Parallel and Distributed Computing
CS3006 (BCS-6C/6D)
Lecture 16

Instructor: Dr. Syed Mohammad Irteza

Assistant Professor, Department of Computer Science, FAST

21 March, 2023

Previous Lecture

- Cloud Computing
  - Components (Hypervisor, Management Software, etc.)
  - Cloud Service Models (IaaS, PaaS, SaaS)
  - Types of clouds (public, private, community, hybrid)
  - Virtualization
- Basic Communication Operations
  - Some assumptions
  - 1-to-all Broadcast
  - All-to-1 Reduction
    - Under linear array, ring, mesh, hypercube
    - Using naïve method and recursive doubling

Basic Communication Operations (One-to-All Broadcast and All-to-One Reduction)

- Balanced Binary Tree

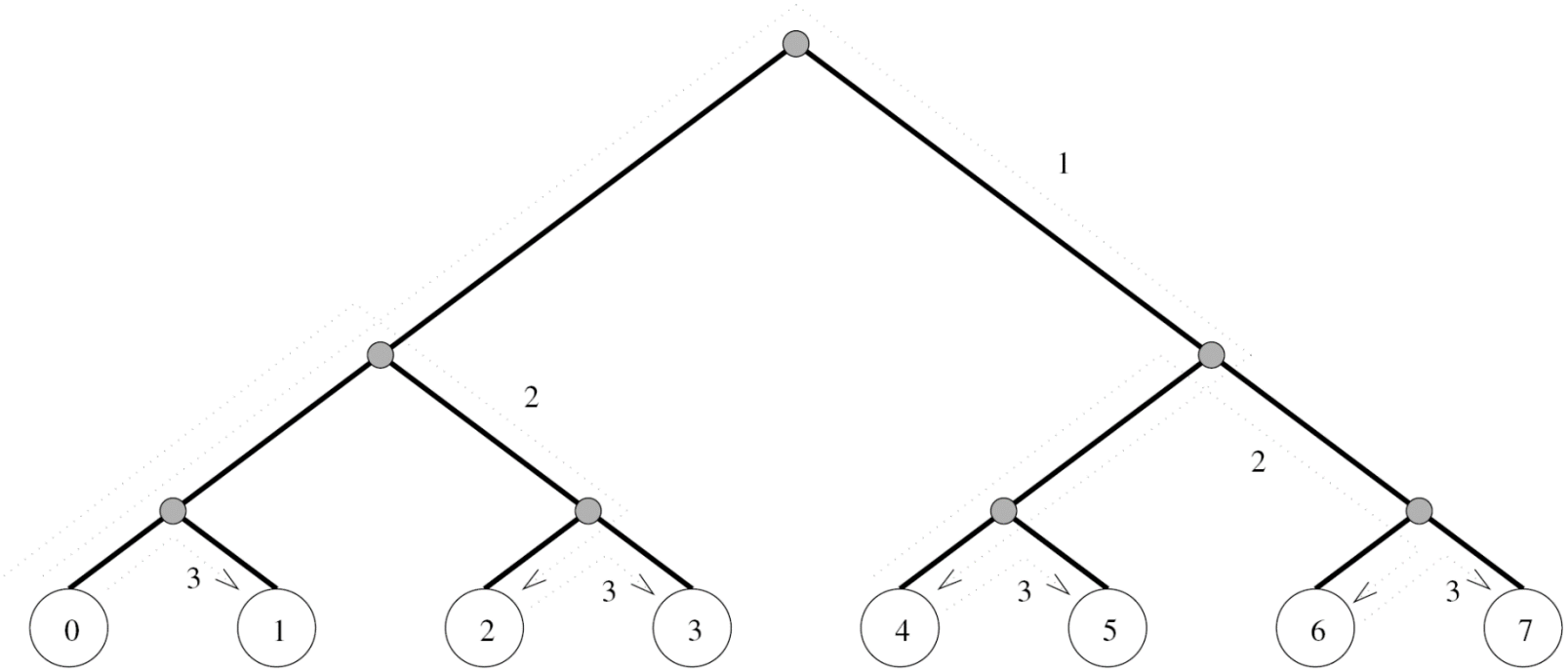


Figure 4.7 One-to-all broadcast on an eight-node tree.

