Parallel and Distributed Computing

CS3006

Lecture 11

Basic Communication Operations

18th March 2024

Dr. Rana Asif Rehman

Basic Communication Operations

Basic Communication Operations

Preliminaries

- Exchanging the data is fundamental requirement for most of the parallel algorithms
- $\boldsymbol{t_s + mt_w}$ - the simplified communication cost model :-
  - Over distributed memory infrastructure
  - Assuming the cut-through routing
- The chapter is about commonly used basic communication patterns over the different interconnections
  - We shall drive communication costs of these operations on different interconnections.

Basic Communication Operations

Assumptions for the Operations

- Interconnections support cut-through routing
- Communication time between any pair of nodes in the network is same (regardless of the number of intermediate nodes)
- Links are bi-directional
  - The directly connected nodes can simultaneously send messages of *m words* without any congestion
- Single-port communication model
  - A node can send on only one of its links at a time
  - A node can receive on only one of its links at a time
- However, a node can receive a message while sending another message at the same time on the same or a different link.
- Consider $\mathbf{p=2}^{i}$ *nodes*

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

Basic Communication Operations

(One-to-All Broadcast and All-to-One Reduction)

One-to-All Broadcast

- A single process sends identical data to all other processes.
  - Initially one process has data of *m* size.
  - After broadcast operation, each of the processes have own copy of the *m* size.

All-to-One Reduction

- Dual of one-to-all broadcast
- The *m-sized* data from all processes are combined through an associative operator
- accumulated at a single destination process into one buffer of size *m*

Basic Communication Operations

(One-to-All Broadcast and All-to-One Reduction)

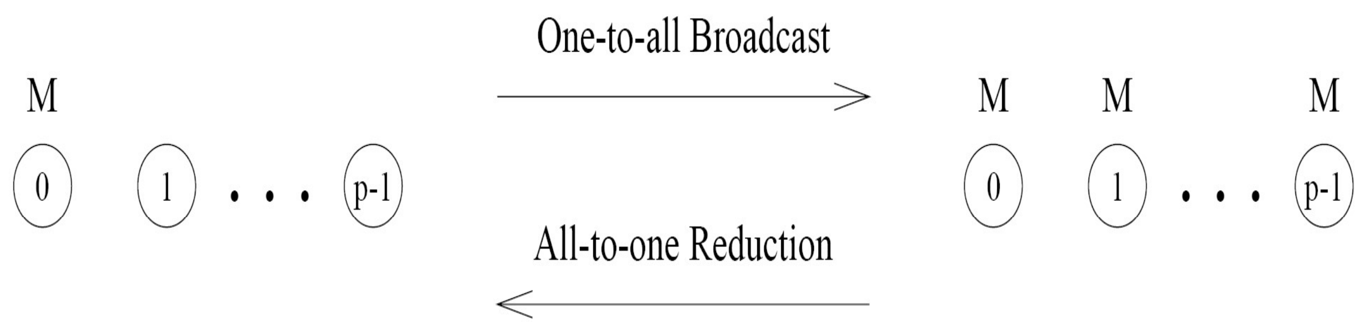

Figure 4.1 One-to-all broadcast and all-to-one reduction.

Basic Communication Operations

(One-to-All Broadcast and All-to-One Reduction)

Linear Array or Ring

- Naïve solution
  - sequentially send p - 1 messages from the source to the other p - 1 processes
    - Bottle necks, and underutilization of communication network
  - Solution?

- Recursive doubling
  - Source process sends the massage to another process
  - In next communication phase both the processes can simultaneously propagate the message

Basic Communication Operations

(One-to-All Broadcast and All-to-One Reduction)

