Parallel and Distributed Computing
CS3006

Lecture 18 - (BCS-6C/6D)

Instructor: Dr. Syed Mohammad Irteza

Assistant Professor, FAST School of Computing

30 March, 2023

Previous Lecture

- Basic Communication Operations
  - All-to-All Broadcast, All-to-All Reduction
    - Over hypercubes
    - Their Cost Estimates
  - All-Reduce
  - Prefix-Sum

Scatter and Gather

Scatter and Gather

- Gather is different from reduction as it doesn’t reduce the results with associative operator

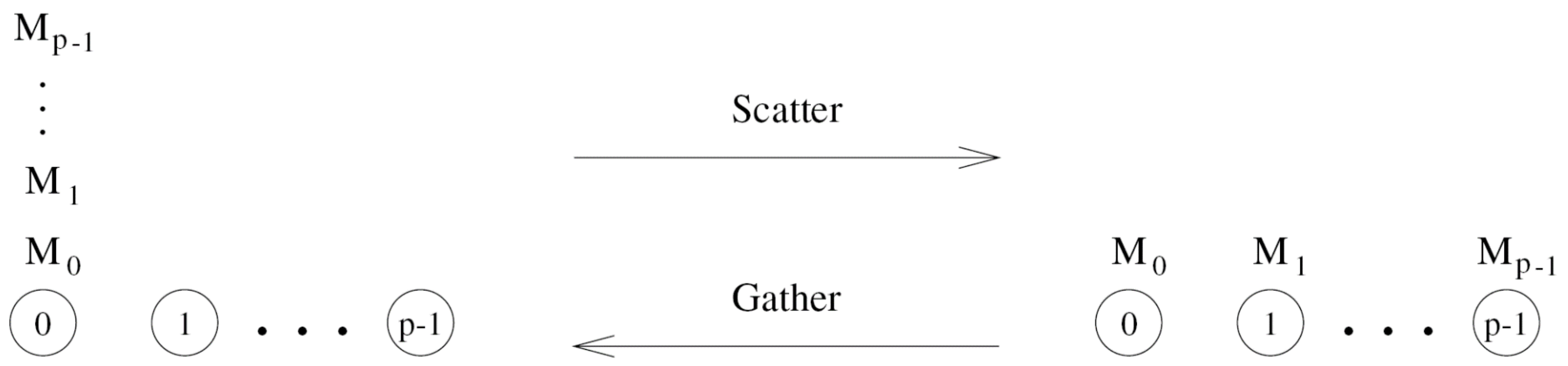
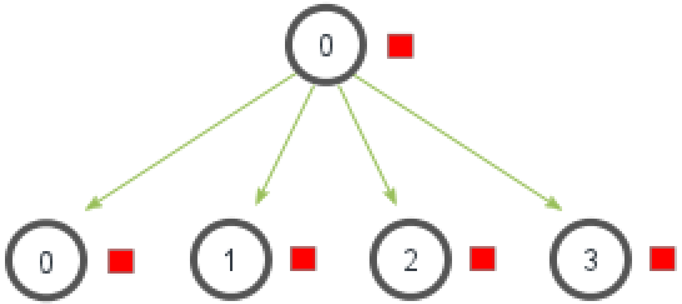


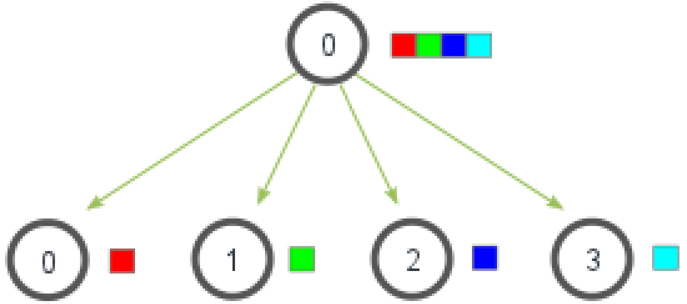
Figure 4.14 Scatter and gather operations.

