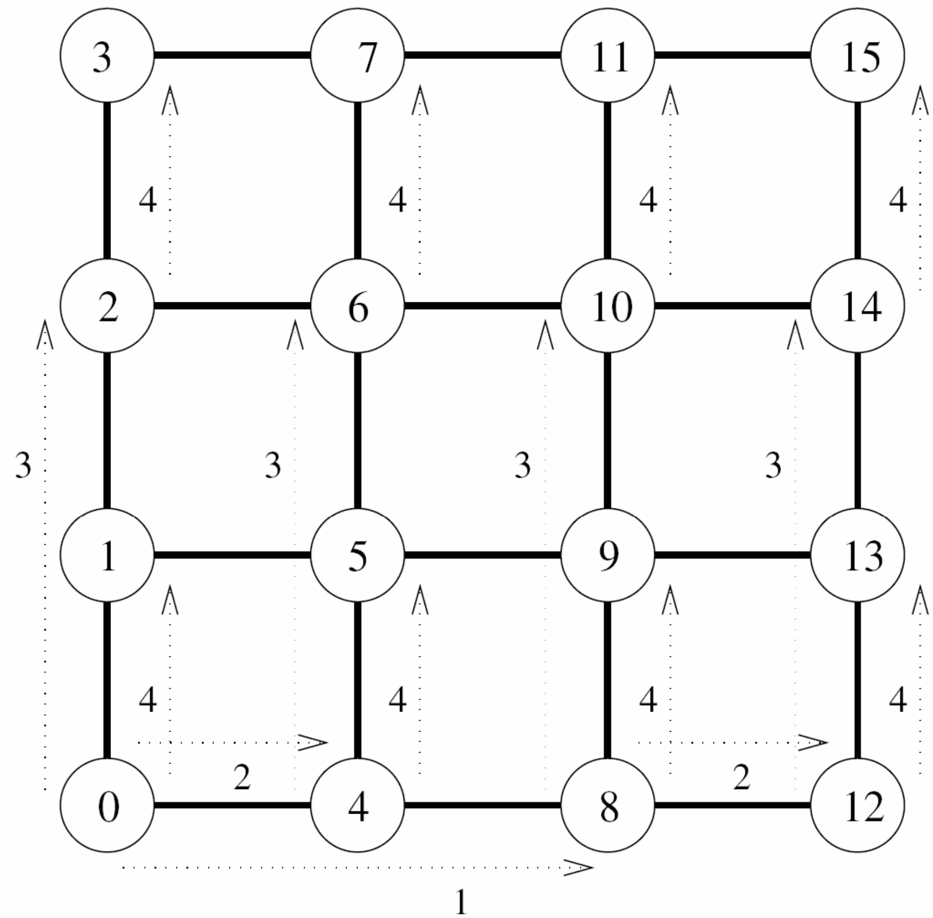


Figure 4.5 One-to-all broadcast on a 16-node mesh.

Broadcast on a Hypercube

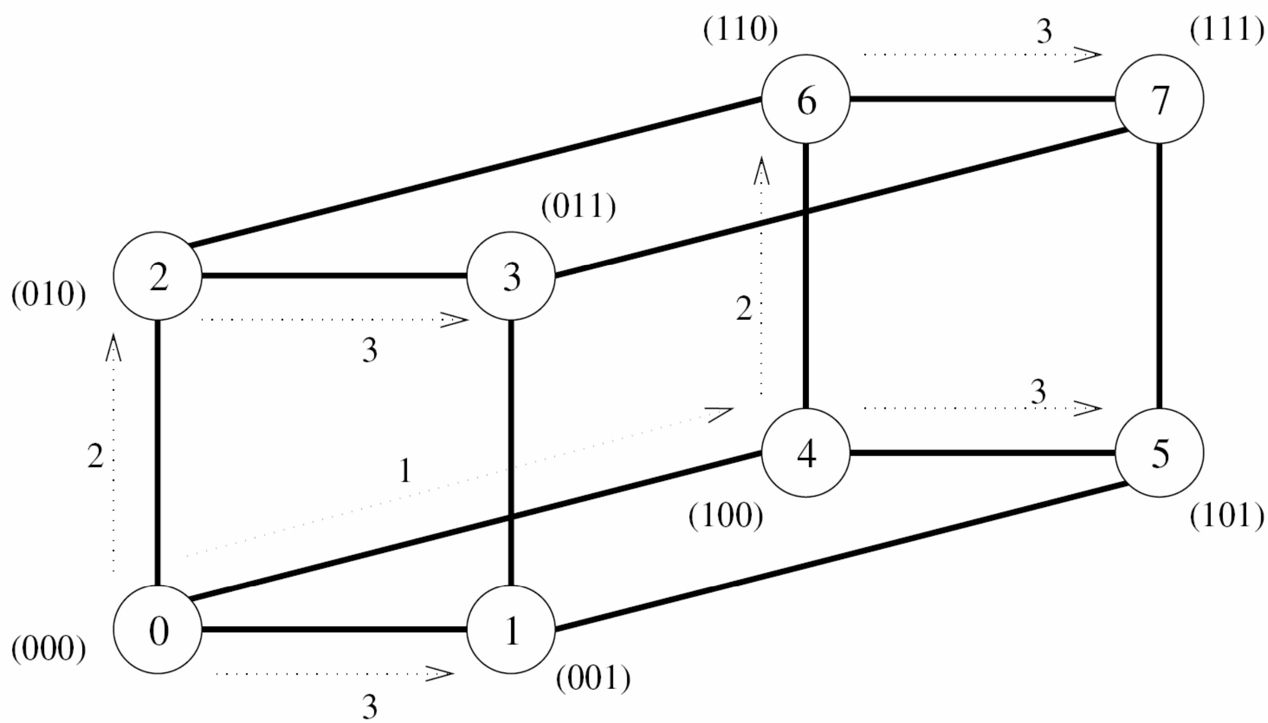


Figure 4.6 One-to-all broadcast on a three-dimensional hypercube. The binary representations of node labels are shown in parentheses.

Code for the Broadcast Source: Root

```
procedure ONE_TO_ALL_BC(d, my_id, X)
begin
    mask := 2^d − 1;                          /* Set all d bits of mask to 1 */
    for i := d − 1 downto 0 do                /* Outer loop */
        mask := mask XOR 2^i;                 /* Set bit i of mask to 0 */
        if (my_id AND mask) = 0 then          /* If lower i bits of my_id are 0 */
            if (my_id AND 2^i) = 0 then
                msg_destination := my_id XOR 2^i;
                send X to msg_destination;
            else
                msg_source := my_id XOR 2^i;
                receive X from msg_source;
            endelse;
        endif;
    endfor;
end ONE_TO_ALL_BC
```

Algorithm 4.1 One-to-all broadcast of a message X from node 0 of a d-dimensional p-node hypercube ($d = \log p$). AND and XOR are bitwise logical-and and exclusive-or operations, respectively.

