Interconnection Networks and Communication Patterns

SC3260/5260 High-Performance Computing

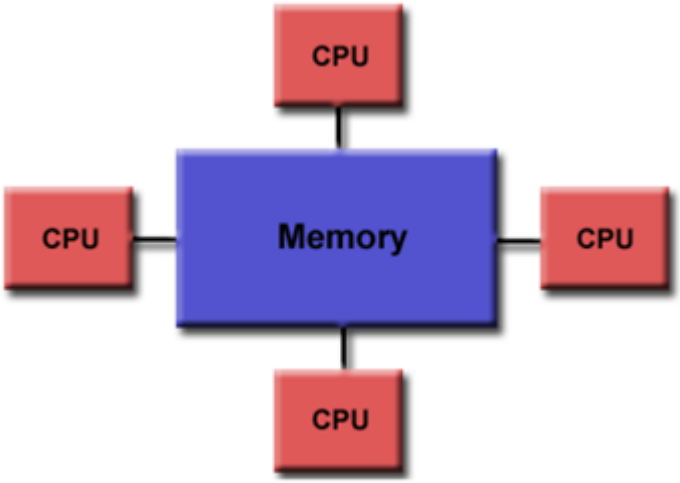
Hongyang Sun
(hongyang.sun@vanderbilt.edu)

Vanderbilt University

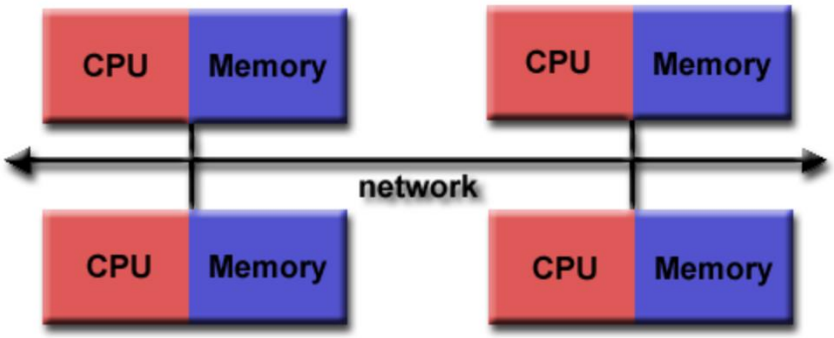
Spring 2020

Shared-Memory vs. Distributed Memory

- Shared-memory architecture



- Distributed-memory architecture

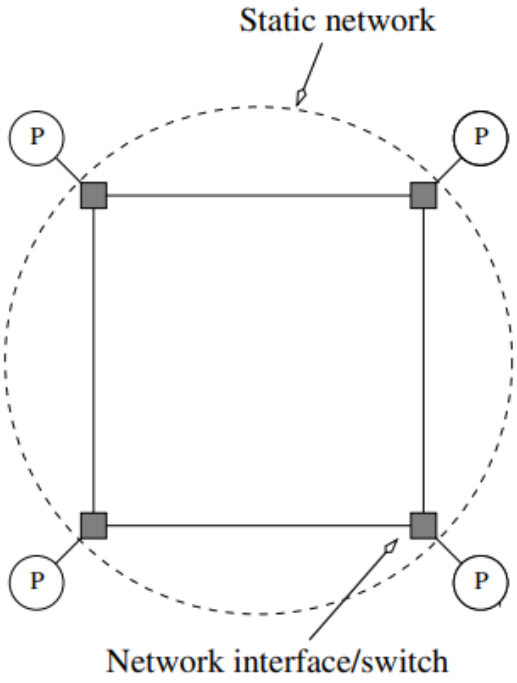


Interconnection Networks

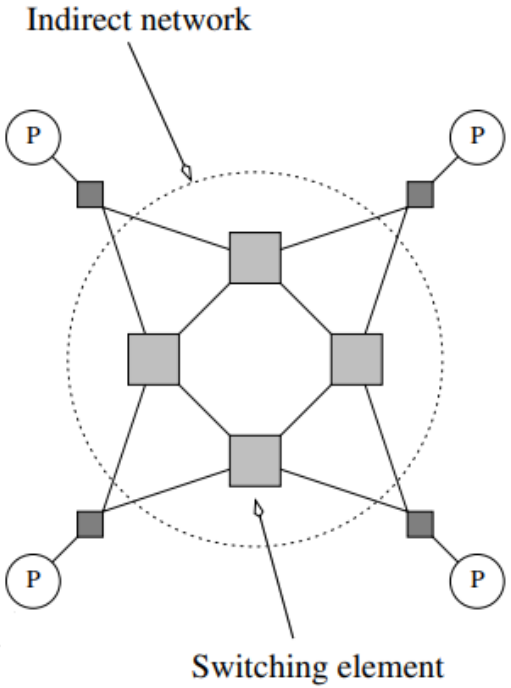
- Interconnection networks (made of switches and links) provide mechanisms for data communication between processors in distributed-memory architectures.
- Interconnects are classified as static or dynamic.
  - Static (direct) networks consist of point-to-point communication links among processing nodes.
  - Dynamic (indirect) networks are built using switches and communication links.

Static and Dynamic Networks

- Static (direct) network

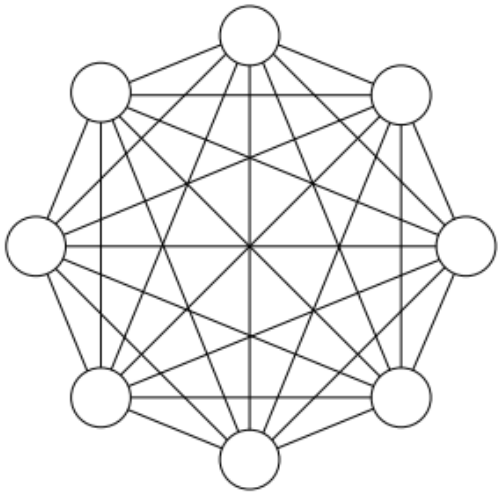


- Dynamic (indirect) network

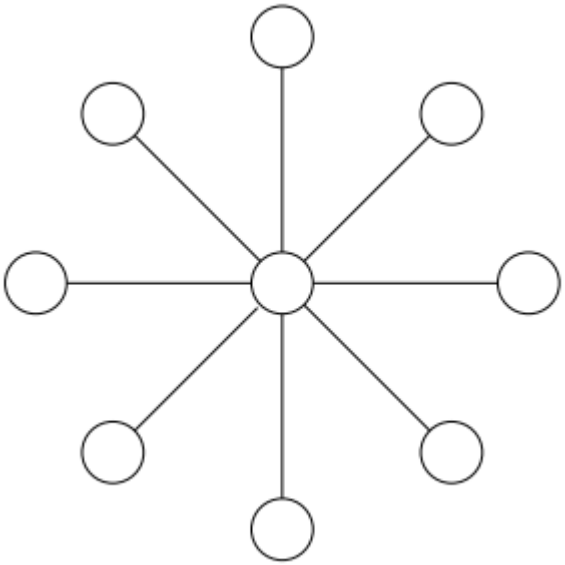


Static (Direct) Networks

- Completely-connected network



- Star network



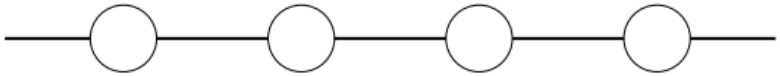
What are the advantageous and disadvantageous about these networks?