Scatter and Gather, Allgather (MPI)

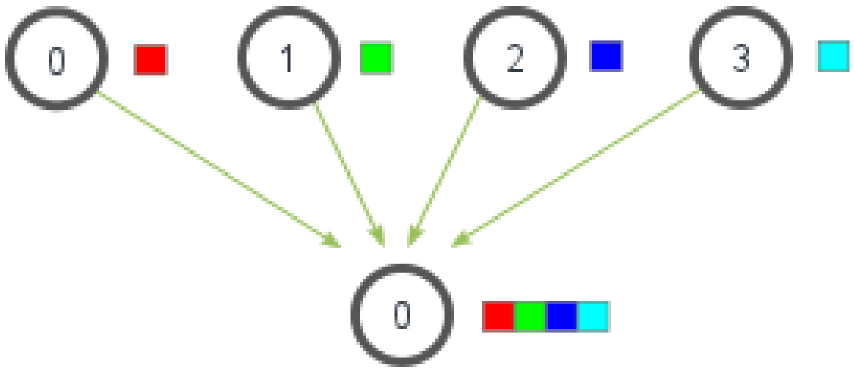
MPI_Bcast



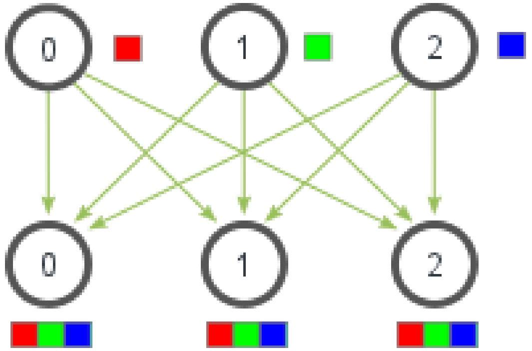
MPI_Scatter



MPI_Gather



MPI_Allgather

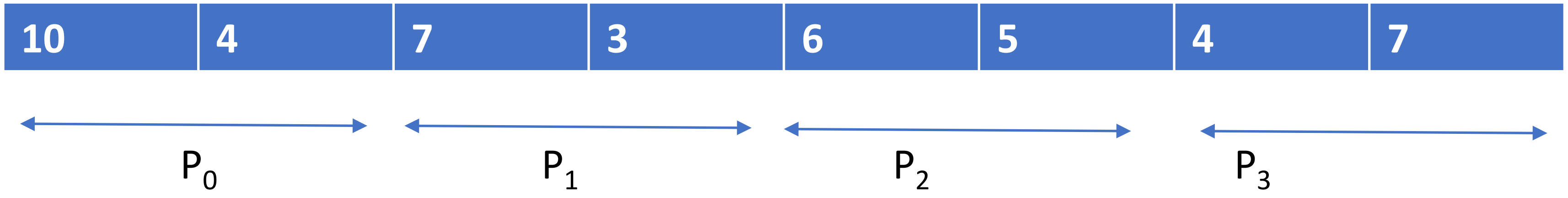


Using Scatter and Gather for Average

- P_0 has

10	4	7	3	6	5	4	7

- It uses scatter to send equal sized parts to all the processes



- Each process will now calculate the average on their local data
- Finally, each process will return its local average value through the gather function.
- Finally, P_0 will calculate the average of its 4 received local averages