Code for Broadcast Arbitrary Source

```
procedure GENERAL_ONE_TO_ALL_BC(d, my_id, source, X)
begin
    my_virtual_id := my_id XOR source;
    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 */
        if (my_virtual_id AND mask) = 0 then
            if (my_virtual_id AND 2^i) = 0 then
                virtual_dest := my_virtual_id XOR 2^i;
                send X to (virtual_dest XOR source);
    /* Convert virtual_dest to the label of the physical destination */
            else
                virtual_source := my_virtual_id XOR 2^i;
                receive X from (virtual_source XOR source);
    /* Convert virtual_source to the label of the physical source */
            endelse;
    endfor;
end GENERAL_ONE_TO_ALL_BC
```

Algorithm 4.2 One-to-all broadcast of a message X initiated by $source$ on a d-dimensional hypothetical hypercube. The AND and XOR operations are bitwise logical operations.

All-to-All Broadcast & Reduction

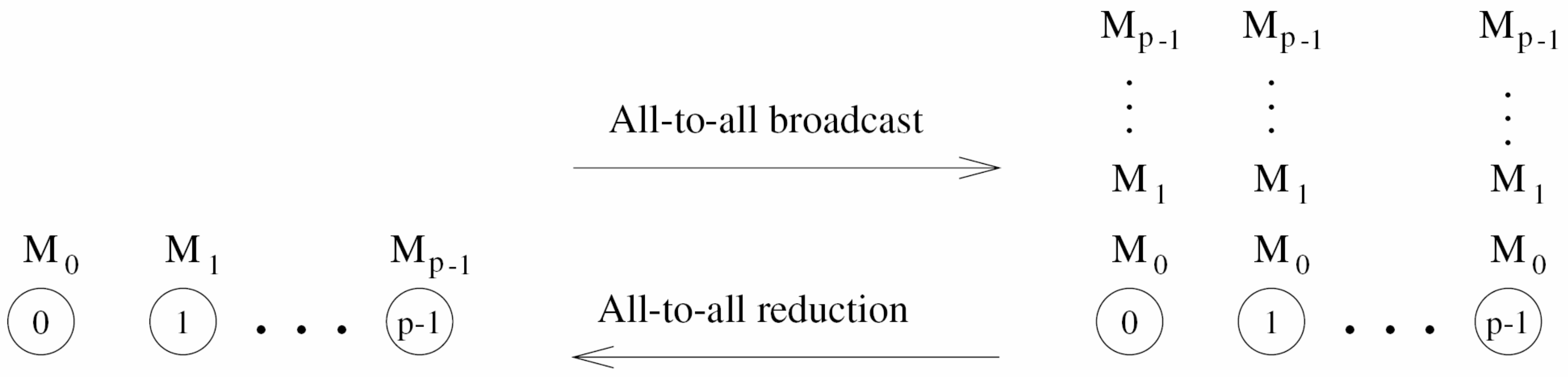
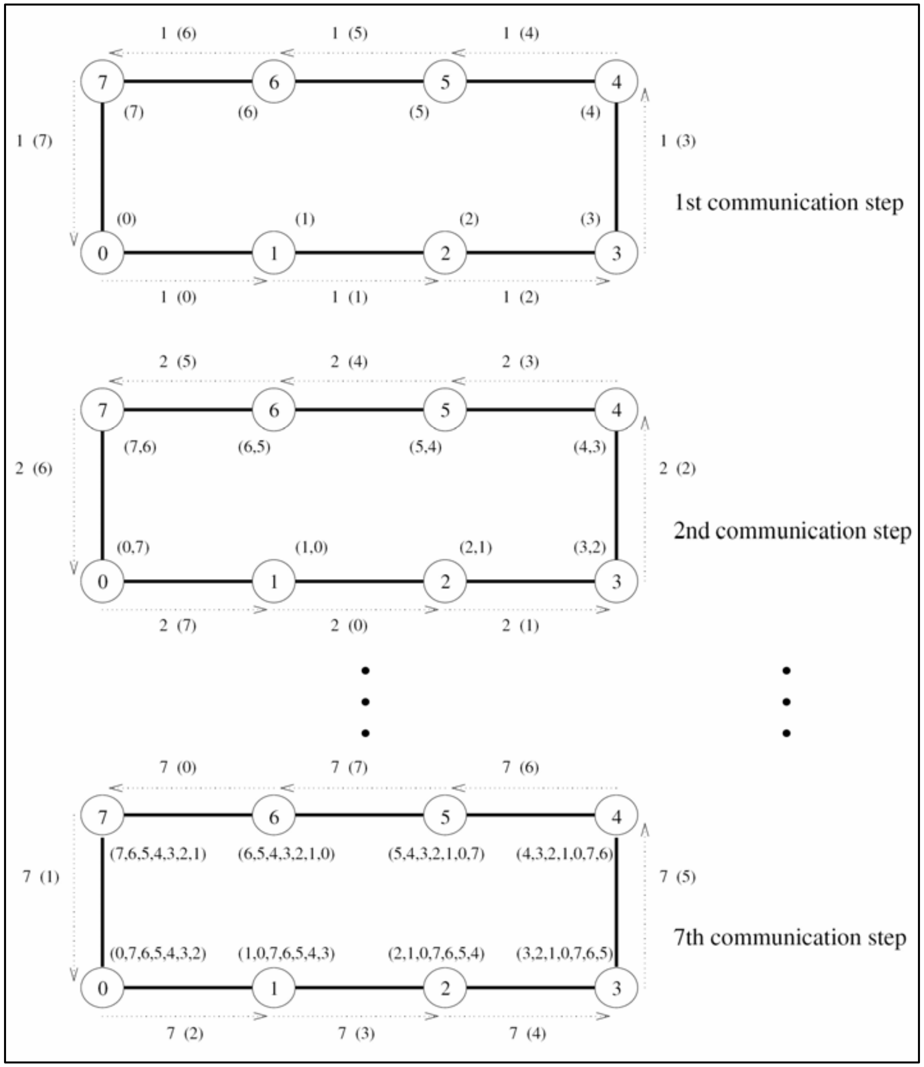


Figure 4.8 All-to-all broadcast and all-to-all reduction.