Basic Communication Operations (One-to-All Broadcast and All-to-One Reduction)

```
1.  procedure ONE_TO_ALL_BC(d, my_id, X)
2.  begin
3.      mask := 2^d − 1;                    /* Set all d bits of mask to 1 */
4.      for i := d − 1 downto 0 do          /* Outer loop */
5.          mask := mask XOR 2^i;           /* Set bit i of mask to 0 */
6.          if (my_id AND mask) = 0 then    /* If lower i bits of my_id are 0 */
7.              if (my_id AND 2^i) = 0 then
8.                  msg_destination := my_id XOR 2^i;
9.                  send X to msg_destination;
10.             else
11.                 msg_source := my_id XOR 2^i;
12.                 receive X from msg_source;
13.             endelse;
14.         endif;
15.     endfor;
16. 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.

Please read the book (chapter 4)

Basic Communication Operations (One-to-All Broadcast and All-to-One Reduction)

```
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;
5.      for i := d − 1 downto 0 do      /* Outer loop */
6.          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
```

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.

Please read the book (chapter 4)

Basic Communication Operations (One-to-All Broadcast and All-to-One Reduction)

```
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
8.                    msg_destination := my_id XOR 2^i;
9.                    send sum to msg_destination;
10.               else
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
```

Please read the book (chapter 4)

Basic Communication Operations (One-to-All Broadcast and All-to-One Reduction)

- **Cost Estimation**
  - Broadcast needs **log(p)** point-to-point simple message transfer steps.
  - Message size of each transfer is **m**
  - Time for each of the transfers is: $\boldsymbol{t_s + mt_w}$

Hence cost for log(p) transfers→ T = $(\boldsymbol{t_s + mt_w}) \log \boldsymbol{p}$

All-to-All Broadcast and All-to-All Reduction

All-to-All Broadcast

- A generalization of one-to-all broadcast.
- Every process broadcasts an *m-word* message.
  - The broadcast-message for each of the processes can be *different* from others

All-to-All Reduction

- Dual of all-to-all broadcast
- Each node is the destination of an all-to-one reduction out of total P reductions.

All-to-All Broadcast and All-to-All Reduction

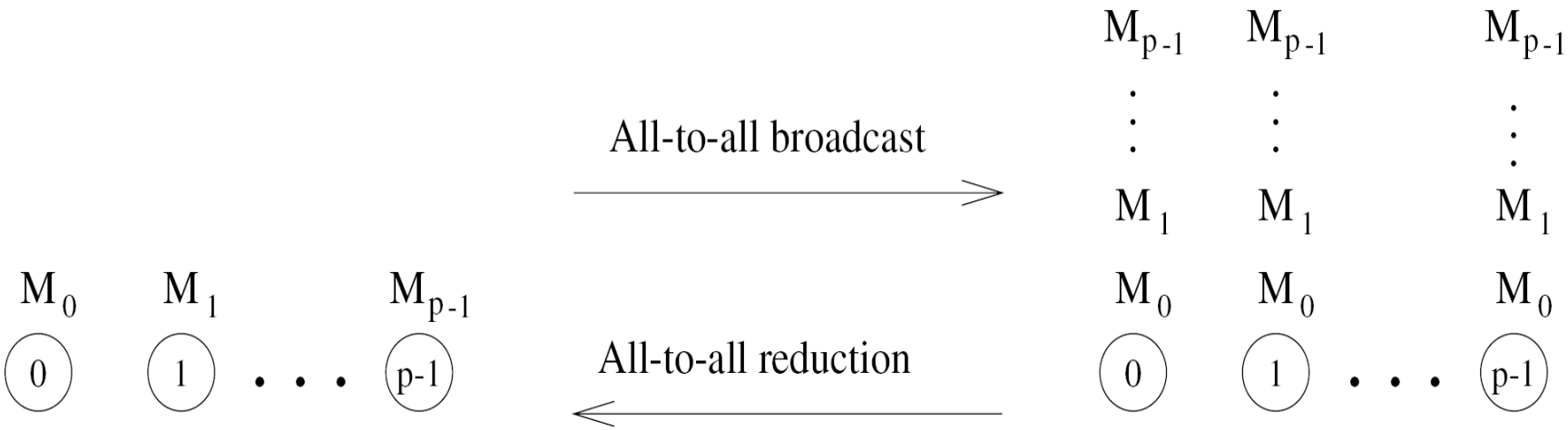


Figure 4.8 All-to-all broadcast and all-to-all reduction.

A naïve Broadcast method may be performing **P** one-to-all broadcasts. This will result **P(log(p)(t(s) + mt(w)))** communication time.
Solution?

All-to-All Broadcast and All-to-All Reduction

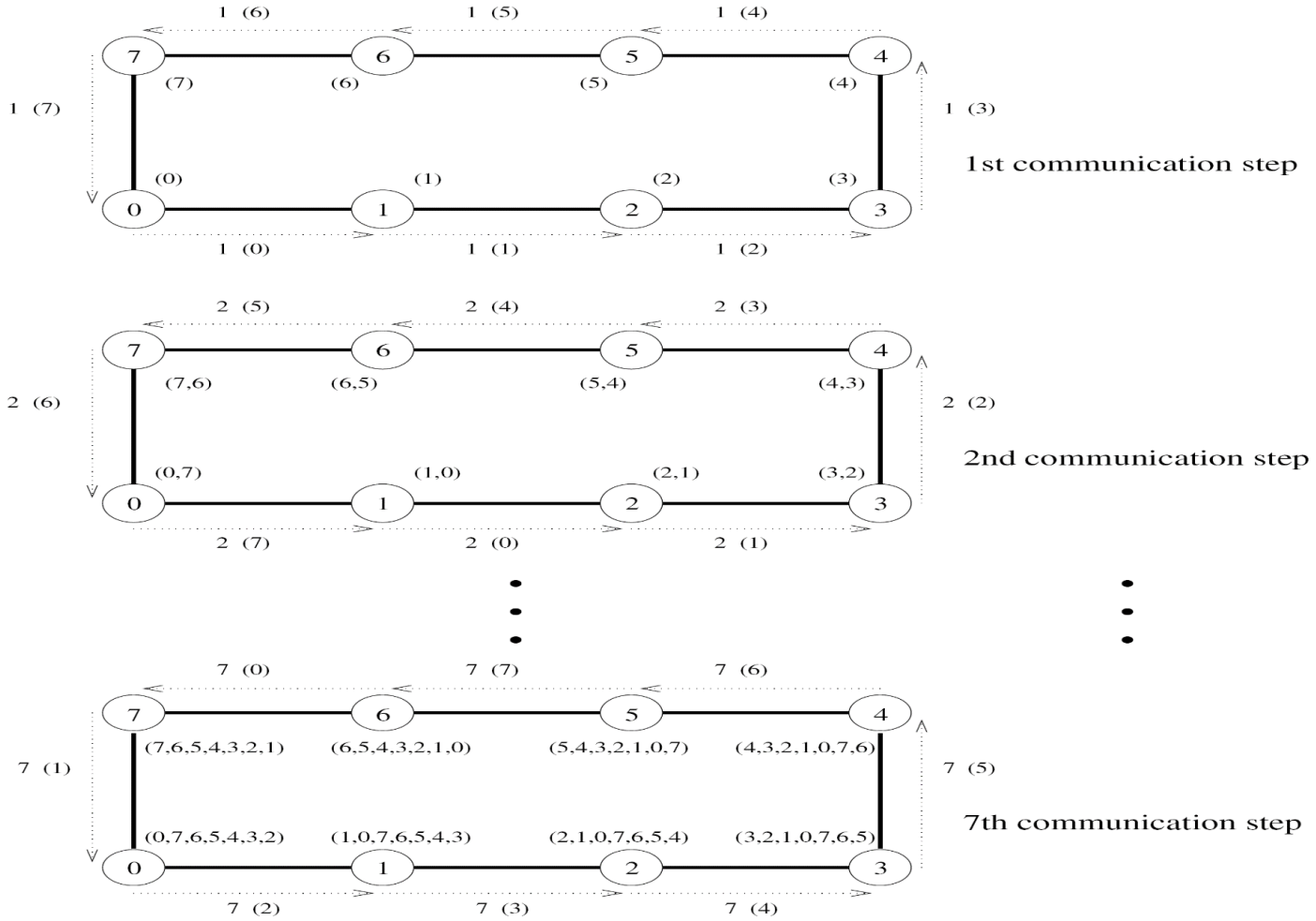


Figure 4.9 All-to-all broadcast on an eight-node ring. The label of each arrow shows the time step and, within parentheses, the label of the node that owned the current message being transferred before the beginning of the broadcast. The number(s) in parentheses next to each node are the labels of nodes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

Linear Ring Broadcast