Scatter and Gather

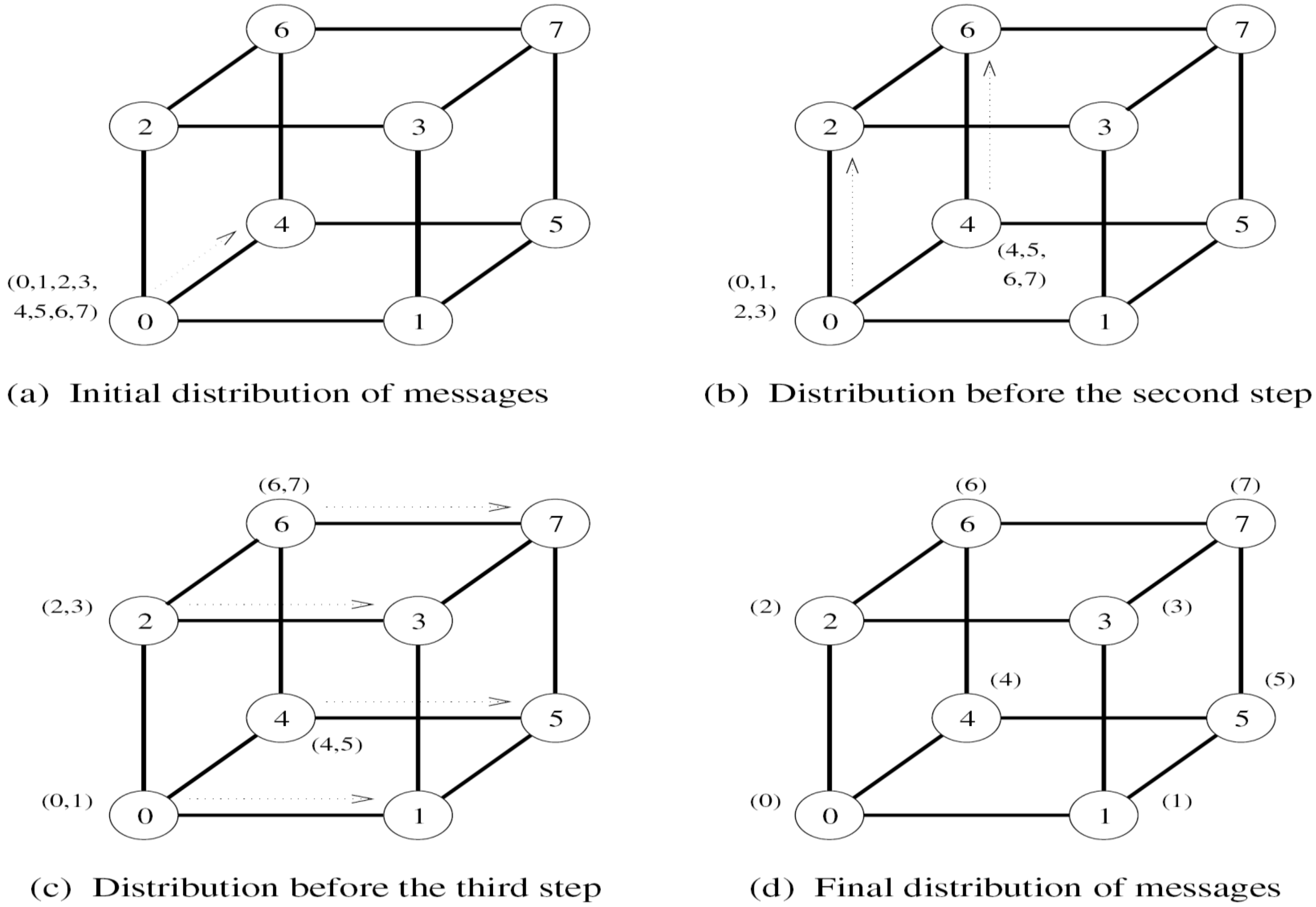


Figure 4.15 The scatter operation on an eight-node hypercube.

All-to-All personalized Communication

All-to-All personalized

- Each node sends a distinct message of size ***m*** to every other node.
- Also known as **total exchange**

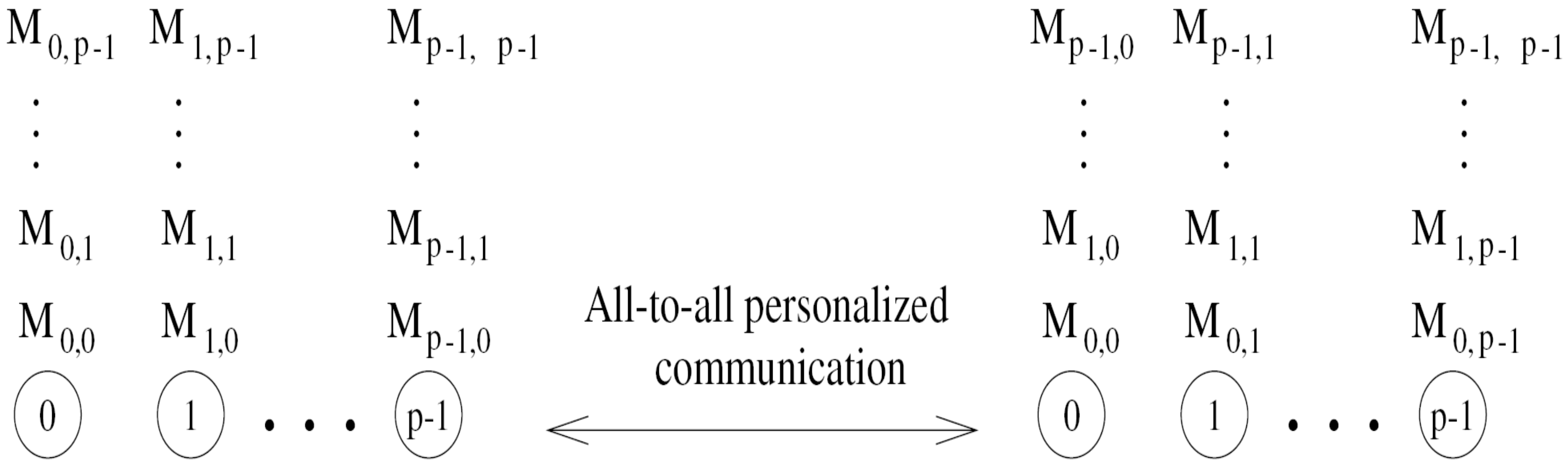
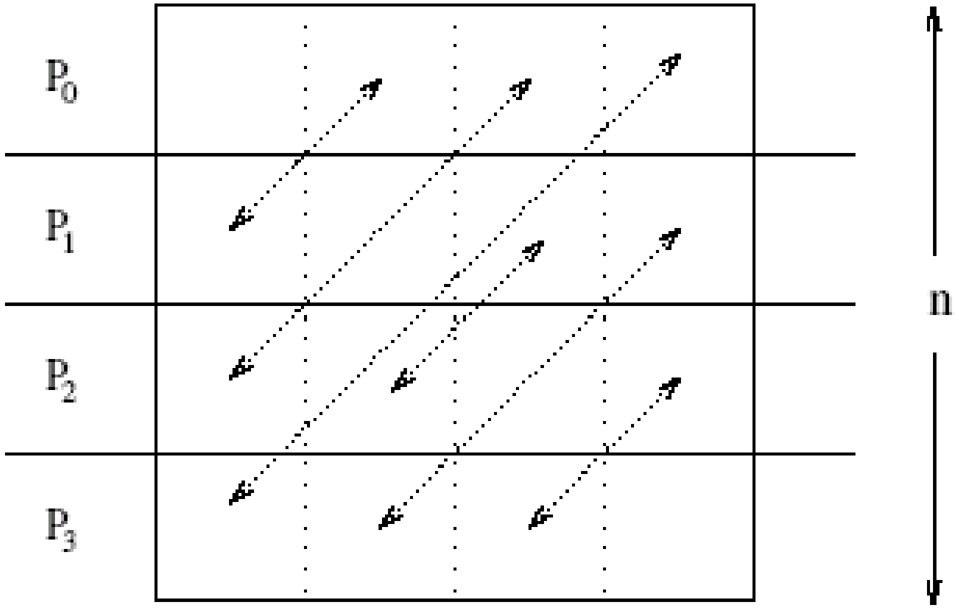