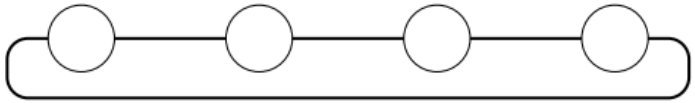
Performance Metrics of Interconnects

- Cost: total number of connecting links/wires in the network.
- Degree: maximum degree of any node in the network.
- Diameter: maximum distance (shortest path) of any two nodes in the network.
- Bisection width: minimum number of links that have to be removed to partition the network into (roughly) two equal halves.
- # Switches: number of switches in the network (for dynamic/indirect networks).

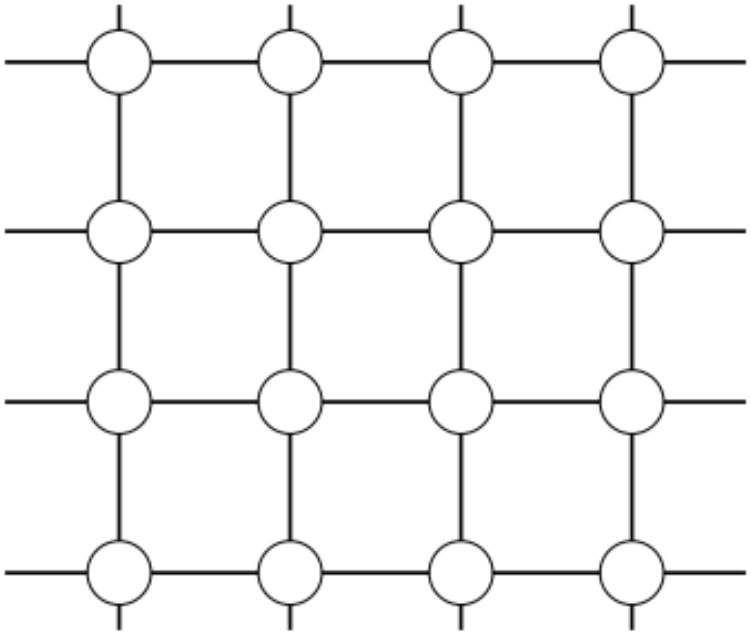
Static Networks

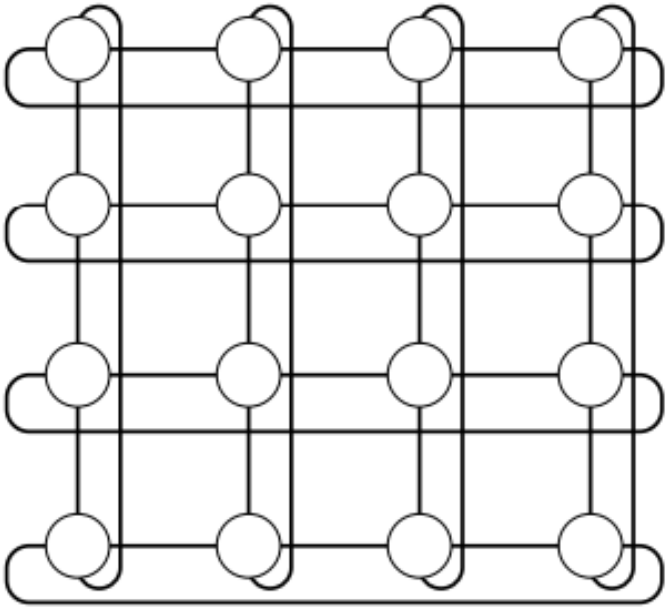
- 1D linear array and ring





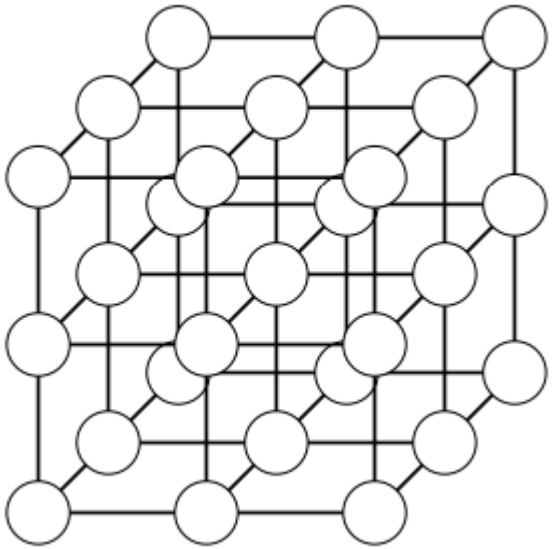
- 2D mesh and torus





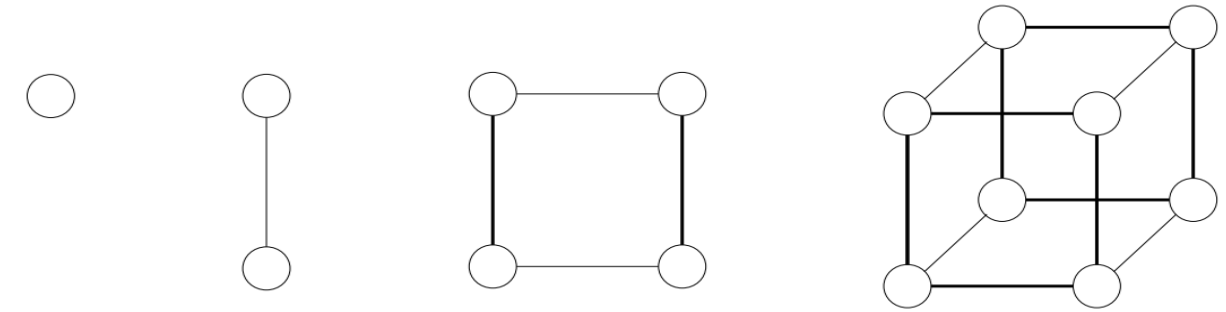
Static Networks

- 3D mesh/torus

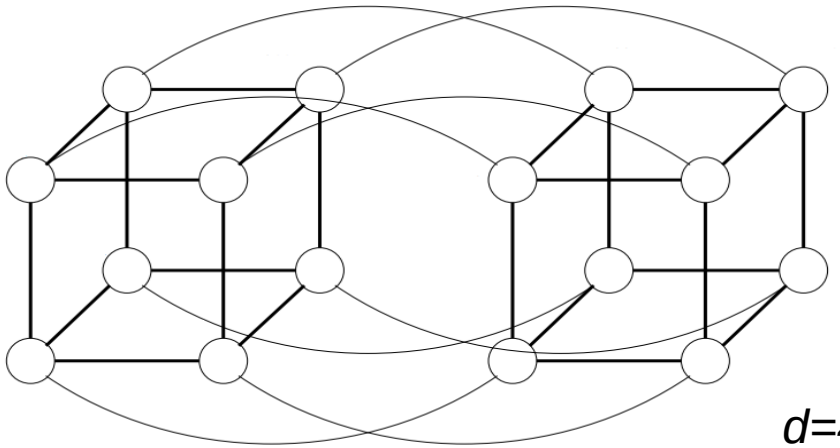


Static Networks

- Hypercube (with dimension *d*): *connect two identical hypercubes of dimension d-1.*



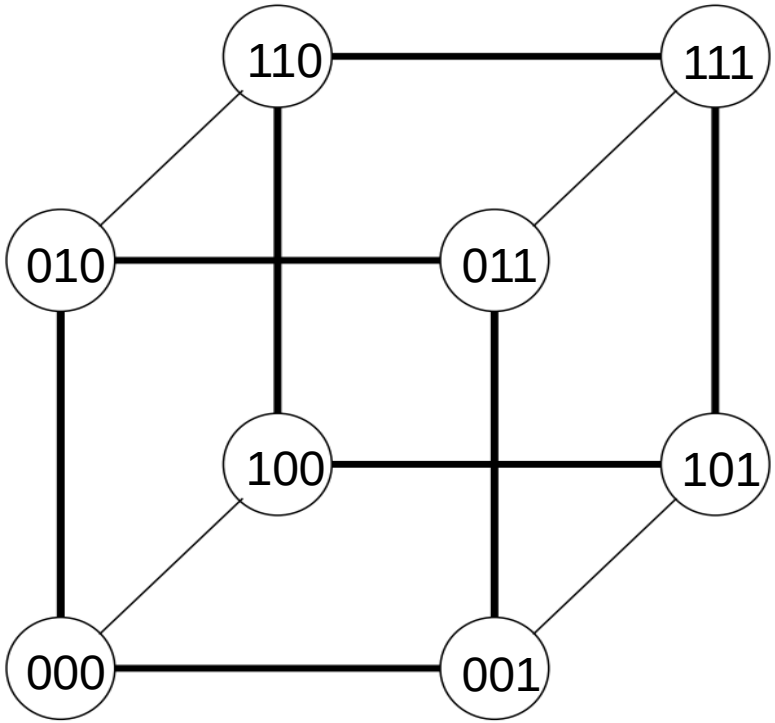
d=0 *d=1* *d=2* *d=3*



d=4

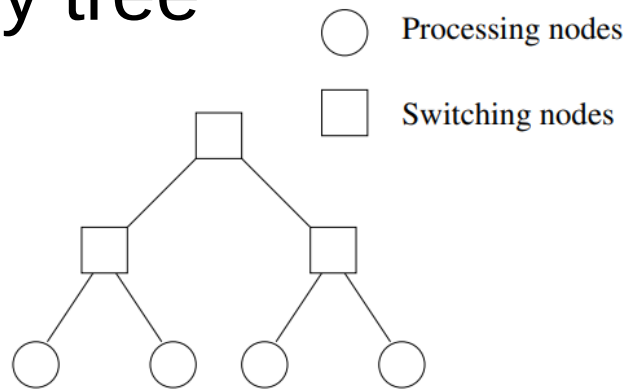
Static Networks

- Hypercube (with dimension *d*): *connect two nodes if their binary representations differ only in 1 bit position.*