```
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
```

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

All-to-All Reduction

Linear Array or Ring

- **Reduction**
  - Draw an All-to-All Broadcast on a P-node linear ring
  - Reverse the directions in each foreach of the step without changing message
  - After each communication step, combine messages having same broadcast destination with associative operator.

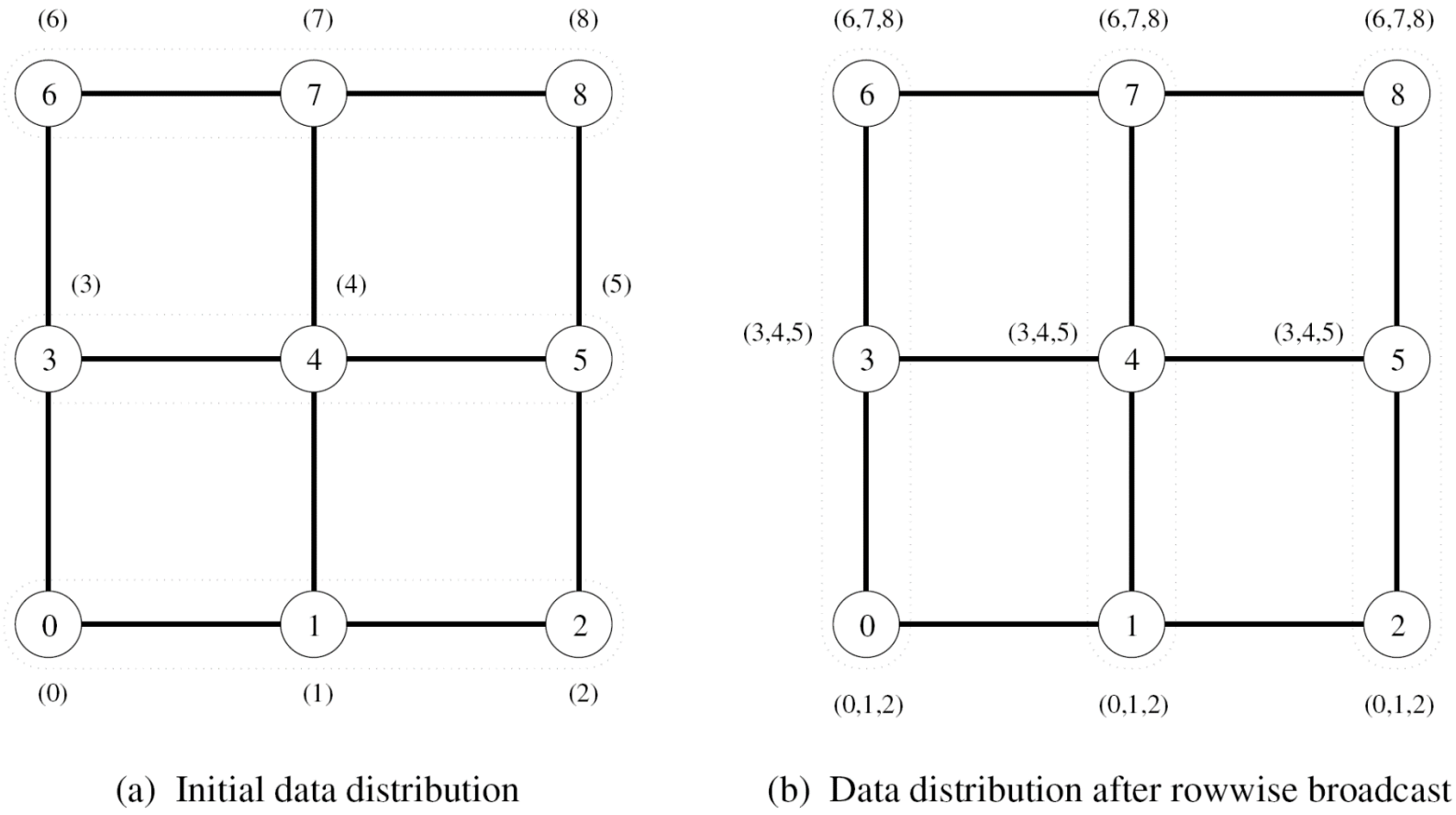
- **Now, Its your turn to draw?**
  - **Draw an All-to-All Broadcast on a 4-node linear ring**
  - **Reverse the directions and combine the results using ‘SUM’**

Linear Ring Reduction

```
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
```

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

All-to-All Broadcast on 2D 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).

All-to-All Broadcast on 2D Mesh Algorithm

```
1.    procedure ALL_TO_ALL_BC_MESH(my_id, my_msg, p, result)
2.    begin

/* Communication along rows */
3.        left := my_id − (my_id mod √p) + (my_id − 1) mod √p;
4.        right := my_id − (my_id mod √p) + (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;

/* Communication along columns */
12.       up := (my_id − √p) mod p;
13.       down := (my_id + √p) mod p;
14.       msg := result;
15.       for i := 1 to √p − 1 do
16.           send msg to down;
17.           receive msg from up;
18.           result := result ∪ msg;
19.       endfor;
20.   end ALL_TO_ALL_BC_MESH
```

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

Useful Links

- https://www.cs.unc.edu/~prins/Classes/633/Readings/Kumar-BasicCommunicationOperations.pdf
- https://phyweb.physics.nus.edu.sg/~phytaysc/cz4102_07/cz4102_le6.pdf
- http://www.math.nsysu.edu.tw/~lam/MPI/lecture/chap4_slides.pdf