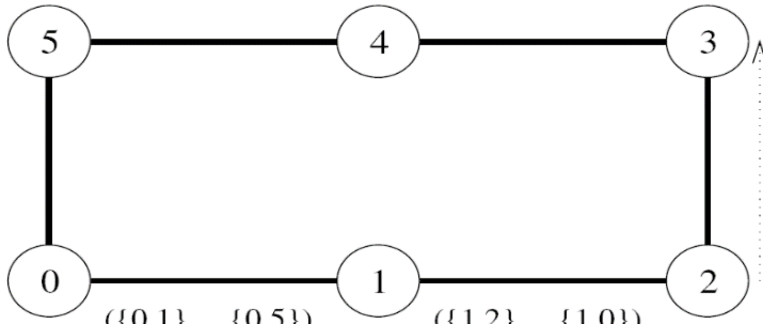
Figure 4.16 All-to-all personalized communication.

All-to-All personalized



All-to-all personalized communication in transposing a *4 x 4* matrix using four processes.

Basic Comm. Operations: (All-to-All personalized [Ring])



- First, each node sends all pieces of data as one consolidated message of size $m(p - 1)$ to one of its neighbors (all nodes communicate in the same direction).
- Of the $m(p - 1)$ words of data received by a node in this step, one m-word packet belongs to it. Therefore, each node extracts the information meant for it from the data received, and forwards the remaining $(p - 2)$ pieces of size m each to the next node.
- This process continues for $p - 1$ steps.
- The total size of data being transferred between nodes decreases by m words in each successive step. In every step, each node adds to its collection one m-word packet originating from a different node.
- Hence, in $p - 1$ steps, every node receives the information from all other nodes in the ensemble.

Basic Com. Operations: (All-to-All personalized [Ring])