All-to-All Broadcast for Ring



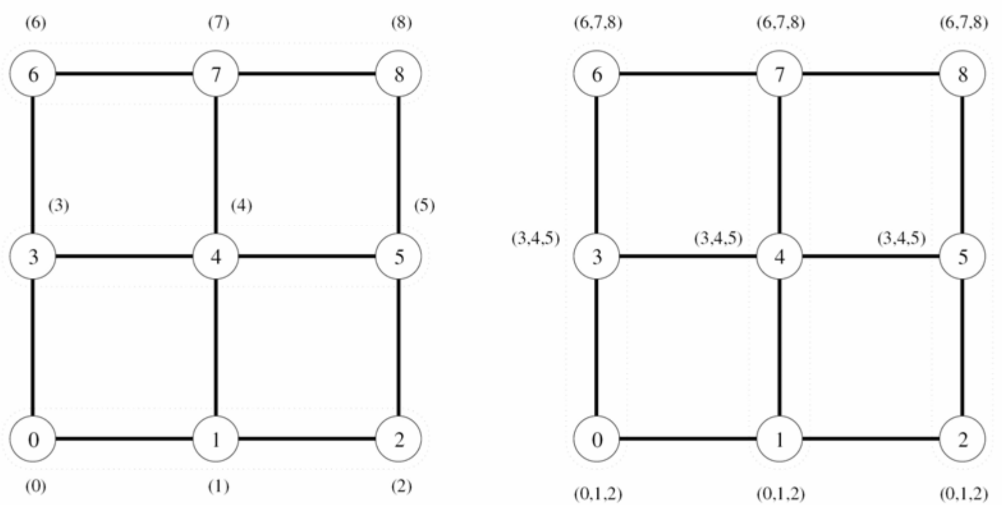

```
procedure ALL_TO_ALL_BC_RING(my_id, my_msg, p, result)
begin
    left := (my_id − 1) mod p;
    right := (my_id + 1) mod p;
    result := my_msg;
    msg := result;
    for i := 1 to p − 1 do
        send msg to right;
        receive msg from left;
        result := result ∪ msg;
    endfor;
end ALL_TO_ALL_BC_RING
```

Algorithm 4.4 All-to-all broadcast on a p-node ring.

```
procedure ALL_TO_ALL_RED_RING(my_id, my_msg, p, result)
begin
    left := (my_id − 1) mod p;
    right := (my_id + 1) mod p;
    recv := 0;
    for i := 1 to p − 1 do
        j := (my_id + i) mod p;
        temp := msg[j] + recv;
        send temp to left;
        receive recv from right;
    endfor;
    result := msg[my_id] + recv;
end ALL_TO_ALL_RED_RING
```

Algorithm 4.5 All-to-all reduction on a p-node ring.

All-to-All Broadcast on a Mesh



(a) Initial data distribution (b) Data distribution after rowwise broadcast

Figure 4.10 All-to-all broadcast on a 3×3 mesh. The groups of nodes communicating with each other in each phase are enclosed by dotted boundaries. By the end of the second phase, all nodes get (0,1,2,3,4,5,6,7) (that is, a message from each node).

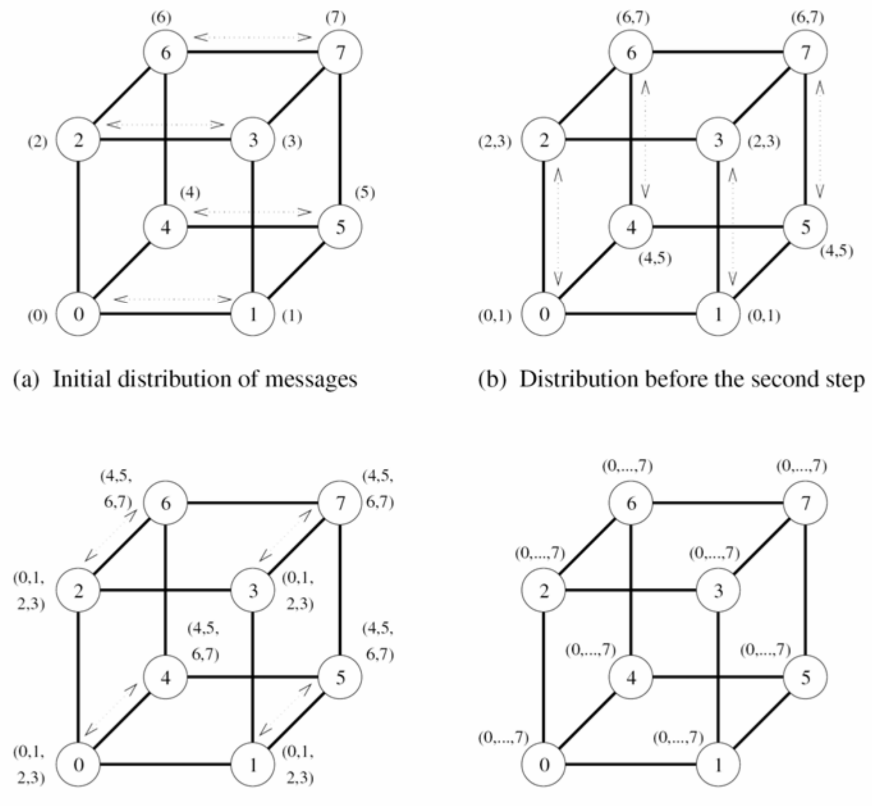
```
procedure ALL_TO_ALL_BC_MESH(my_id, my_msg, p, result)
begin

/* Communication along rows */
    left := my_id − (my_id mod √p) + (my_id − 1)mod √p;
    right := my_id − (my_id mod √p) + (my_id + 1) mod √p;
    result := my_msg;
    msg := result;
    for i := 1 to √p − 1 do
        send msg to right;
        receive msg from left;
        result := result ∪ msg;
    endfor;

/* Communication along columns */
    up := (my_id − √p) mod p;
    down := (my_id + √p) mod p;
    msg := result;
    for i := 1 to √p − 1 do
        send msg to down;
        receive msg from up;
        result := result ∪ msg;
    endfor;
end ALL_TO_ALL_BC_MESH
```

Algorithm 4.6 All-to-all broadcast on a square mesh of p nodes.

All-to-All Broadcast on a HCube



(a) Initial distribution of messages

(b) Distribution before the second step

(c) Distribution before the third step

(d) Final distribution of messages

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

```
procedure ALL_TO_ALL_BC_HCUBE(my_id, my_msg, d, result)
begin
    result := my_msg;
    for i := 0 to d − 1 do
        partner := my_id XOR 2^i;
        send result to partner;
        receive msg from partner;
        result := result ∪ msg;
    endfor;
end ALL_TO_ALL_BC_HCUBE
```

Algorithm 4.7 All-to-all broadcast on a d-dimensional hypercube.

```
procedure ALL_TO_ALL_RED_HCUBE(my_id, msg, d, result)
begin
    recloc := 0;
    for i := d − 1 to 0 do
        partner := my_id XOR 2^i;
        j := my_id AND 2^i;
        k := (my_id XOR 2^i) AND 2^i;
        senloc := recloc + k;
        recloc := recloc + j;
        send msg[senloc .. senloc + 2^i − 1] to partner;
        receive temp[0 .. 2^i − 1] from partner;
        for j := 0 to 2^i − 1 do
            msg[recloc + j] := msg[recloc + j] + temp[j];
        endfor;
    endfor;
    result := msg[my_id];
end ALL_TO_ALL_RED_HCUBE
```

Algorithm 4.8 All-to-all broadcast on a d-dimensional hypercube. AND and XOR are bitwise logical-and and exclusive-or operations, respectively.