Linear Array or Ring

- Recursive Doubling Broadcast

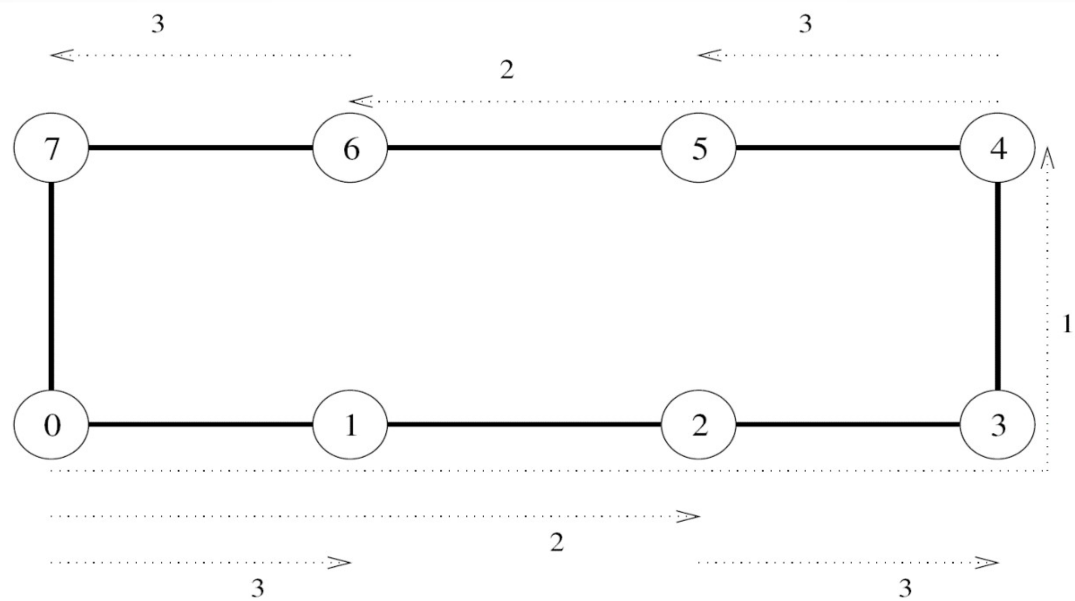

Figure 4.2 One-to-all broadcast on an eight-node ring. Node 0 is the source of the broadcast. Each message transfer step is shown by a numbered, dotted arrow from the source of the message to its destination. The number on an arrow indicates the time step during which the message is transferred.

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

Linear Array or Ring

- Recursive Doubling Reduction

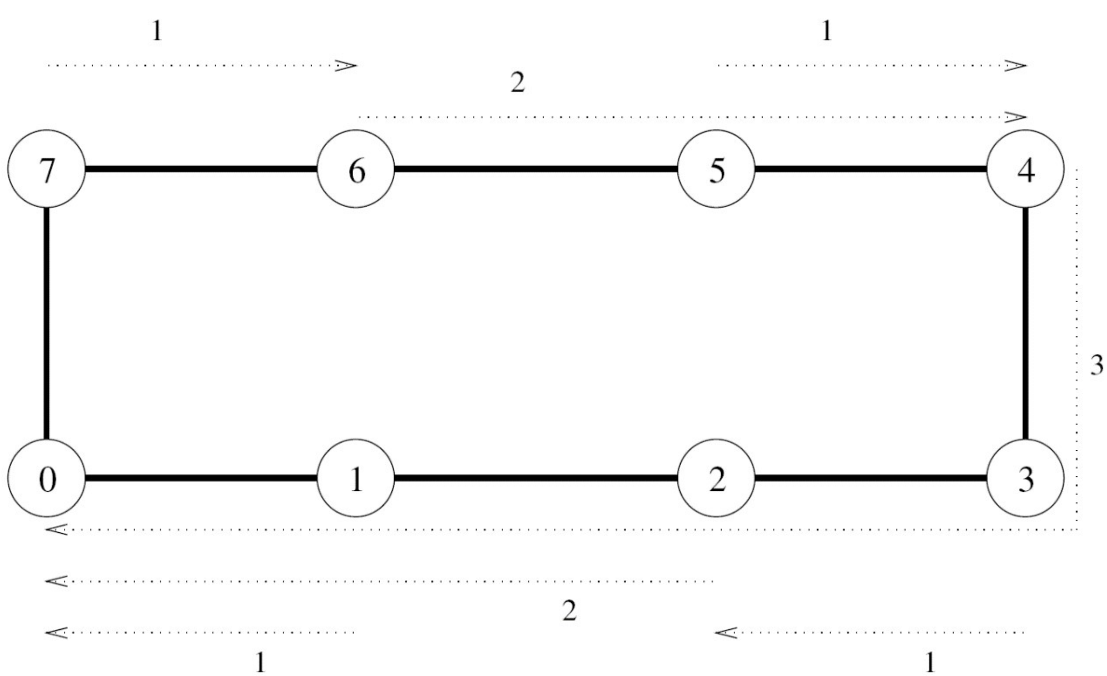

Figure 4.3 Reduction on an eight-node ring with node 0 as the destination of the reduction.