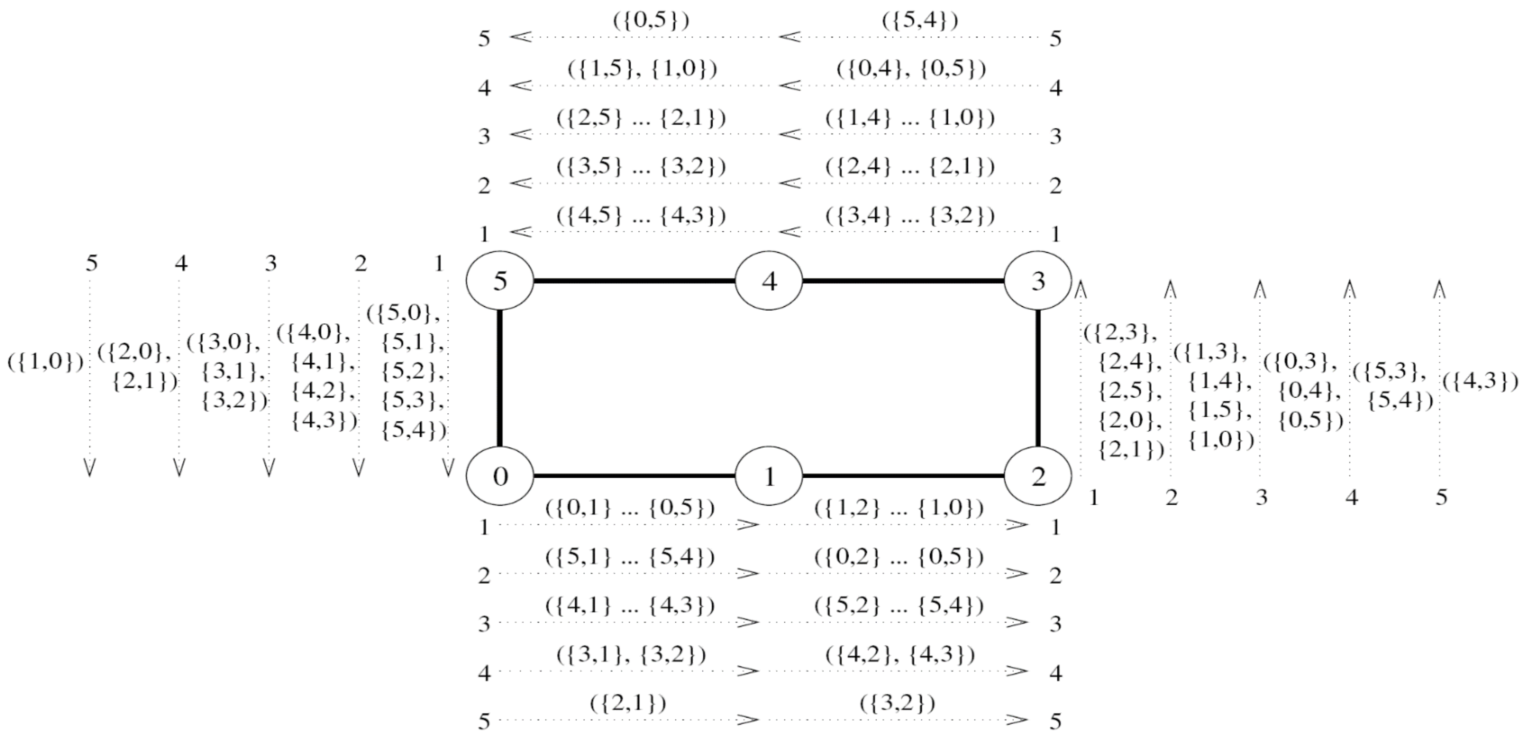


Figure 4.18 All-to-all personalized communication on a six-node ring. The label of each message is of the form $\{x, y\}$, where x is the label of the node that originally owned the message, and y is the label of the node that is the final destination of the message. The label $(\{x_1, y_1\}, \{x_2,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

All-to-All personalized [Ring]

Cost Analysis

- $T = \sum_{i=1}^{(p-1)} (t_s + (p-i)mt_w)$
  - $= \sum_{i=1}^{(p-1)} (t_s) + \; mt_w \sum_{i=1}^{(p-1)} (p-i)\ )$

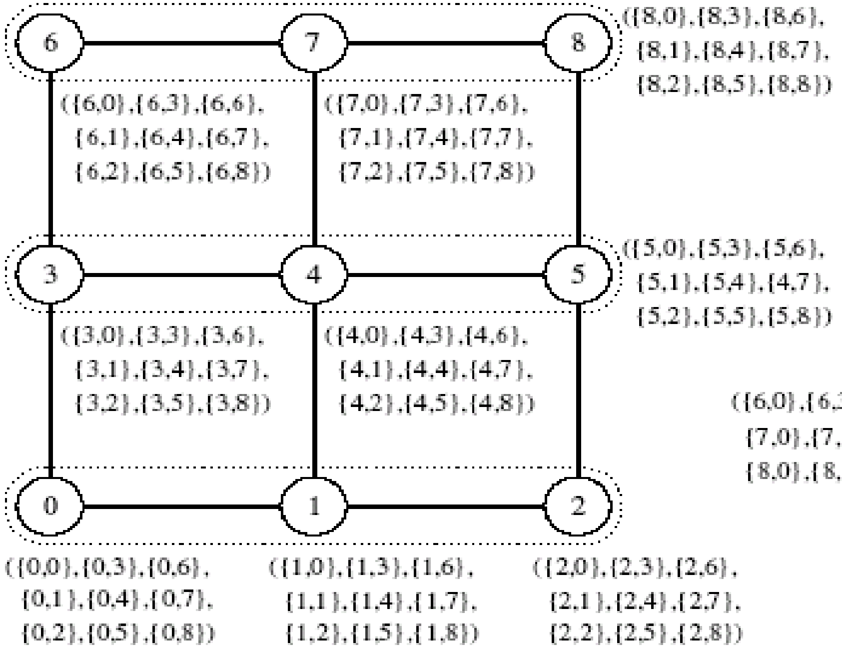
  $\rightarrow (\mathbf{p-1})(t_s) \quad + \quad mt_w \sum_{i=1}^{(p-1)} (\boldsymbol{i})$