All-Reduce & Prefix-Sum

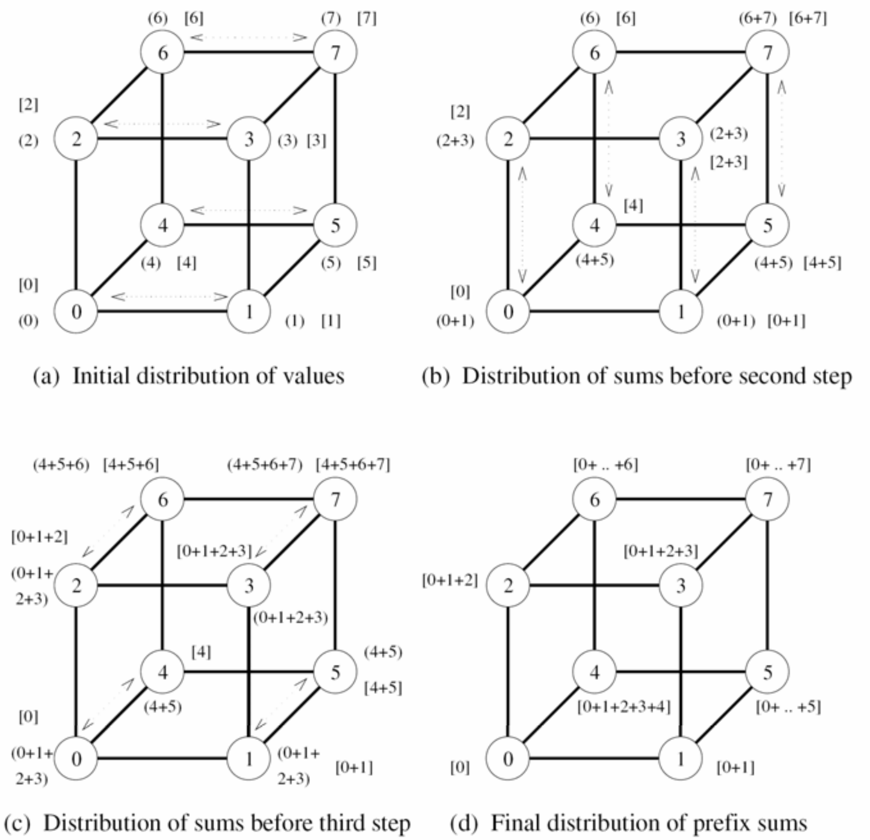


Figure 4.13 Computing prefix sums on an eight-node hypercube. At each node, square brackets show the local prefix sum accumulated in the result buffer and parentheses enclose the contents of the outgoing message buffer for the next step.

```
procedure PREFIX_SUMS_HCUBE(my_id, my_number, d, result)
begin
    result := my_number;
    msg := result;
    for i := 0 to d − 1 do
        partner := my_id XOR 2^i;
        send msg to partner;
        receive number from partner;
        msg := msg + number;
        if (partner < my_id) then result := result + number;
    endfor;
end PREFIX_SUMS_HCUBE
```

Algorithm 4.9 Prefix sums on a d-dimensional hypercube.

Scatter & Gather

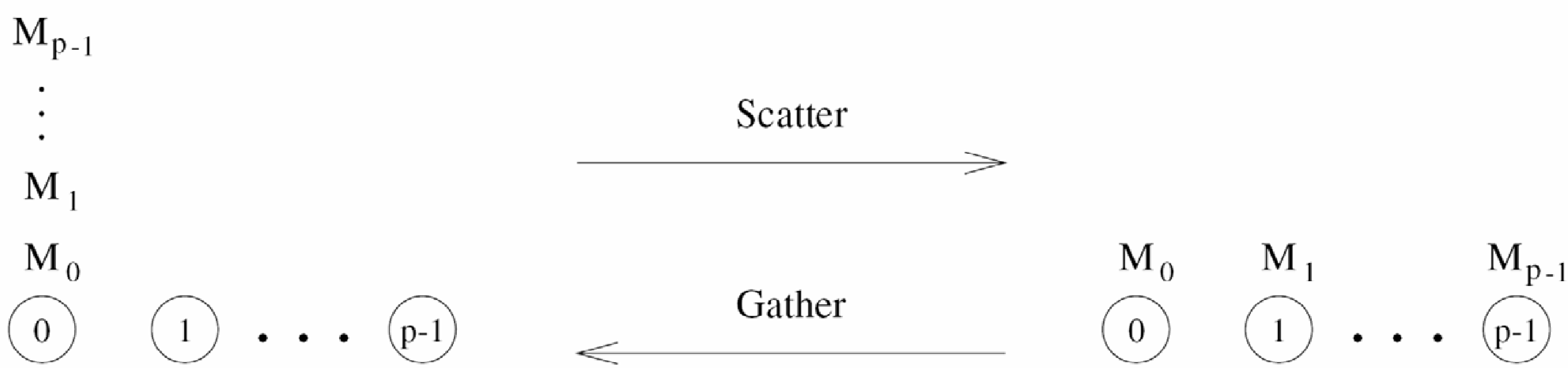
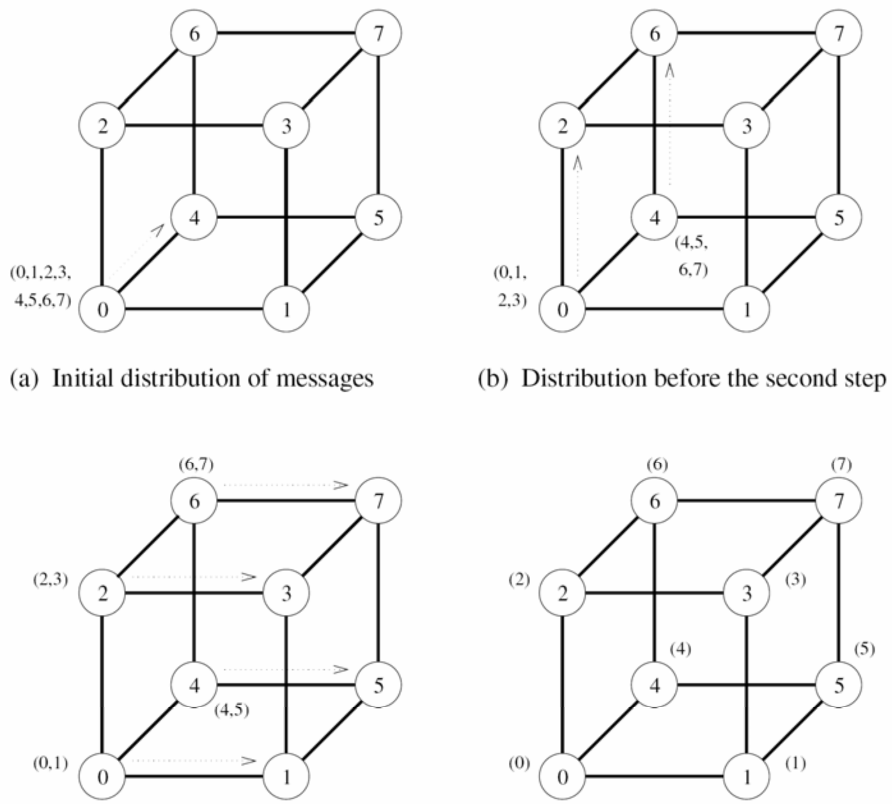


Figure 4.14 Scatter and gather operations.