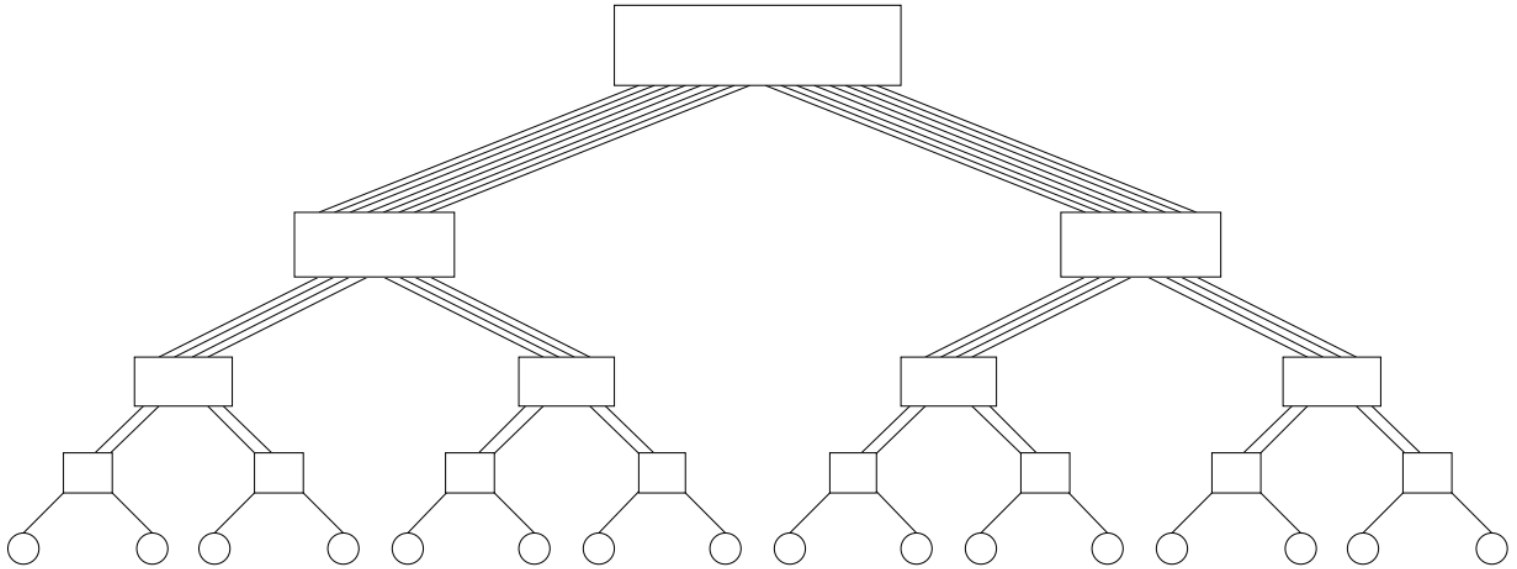


Dynamic Networks

- Complete binary tree



- Fat Tree



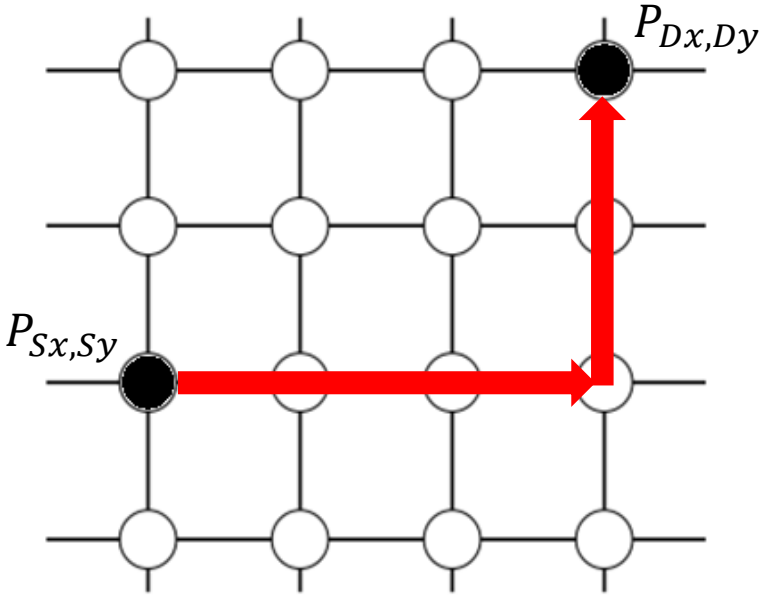
Communication Patterns

- Point-to-point communication
  - Routing a message from one source node to one destination node in the network.
- Collective communication
  - Sending/receiving messages involving a group of nodes in well-defined patterns (e.g., one-to-all broadcast, all-to-all broadcast, all-reduce).

Efficient implementations of the communication patterns leverage underlying interconnection networks and dynamic traffic, but they are transparent to the programmers.

Point-to-Point Communication

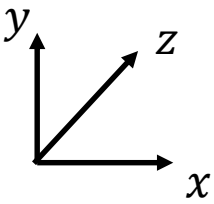
- XY-routing (for 2D mesh):
  - first route along X dimension to column of destination node, and then along Y dimension to destination.



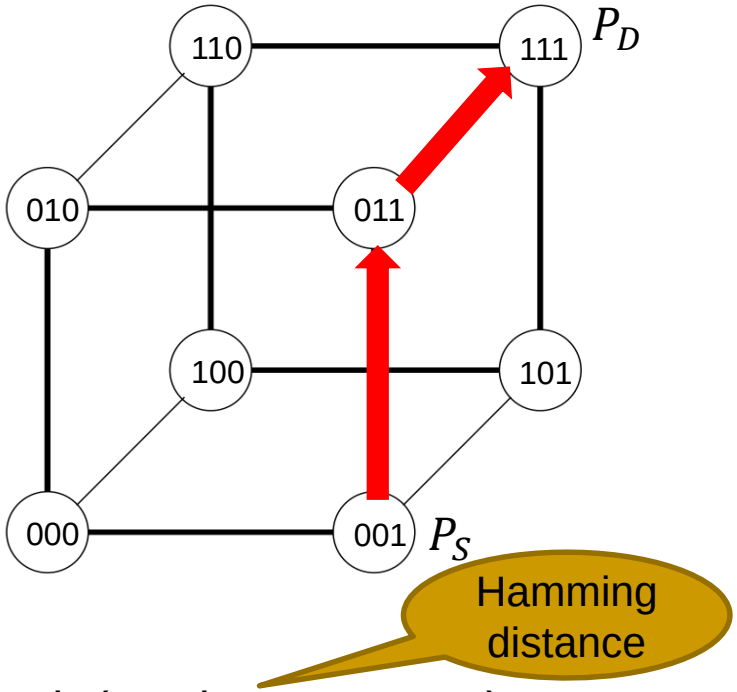
Manhattan distance

XY-routing returns a path of minimum length $|S_x - D_x| + |S_y - D_y|$

Point-to-Point Communication



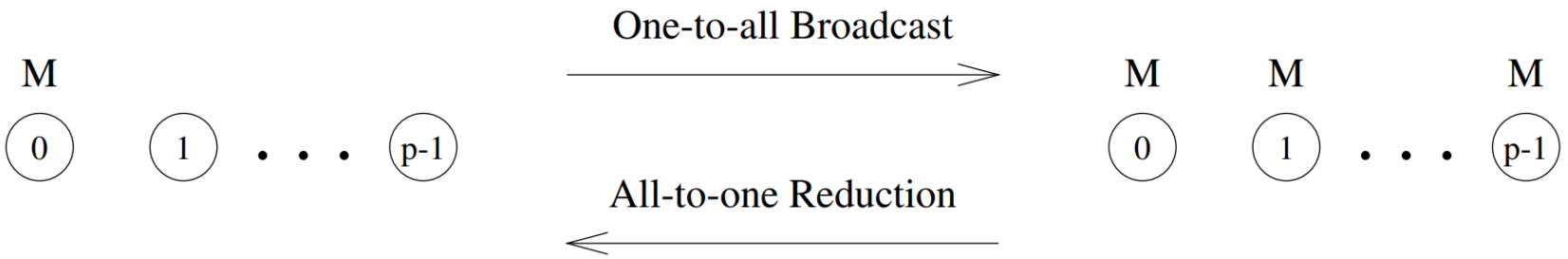
- E-cube routing (for hypercube):
  - Each bit position represents a dimension (e.g., zyx)
  - Compute bit-wise exclusive-OR (XOR) operation of the source and destination node representation $Q = P_S \oplus P_D$
  - Route message along each dimension where the corresponding bit in Q is 1 (from least significant bit).