  $\rightarrow \left((t_s + \left(\frac{\mathbf{1}}{\mathbf{2}}\right) \boldsymbol{pm} t_w \right) (p-1)$

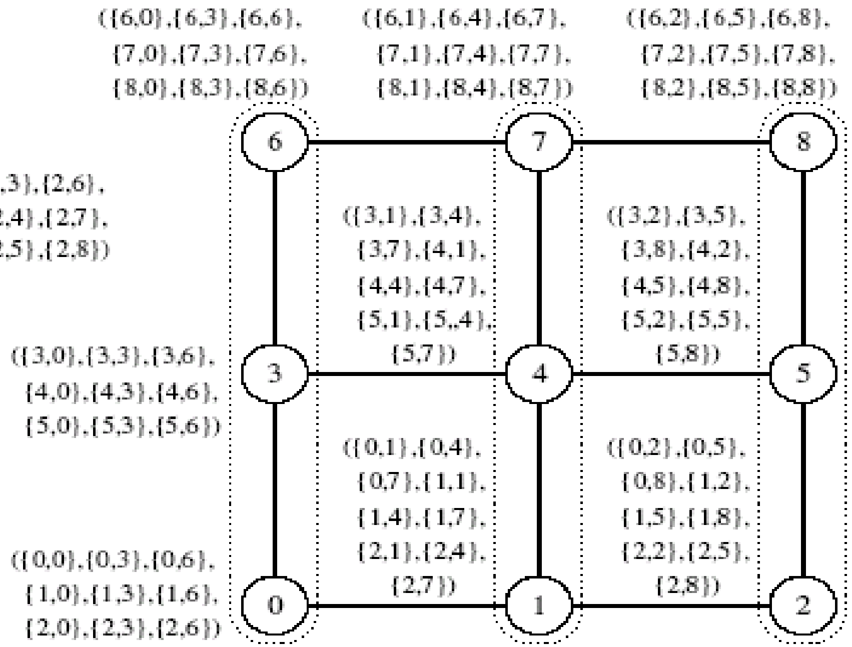
Basic Comm. Operations: (All-to-All personalized [Mesh, 3 X 3])

- In all-to-all personalized communication on a $\sqrt{p} * \sqrt{p}$ mesh, each node first groups its *p* messages according to the columns of their destination nodes.
- Figure 4.19 shows a 3 x 3 mesh, in which every node initially has nine *m*-word messages, one meant for each node.
- Each node assembles its data into three groups of three messages each (in general, $\sqrt{p}$ groups of $\sqrt{p}$ messages each).
- The first group contains the messages destined for nodes labeled 0, 3, and 6; the second group contains the messages for nodes labeled 1, 4, and 7; and the last group has messages for nodes labeled 2, 5, and 8.

All-to-All personalized [Mesh]



(a) Data distribution at the beginning of first phase



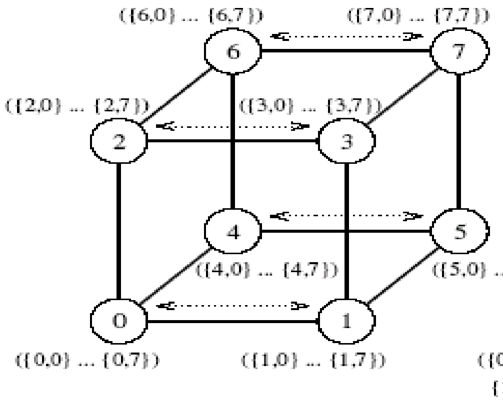
(b) Data distribution at the beginning of second phase

All-to-All personalized [Mesh]