Scatter Operation on HCube



(a) Initial distribution of messages

(b) Distribution before the second step

(c) Distribution before the third step

(d) Final distribution of messages

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

All-to-All Personalized (Transpose)

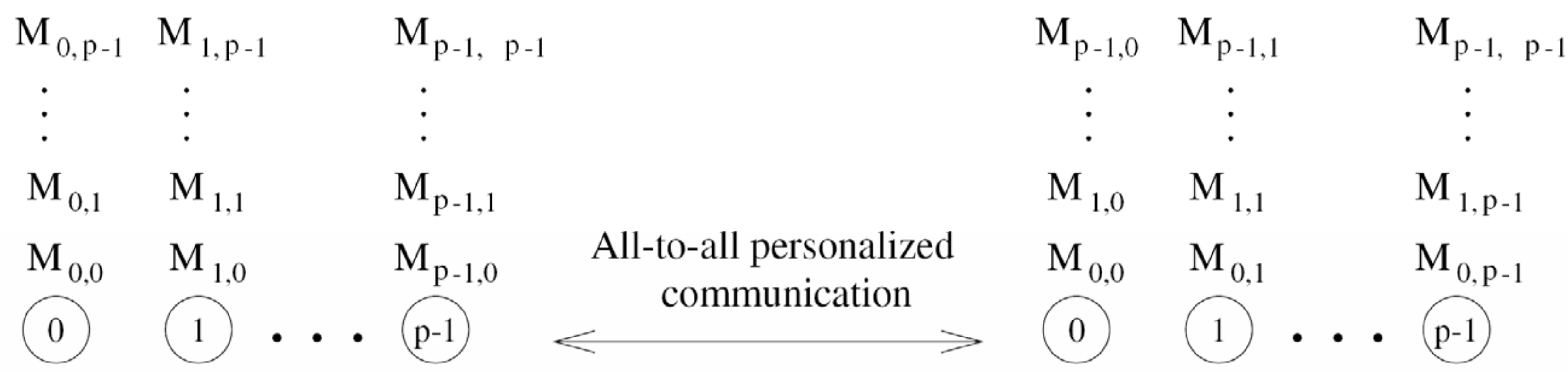


Figure 4.16 All-to-all personalized communication.

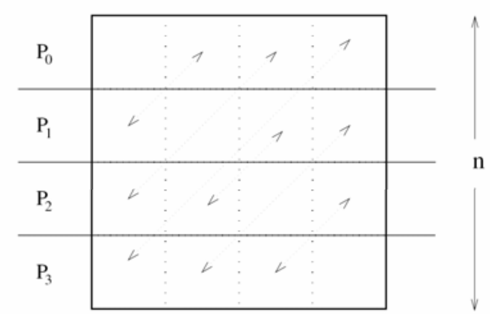


Figure 4.17 All-to-all personalized communication in transposing a 4×4 matrix using four processes.

All-to-all Personalized on a Ring

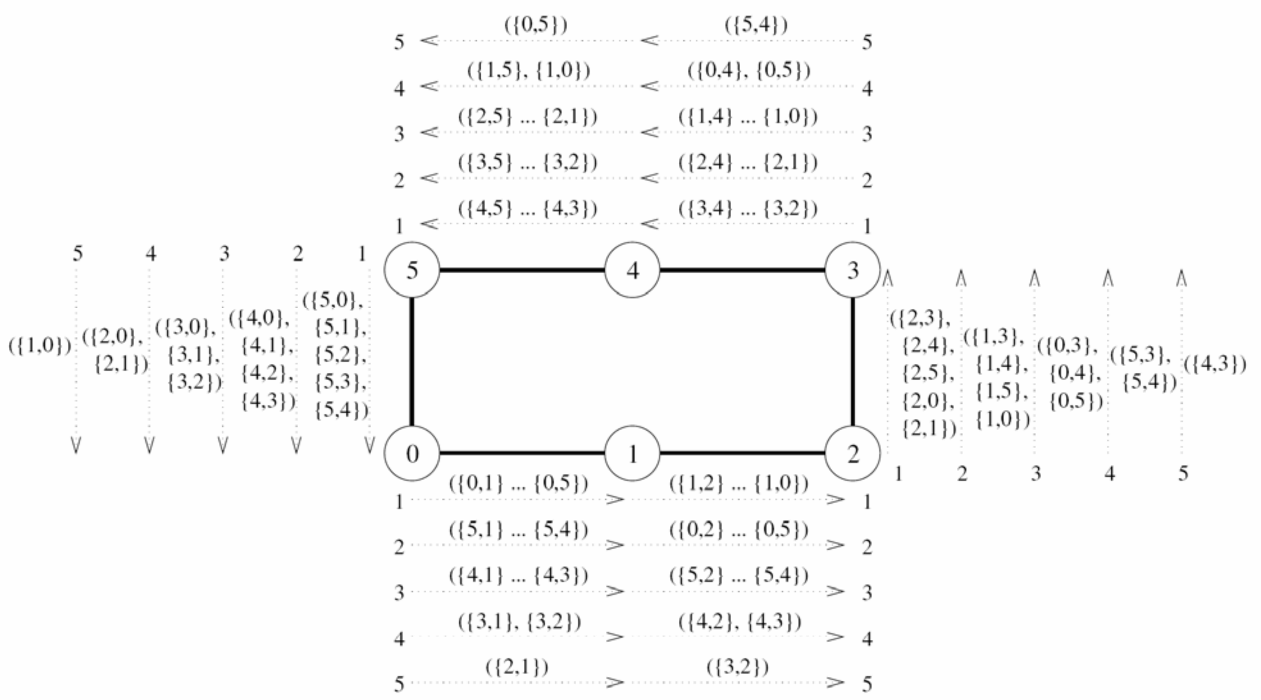


Figure 4.18 All-to-all personalized communication on a six-node ring. The label of each message is of the form $\{x, y\}$, where x is the label of the node that originally owned the message, and y is the label of the node that is the final destination of the message. The label $(\{x_1, y_1\}, \{x_2,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