E-cube routing returns a path of minimum length (#1s in $Q = P_S \oplus P_D$)

Collective Communication

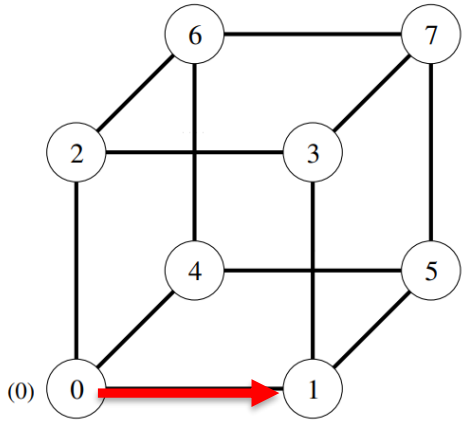
- One-to-all broadcast / All-to-one reduction

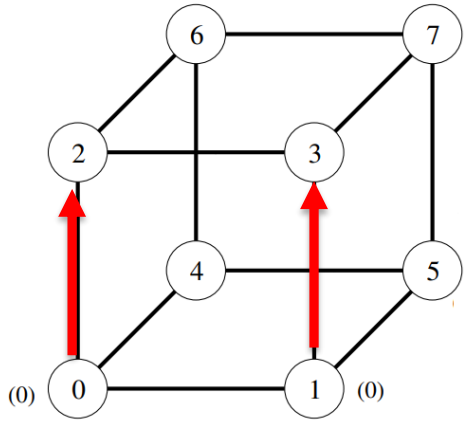


Collective Communication

- One-to-all broadcast **on hypercube** ($\log P$ steps)

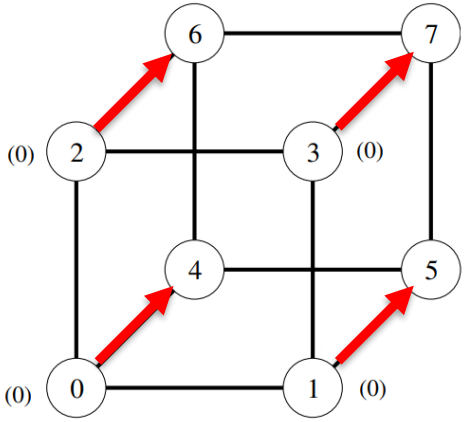
Step 0

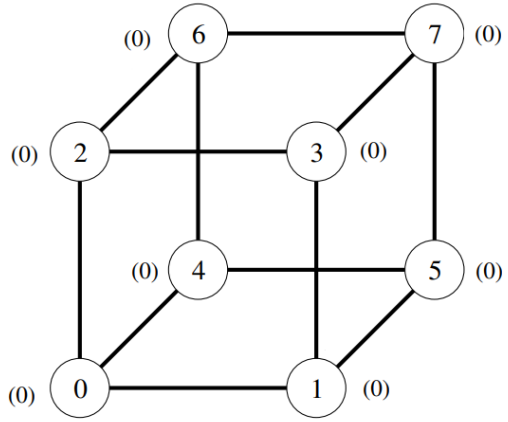




Step 1

Step 2

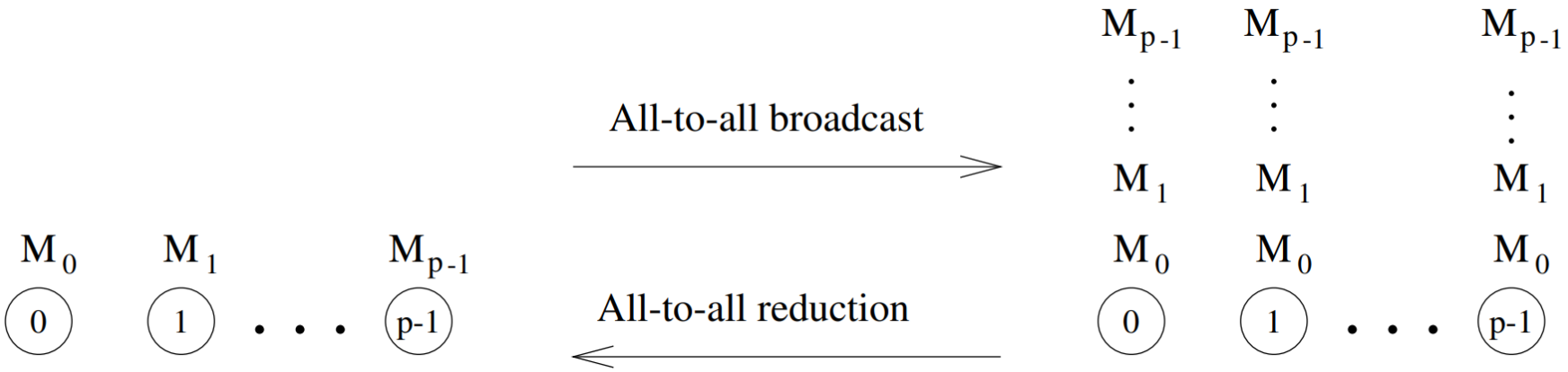




Step 3

Collective Communication

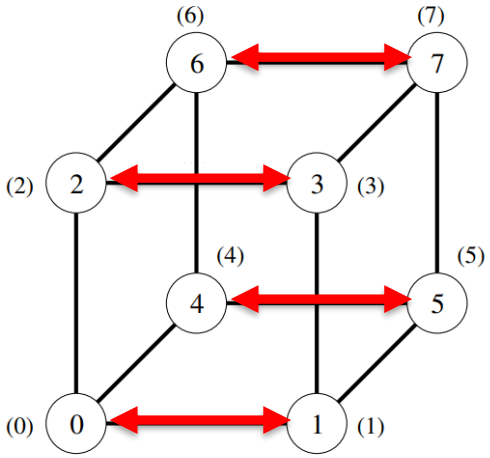
- All-to-all broadcast / All-to-all reduction

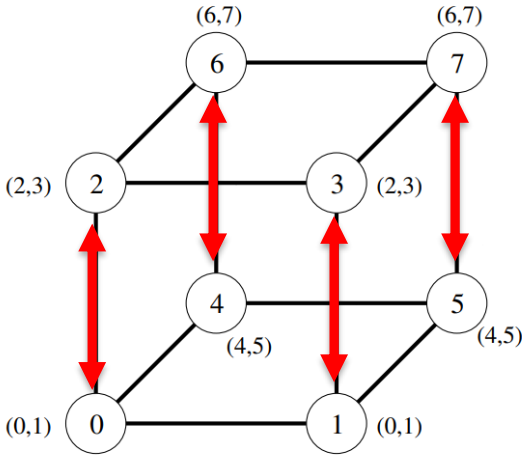


Collective Communication

- All-to-all broadcast **on hypercube** ($\log P$ steps)