Basic Communication Operations

(One-to-All Broadcast and All-to-One Reduction)

Matrix-Vector Multiplication (An Application)

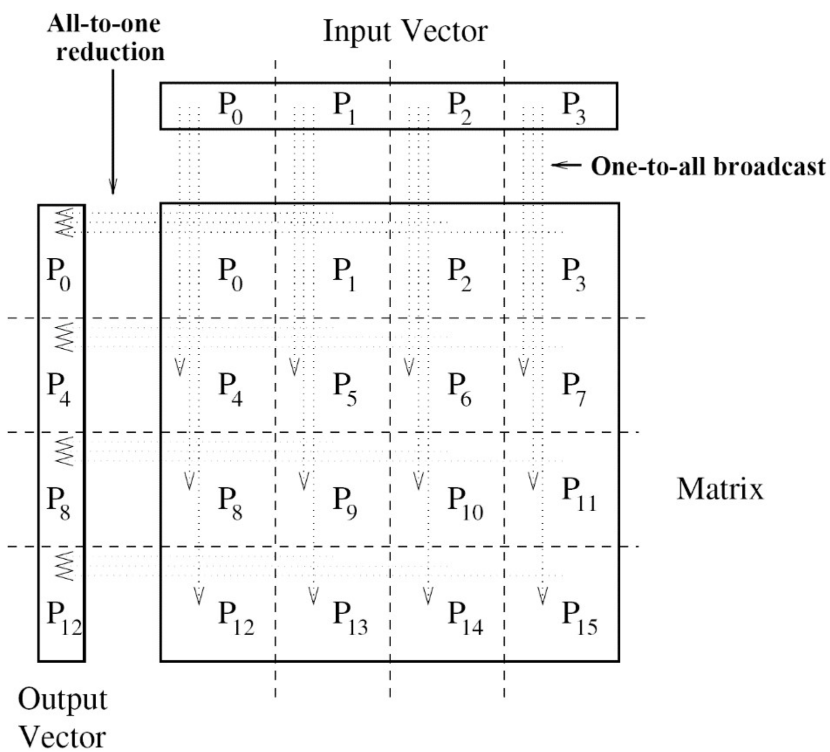

Figure 4.4 One-to-all broadcast and all-to-one reduction in the multiplication of a 4×4 matrix with a 4×1 vector.

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

Mesh

- We can regard each row and column of a square mesh of *p* nodes as a linear array of nodes
- Communication algorithms on the mesh are simple extensions of their linear array counterparts

- **Broadcast and Reduction**
  - Two step breakdown:
    I. The operation is performed along one by treating the row as linear array
    II. Then the all the columns are treated similarly

Basic Communication Operations

(One-to-All Broadcast and All-to-One Reduction)

Mesh (Broadcast and Reduction)