All-to-all Personalized on a Mesh

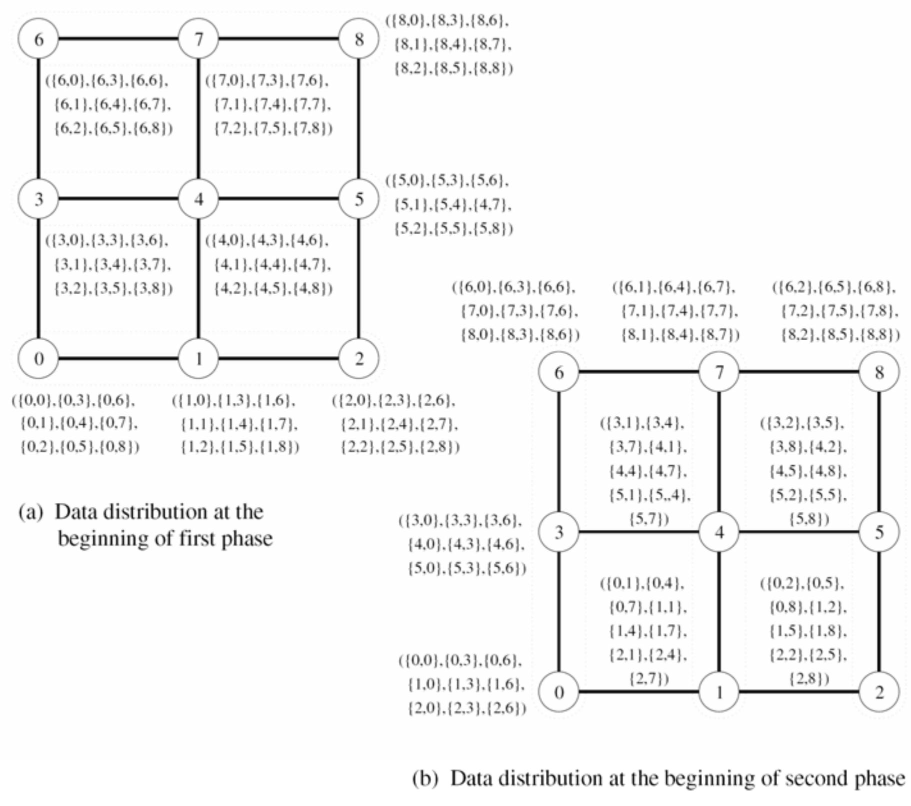


Figure 4.19 The distribution of messages at the beginning of each phase of all-to-all personalized communication on a 3×3 mesh. At the end of the second phase, node i has messages $(\{0,i\}, \ldots, \{8,i\})$, where $0 \le i \le 8$. The groups of nodes communicating together in each phase are enclosed in dotted boundaries.

All-to-all Personalized on a HCube

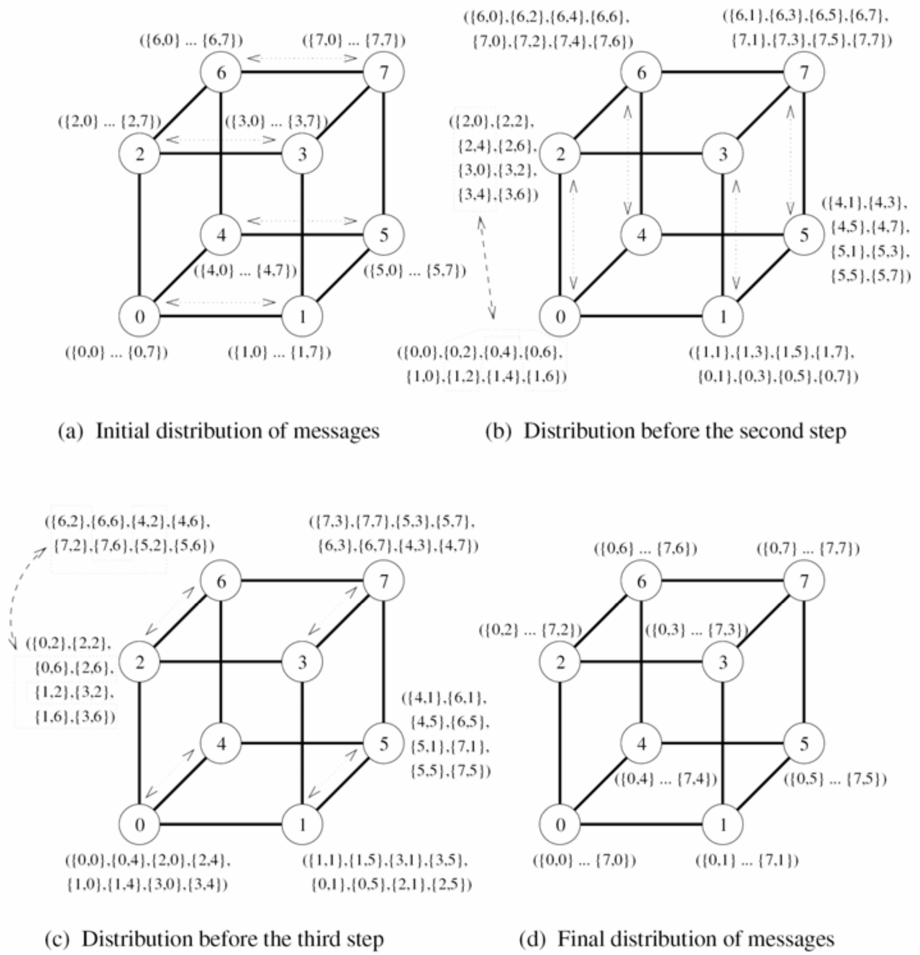


Figure 4.20 An all-to-all personalized communication algorithm on a three-dimensional hypercube.