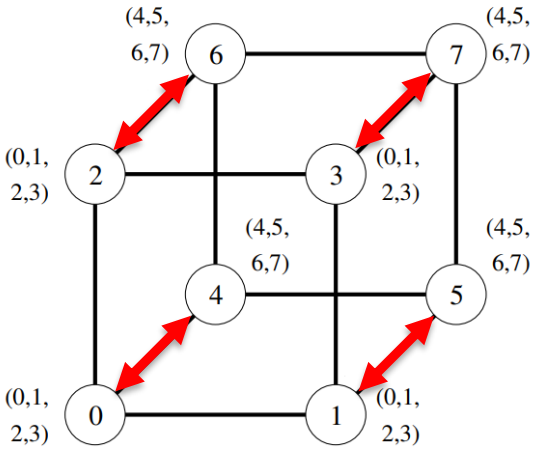
Step 0

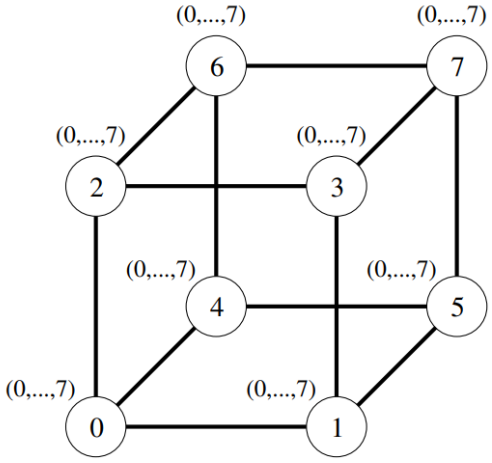




Step 1

Step 2

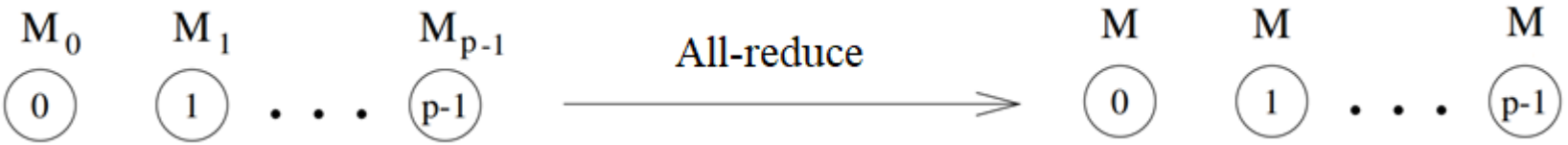




Step 3

Collective Communication

- All-reduce

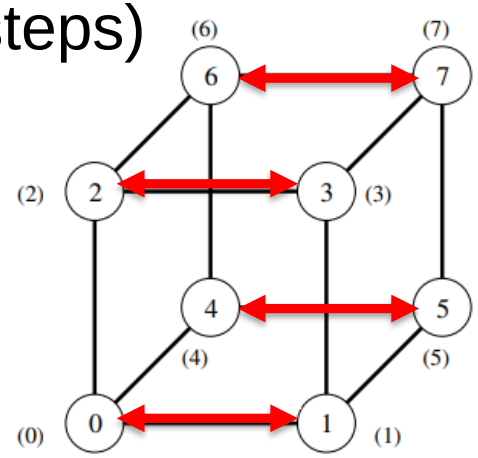


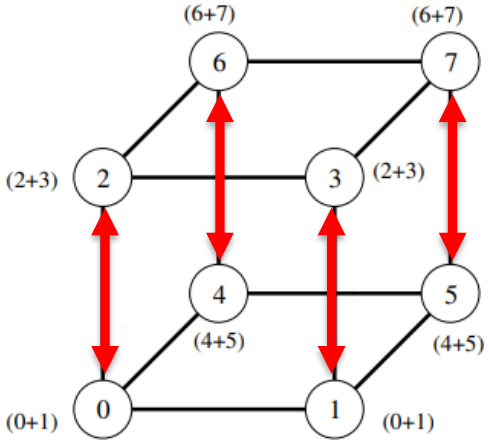
  - Implementation 1:
    - All-to-one reduction, followed by one-to-all broadcast.
    - $2 \log P$ communication steps.
  - Implementation 2:
    - Using the same idea of all-to-all broadcast.
    - $\log P$ communication steps but redundant computation.

Collective Communication

- Implementation 2: All-reduce **on hypercube** ($\log P$ steps)