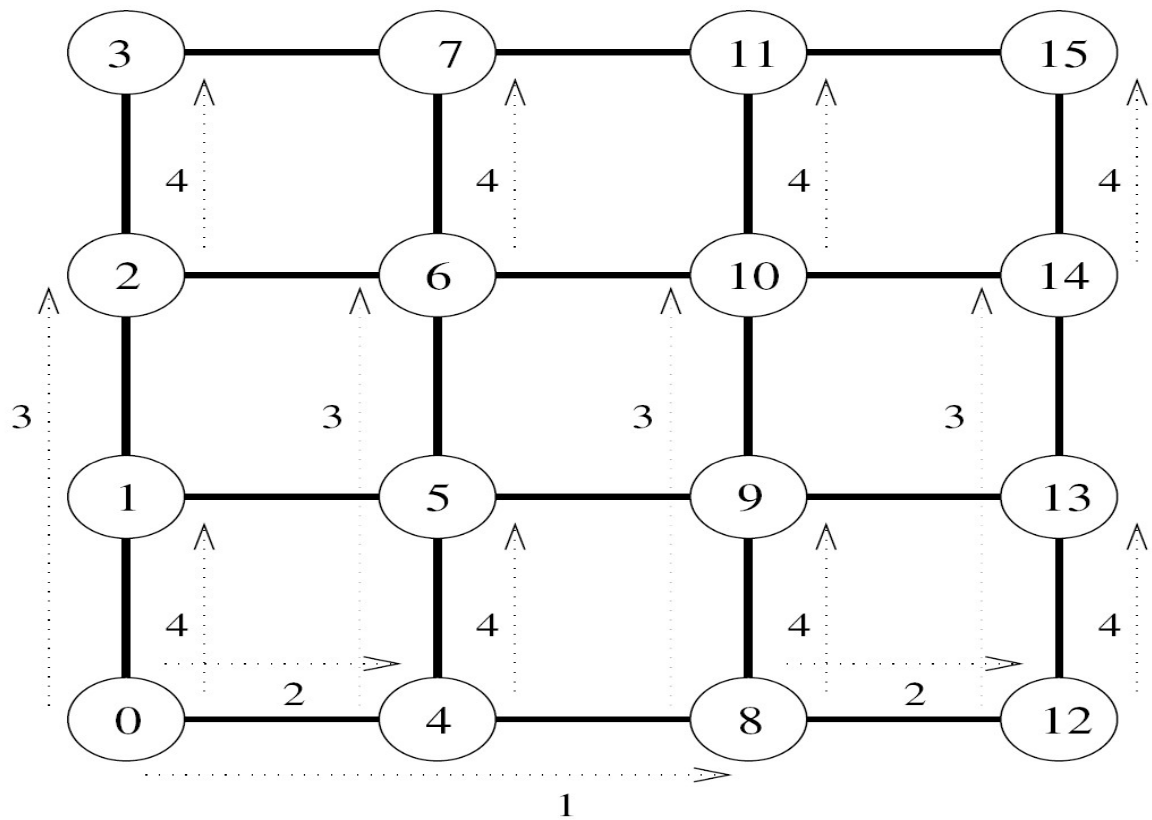

Figure 4.5 One-to-all broadcast on a 16-node mesh.

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

Balanced Binary Tree

- Broadcast

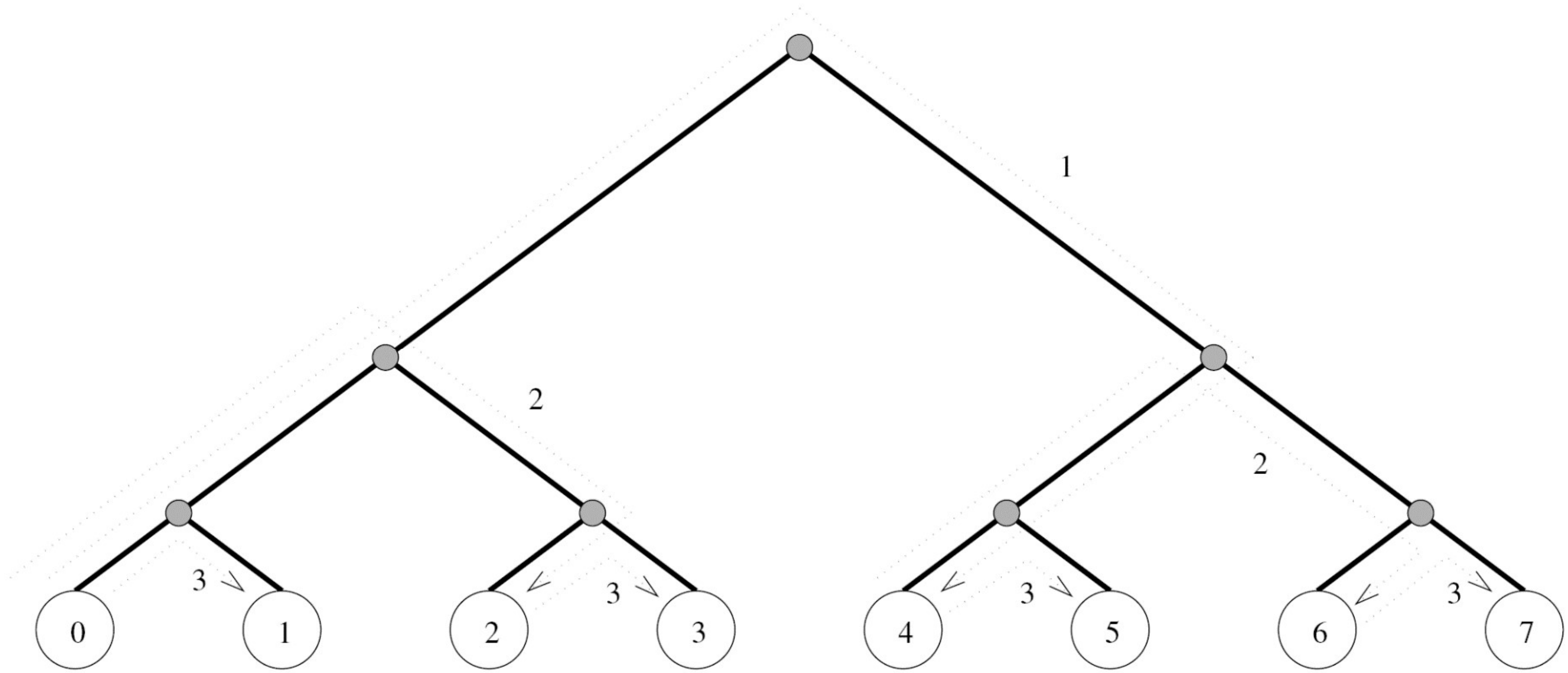

Figure 4.7 One-to-all broadcast on an eight-node tree.