All-to-all Personalized on a HCube Improved Algorithm

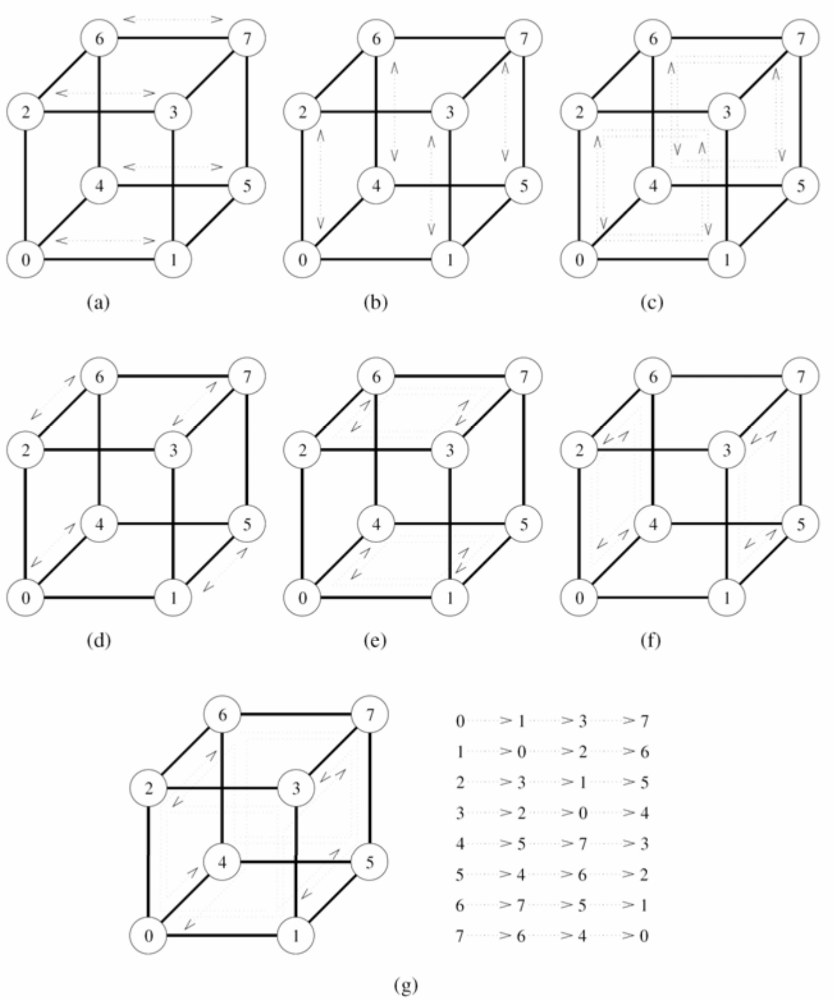


Figure 4.21 Seven steps in all-to-all personalized communication on an eight-node hypercube.

Perform log(p) point-to-point communication steps

Processor *i* communicates with processor *iXORj* during the *j*th communication step.

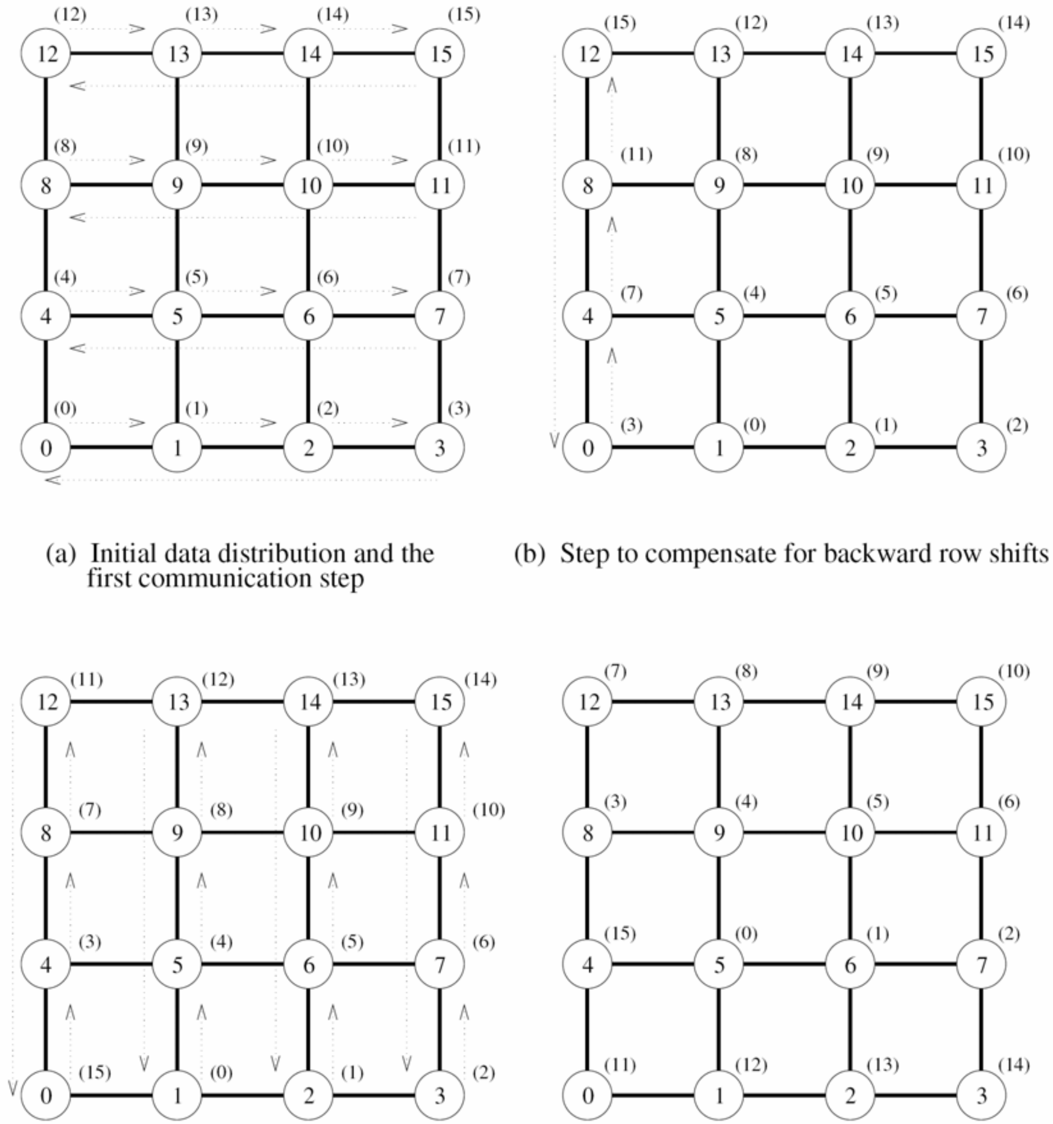
procedure ALL_TO_ALL_PERSONAL(d, my_id)
begin
  for $i := 1$ **to** $2^d - 1$ **do**
  begin
    $partner := my\_id$ XOR i;
    send $M_{my\_id, partner}$ to $partner$;
    receive $M_{partner, my\_id}$ from $partner$;
  endfor;
end ALL_TO_ALL_PERSONAL

Algorithm 4.10 A procedure to perform all-to-all personalized communication on a d-dimensional hypercube. The message $M_{i,j}$ initially resides on node i and is destined for node j.

Complexities

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

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



(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.