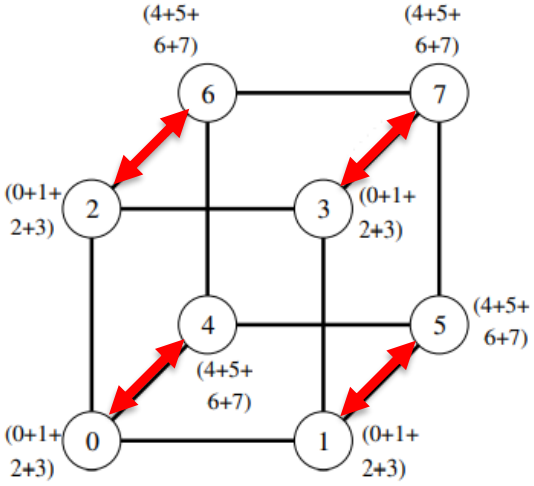
Step 0

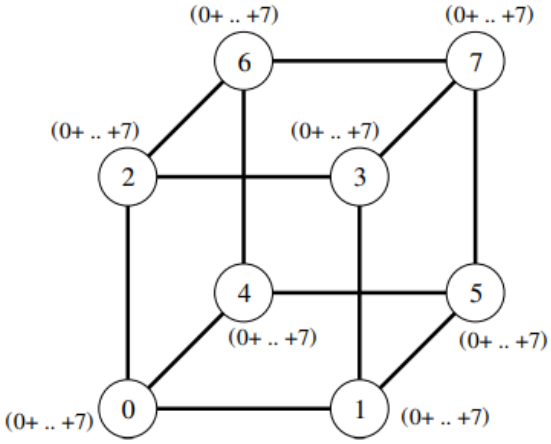




Step 1

Step 2

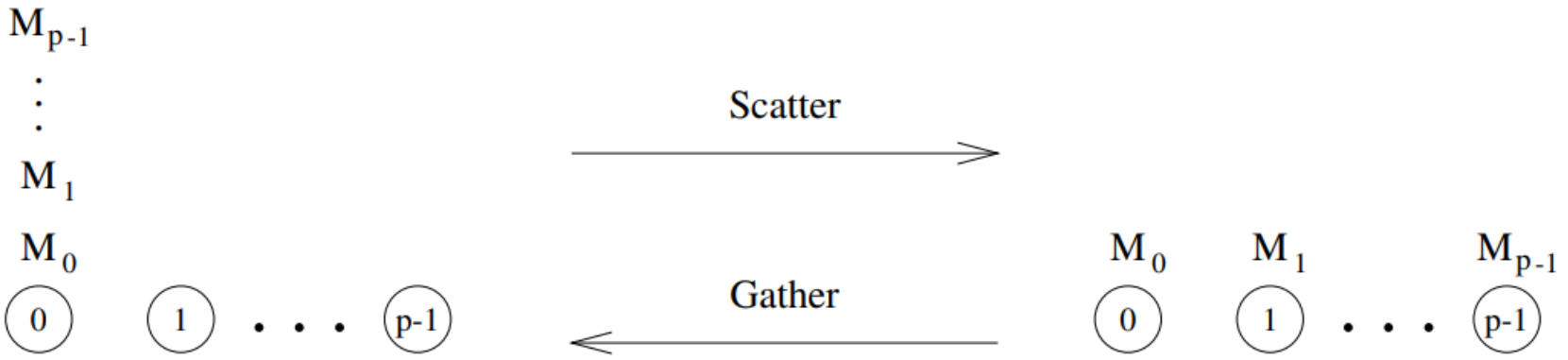




Step 3

Collective Communication

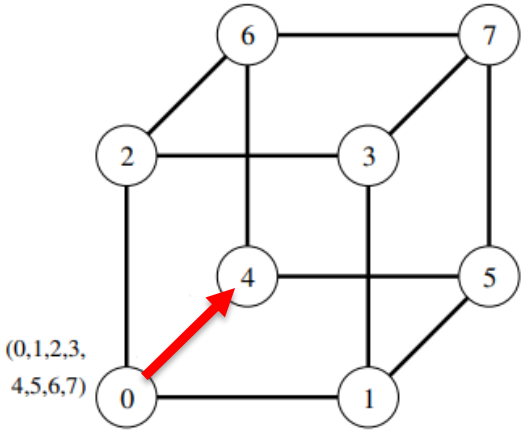
- Scatter / Gather

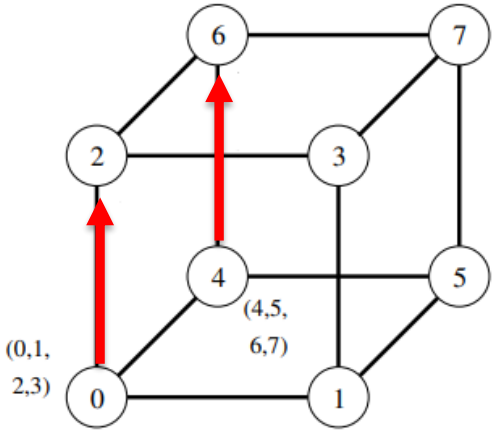


Collective Communication

- Scatter **on hypercube** (log P steps)

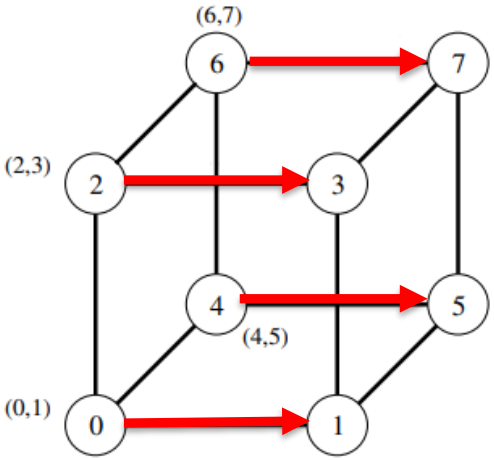
Step 0

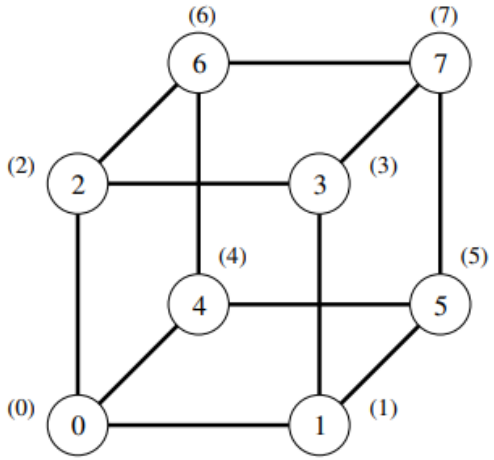