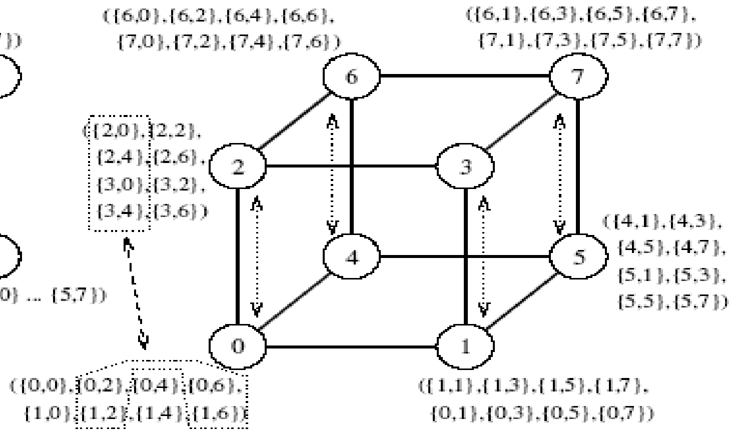
Cost Analysis

- Time for the first phase is identical to that in a ring with $\sqrt{p}$ processors, i.e., $(t_s + t_w mp/2)(\sqrt{p} - 1)$.
  - Here $\boldsymbol{m t_w}$ *becomes* $\boldsymbol{\sqrt{p}\, m t_w}$ and **P** becomes $\boldsymbol{\sqrt{p}}$
- Time in the second phase is identical to the first phase. Therefore, total time is twice of this time, i.e.,

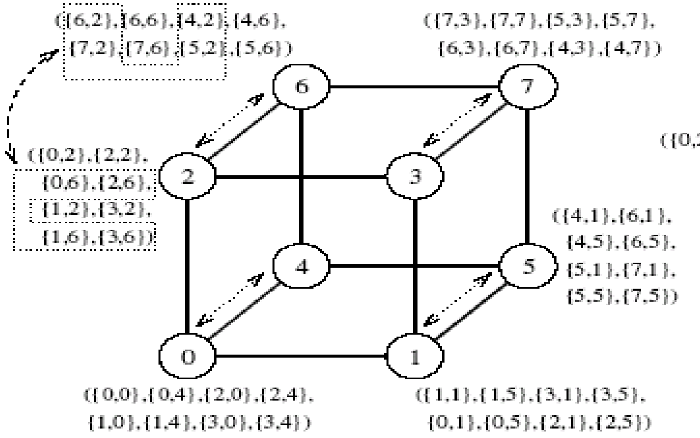
Basic Comm. Operations: (All-to-All personalized [Hyper Cube])



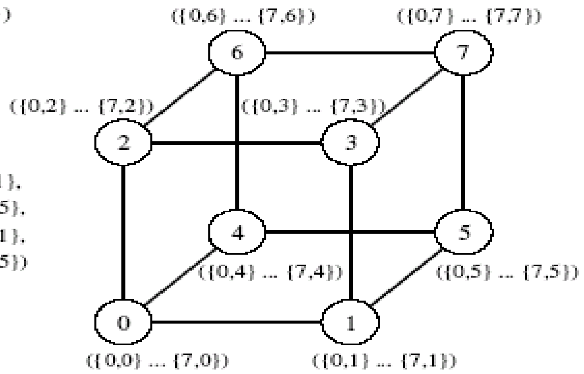
(a) Initial distribution of messages



(b) Distribution before the second step



(c) Distribution before the third step



(d) Final distribution of messages

Message Passing and MPI

Message Passing Paradigm

Programming Using the message passing paradigm:

- Oldest and most widely used approach for distributed programming.
- The logical view of a machine supporting the message-passing paradigm consists of p processes, each with its own exclusive address space.
- Most of the communication is done using simple send/receive message passing.

Message Passing Paradigm

Characteristics:

- Provides high scalability
- Complex to program
- High communication costs
- No support for incremental parallelism

Message Passing Interface (MPI)

- MPI defines a *standard library for message-passing* that can be used to develop portable message-passing programs using either C or Fortran.

- The MPI standard defines both the *syntax* as well as the *semantics* of a core set of library routines.

- It is possible to write fully-functional message-passing programs by using only the following six routines.

Message Passing Interface (MPI)

- The minimal set of MPI routines:

`MPI_Init`	Initializes MPI.
`MPI_Finalize`	Terminates MPI.
`MPI_Comm_size`	Determines the number of processes.
`MPI_Comm_rank`	Determines the label of calling process.
`MPI_Send`	Sends a message.
`MPI_Recv`	Receives a message.

Starting and Terminating the MPI Library

- `MPI_Init` is called prior to any calls to other MPI routines. Its purpose is to *initialize the MPI environment*.
- `MPI_Finalize` is called at the end of the computation, and it performs various *clean-up tasks to terminate the MPI environment*.
- The prototypes of these two functions are:

```
int MPI_Init(int *argc, char ***argv)

int MPI_Finalize()
```

- `MPI_Init` also strips off any MPI related *command-line arguments*.
- All MPI routines, data-types, and constants are prefixed by "`MPI_`". The *return code for successful completion* is `MPI_SUCCESS`.