Basic Communication Operations

(One-to-All Broadcast and All-to-One Reduction)

Hypercube

- Broadcast
  - Source node first send data to one node in the highest dimension
  - The communication successively proceeds along lower dimensions in the subsequent steps
  - The algorithm is same as used for linear array
    - But, here changing order of dimension does not congest the network

Basic Communication Operations

(One-to-All Broadcast and All-to-One Reduction)

Hypercube

- Broadcast

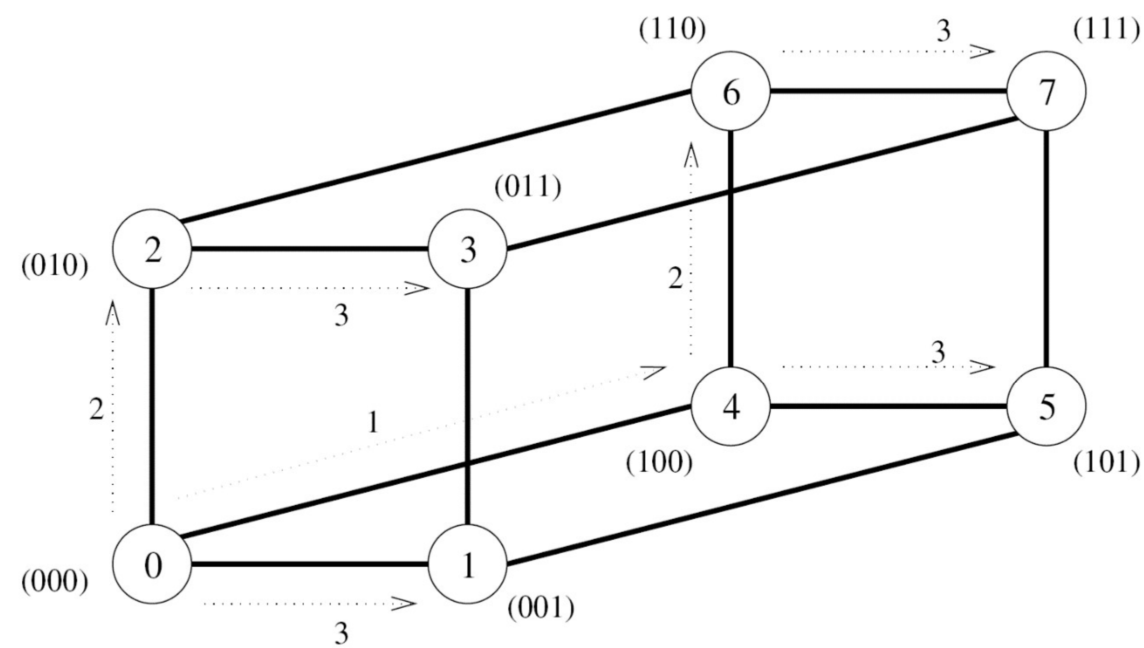

Figure 4.6 One-to-all broadcast on a three-dimensional hypercube. The binary representations of node labels are shown in parentheses.

Basic Communication Operations

(One-to-All Broadcast and All-to-One Reduction)

```
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.

Basic Communication Operations

(One-to-All Broadcast and All-to-One Reduction)

```
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.

Basic Communication Operations

(One-to-All Broadcast and All-to-One Reduction)

```
procedure ALL_TO_ONE_REDUCE(d, my_id, m, X, sum)
begin
    for j := 0 to m - 1 do sum[j] := X[j];
    mask := 0;
    for i := 0 to d - 1 do
        /* Select nodes whose lower i bits are 0 */
        if (my_id AND mask) = 0 then
            if (my_id AND 2^i) ≠ 0 then
                msg_destination := my_id XOR 2^i;
                send sum to msg_destination;
            else
                msg_source := my_id XOR 2^i;
                receive X from msg_source;
                for j := 0 to m - 1 do
                    sum[j] :=sum[j] + X[j];
            endelse;
        mask := mask XOR 2^i;    /* Set bit i of mask to 1 */
    endfor;
end ALL_TO_ONE_REDUCE
```

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= $(\boldsymbol{t_s}+\boldsymbol{mt_w}) \log \boldsymbol{p}$

Questions

References

1. Kumar, V., Grama, A., Gupta, A., & Karypis, G. (1994). *Introduction to parallel computing* (Vol. 110). Redwood City, CA: Benjamin/Cummings.
2. Quinn, M. J. Parallel Programming in C with MPI and OpenMP,(2003).