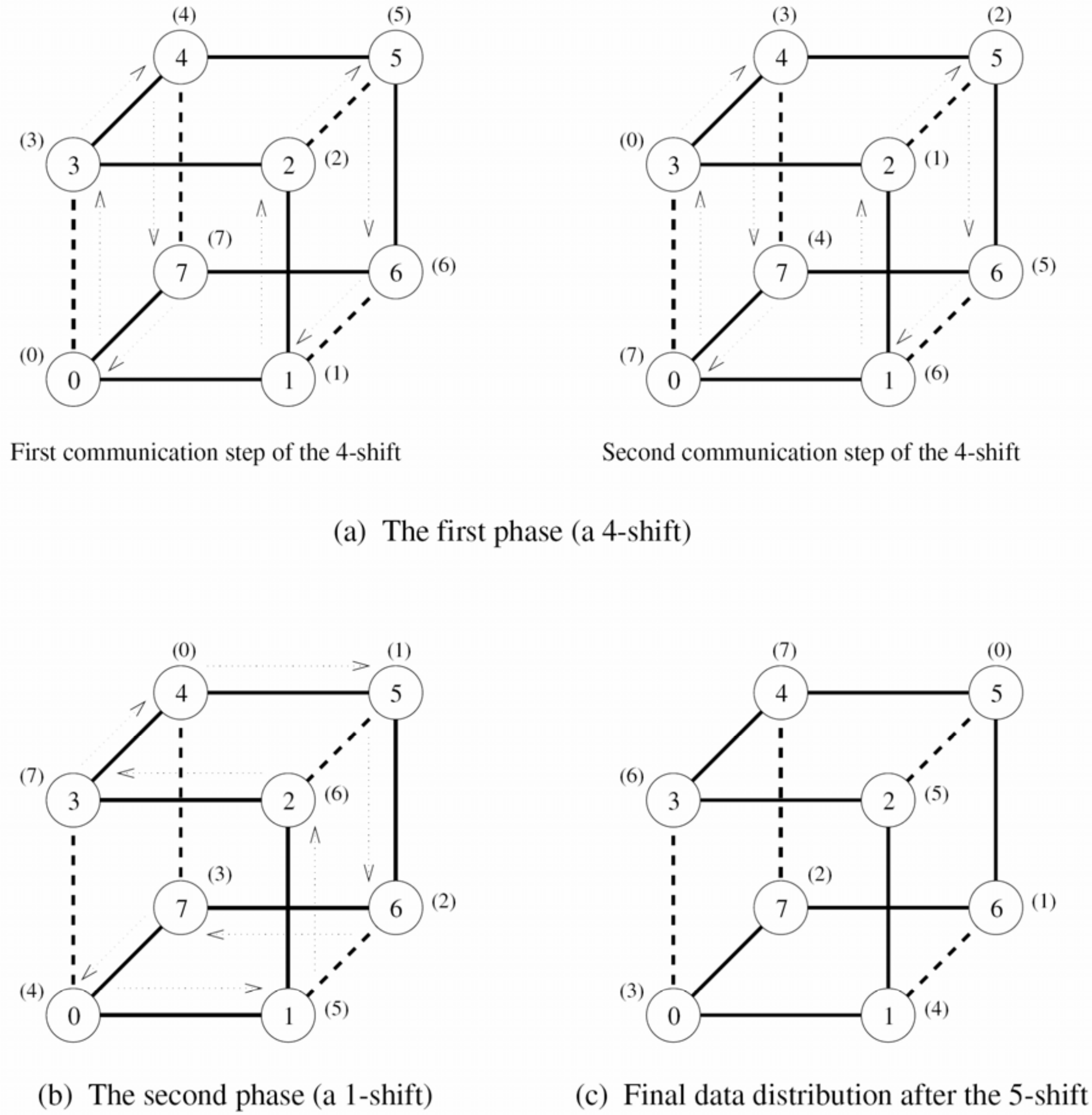


Figure 4.23 The mapping of an eight-node linear array onto a three-dimensional hypercube to perform a circular 5-shift as a combination of a 4-shift and a 1-shift.

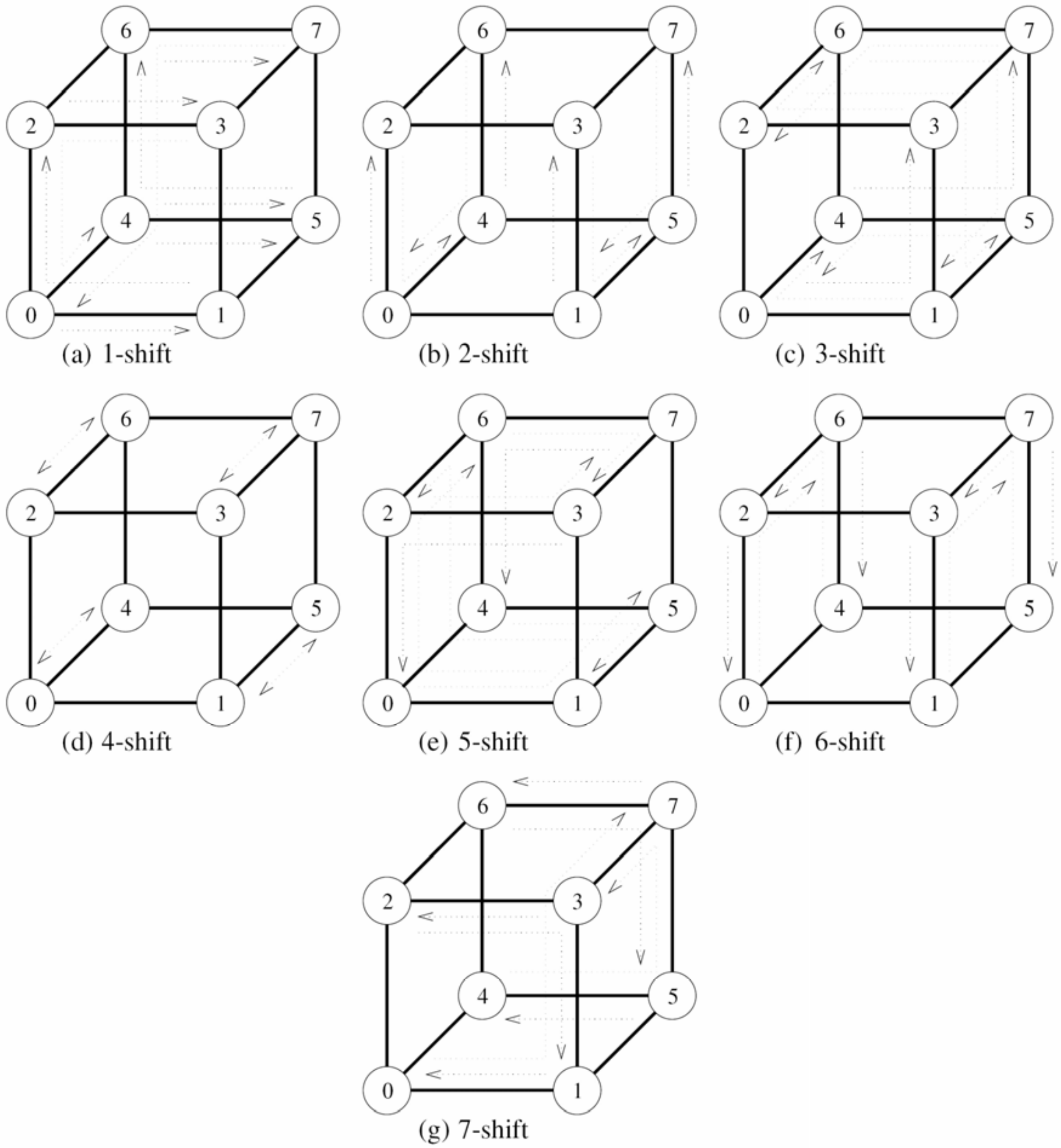


Figure 4.24 Circular q-shifts on an 8-node hypercube for $1 \leq q < 8$.