Step 1

Step 2

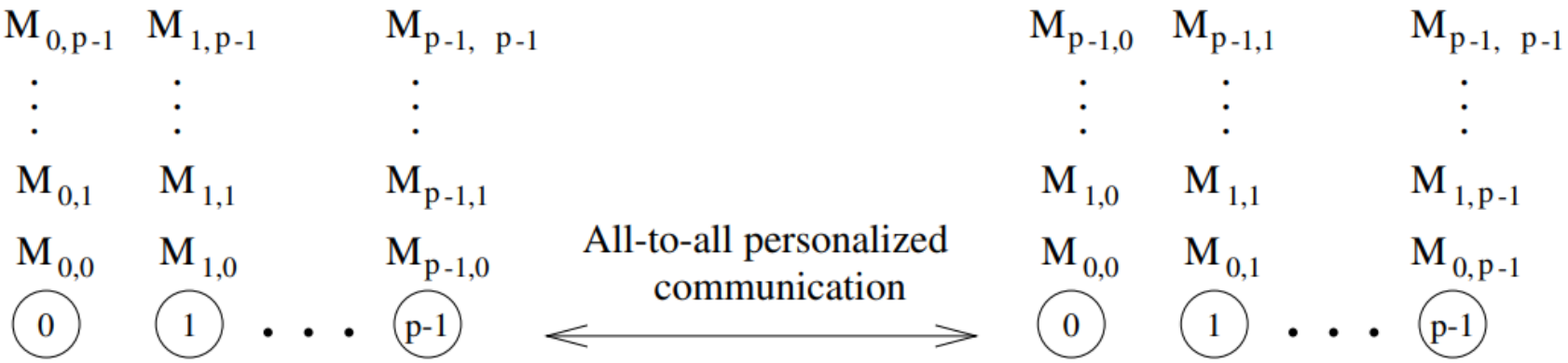




Step 3

Collective Communication

- All-to-all personalized communication



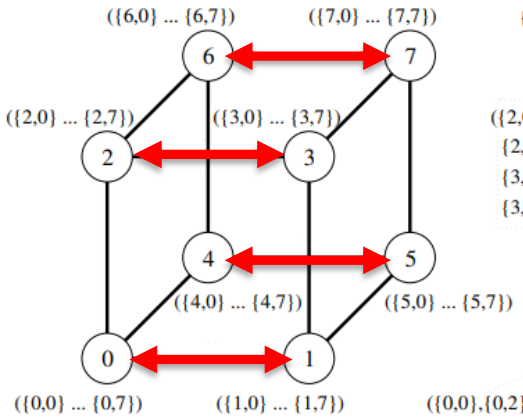
Effectively a matrix transpose operation

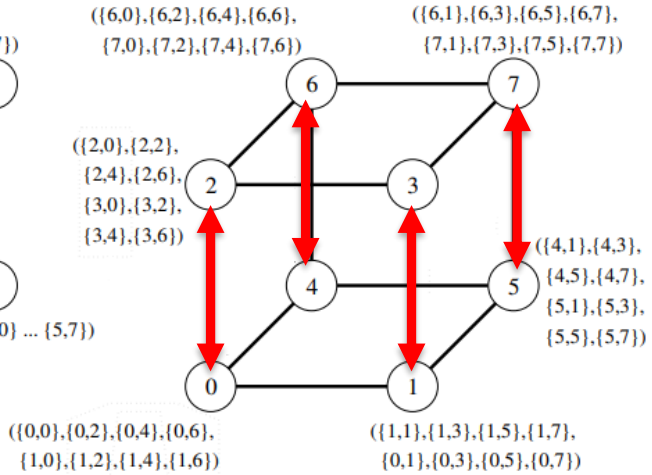
  - Implementation 1:
    - Using the same idea of all-to-all broadcast ($\log P$ steps).
  - Implementation 2:
    - Direct communication ($P - 1$ steps without congestion).