Communicators

- A communicator defines a *communication domain*
  - a set of processes that can communicate with each other.
- Information about communication domains is stored in variables of type `MPI_Comm`.
- Communicators are used as arguments to all message transfer MPI routines.
- A *process can belong to many different* (possibly overlapping) communication domains.
- MPI defines a default communicator called `MPI_COMM_WORLD` which *includes all the processes*.

Querying Information

- The `MPI_Comm_size` and `MPI_Comm_rank` functions are used to determine the *number of processes* and the *label of the calling process*, respectively.
- The calling sequences of these routines are as follows:

```
int MPI_Comm_size(MPI_Comm comm, int *size)
int MPI_Comm_rank(MPI_Comm comm, int *rank)
```

- The rank of a process is an integer that ranges from zero up to the size of the communicator minus one.

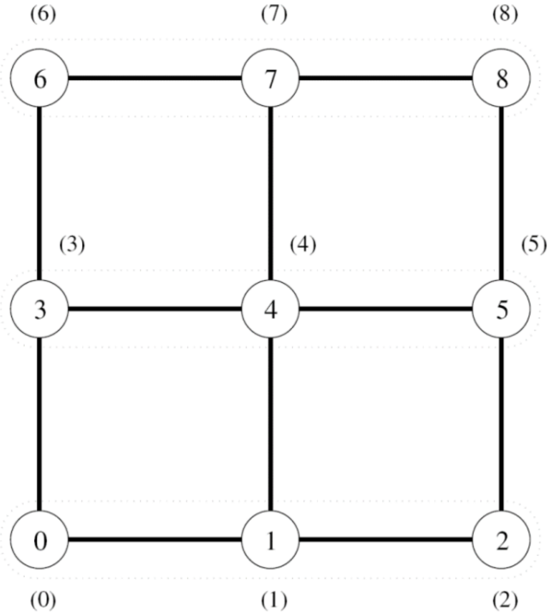
Hello World Program

```
#include <mpi.h>
main(int argc, char *argv[])
{
     int np, myrank;
     MPI_Init(&argc, &argv);
     MPI_Comm_size(MPI_COMM_WORLD, &np);
     MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
     printf("From process %d out of %d,   HelloWorld!\n",
myrank, np);
     MPI_Finalize();
}
```

References

1. Slides from Dr. Rana Asif Rehman & Dr. Haroon Mahmood
2. Kumar, V., Grama, A., Gupta, A., & Karypis, G. (1994). *Introduction to parallel computing* (Vol. 110). Redwood City, CA: Benjamin/Cummings.
3. Quinn, M. J. Parallel Programming in C with MPI and OpenMP, (2003).
4. https://mpitutorial.com/tutorials/mpi-scatter-gather-and-allgather/

Quiz-03 (CS-6D/DS-6A) (12 minutes)

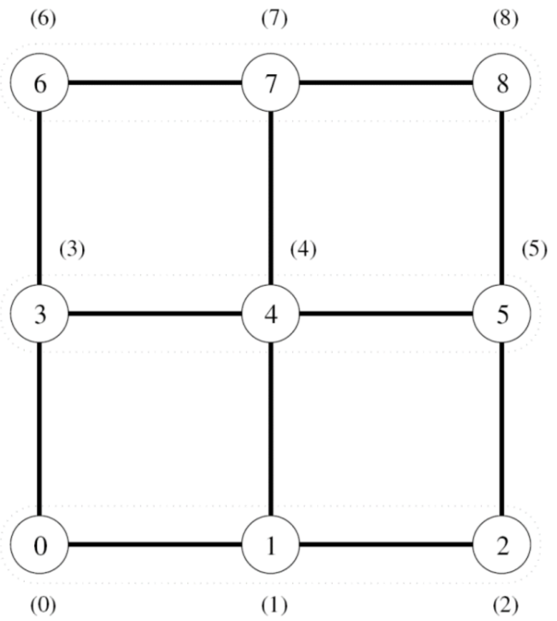


1) Apply an all-to-one reduction on the following 3*3 mesh, for node 0: [4m]

2) Explain how an all-to-all broadcast would work on a 4-node linear ring (assume the individual values are the last 4 digits of your roll number) [4m]

3) Provide one real world example of IaaS and one of SaaS [2m]

Quiz-03 (CS-6C) (12 minutes)



1) Apply a one-to-all broadcast on the following 3*3 mesh: [4m]

2) Explain how an all-to-all reduction (operation: sum) would work on a 4-node linear ring (assume the individual values are the last 4 digits of your roll number) [4m]

3) Provide one real world example of PaaS and one of SaaS [2m]