Collective Communication

- Implementation 1: All-to-all personalized communication **on hypercube** ($\log P$ steps)

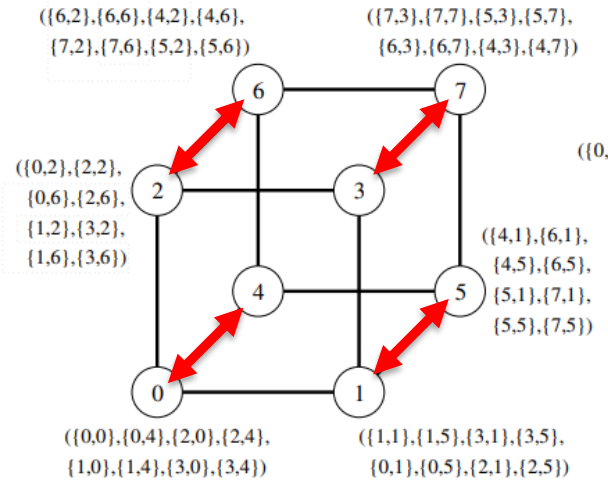
Step 0

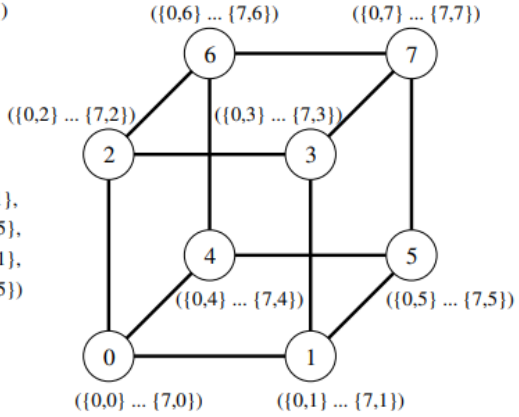




Step 1

Step 2

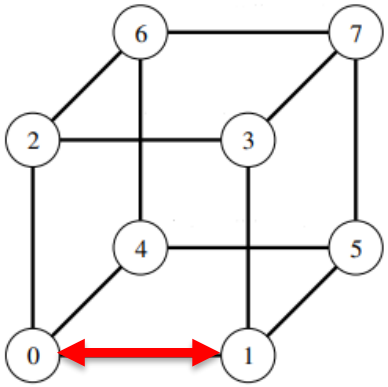




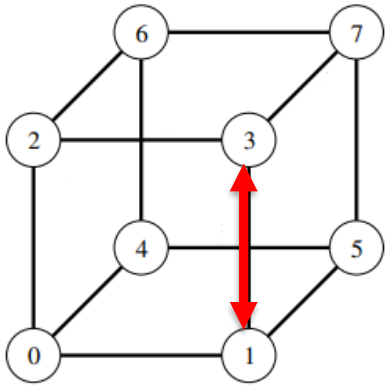
Step 3

Collective Communication

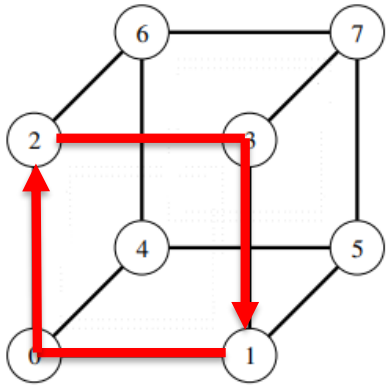
- Implementation 2: All-to-all personalized communication **on hypercube** ($P - 1$ steps)



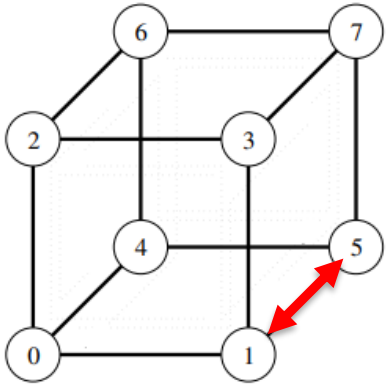
Step 1 (001)



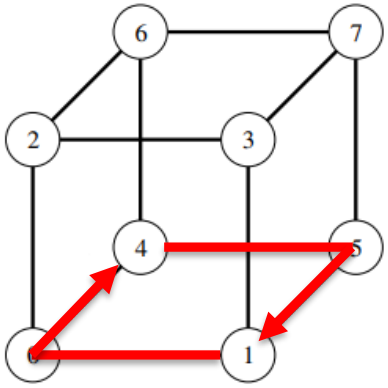
Step 2 (010)



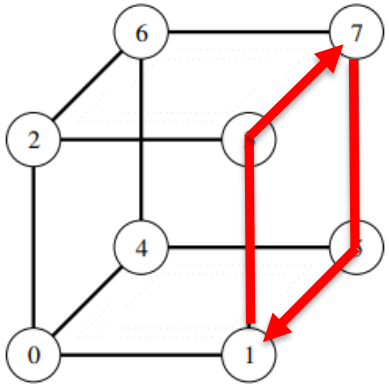
Step 3 (011)



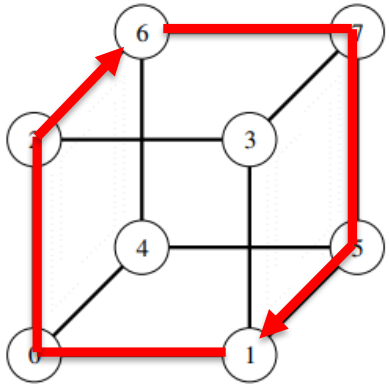
Step 4 (100)



Step 5 (101)



Step 6 (110)



Step 7 (111)

Rule: *At jth step (j=1..P-1), node i exchanges one message with node (i XOR j) via E-cube routing.* ***Path involving node 1 at each step is highlighted.***

MPI Names of Different Collective Communication Operations

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

References

- A. Grama, A. Gupta, G. Karypis, V. Kumar. Introduction to Parallel Computing (2nd Edition). *Addison-Wesley Professional.*
- B. Barney. Introduction to Parallel Computing Tutorial, Lawrence Livermore National Laboratory. https://computing.llnl.gov/tutorials/parallel_comp/