UNIT III

CHAPTER 3

Basic Communication

Syllabus Topics

Operations - One-to-All Broadcast and All-to-One Reduction, All-to-All Broadcast and Reduction, All-Reduce and Prefix-Sum Operations, Scatter and Gather, All-to-All Personalized Communication, Circular Shift, Improving the Speed of Some Communication Operations.

Syllabus Topic : Operations - One-to-All Broadcast and All-to-one Reduction

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

Q. 3.1.1 Explain broadcast and reduce operation with the help of diagram. (Refer section 3.1) **(5 Marks)**

Most often parallel algorithm design contains a number of processes where one process may send the identical data to all other processes. In fact the one - to - all broadcast operation handles transfer of data to other processes.

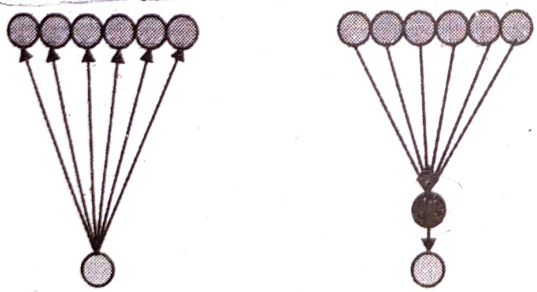


Fig. 3.1.1 : Broadcast and Reduce operations

- At the initial stage the data of size S to be broadcasted is contained in the source process.
- When the overall process gets terminated, all processes will contain the copy of the same data because of broadcasting.
- If assume there are N number of processes so N multiple copies of the initial data will be created.
- The reduction is also a type of broadcasting operation, in which one process in a group collects messages from all other processes and combine them to a single message.
- The combined message can be accessed by all other processes in the group. This way all - to - one reduction is the dual of one - to -all broadcast.
- Suppose **P** is the number of processes taking part in the operation with buffer **M** containing **w** words.
- Here an associative operation is used for combining the data from all processes and collected in one buffer of size **w** of a single destination process.
- The reduction operation can be used to find the sum, product, maximum, minimum of sets of numbers.
- The Fig. 3.1.2 shows one-to-all broadcast and all-to-one reduction among **p** processes.

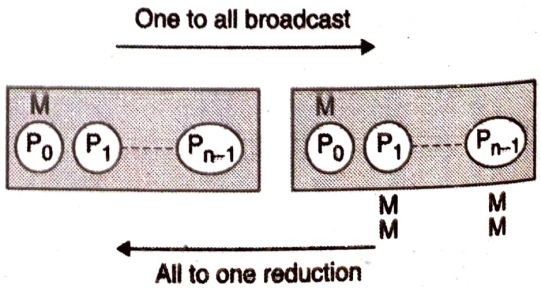


Fig. 3.1.2 : One - to - all broadcast and all - to - one reduction

Now we will deal with the implementation of one - to - all broadcasting using the different types of interconnection topologies.

3.1.1 Ring and Linear Array

- The approach to send P_1 messages from the source to rest of the P_1 processes may suffer with the bottleneck at source side.
- One important point is the underlying communication network is underutilized due to establish the connection only between a single pair of nodes at a time.
- The efficient utilization can be done by devising a broadcasting algorithm with the method known as **recursive doubling.**
- In this technique first the source process sends a message to another process, thereafter these two processes sends messages to two other processes waiting for the messages.
- This message sending operation is carried on till all processes received the message. In this way total **log p** stapes required for message broadcasting.

☞ One-to-All Broadcast on a Ring

Q. 3.1.2 Write short note on: One-to-All Broadcast on a Ring. (Refer section 3.1.1) **(5 Marks)**

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.

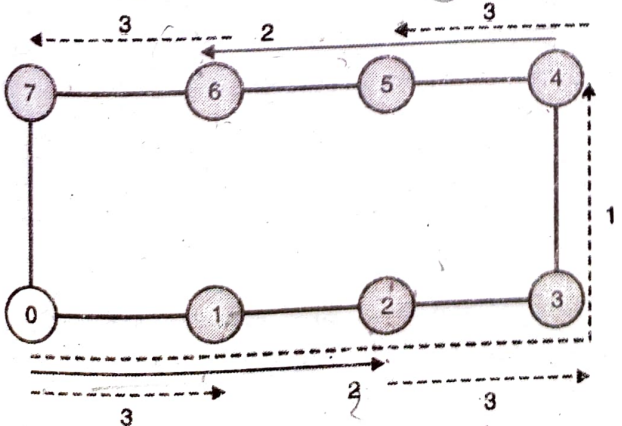


Fig. 3.1.3 : One to all broadcast on a ring

- It is important to carefully choose the destination node to which the message is destined on a linear array.
- In the broadcasting of messages almost care is required to avoid the congestion issues among the nodes.
- For example in the Fig. 3.1.3 consider node 0 as the source node, then in step - 1, the messages will first send to the node 4 which is the farthest node from node 0.
- In the next step the message is sent from node 4 to node 6. In this way the distance between sending and receiving nodes is halved and this is repeated till all nodes have been covered.
- The selection of recipient node is crucial to avoid congestion.
- For example in a case where node 0 sends message to node 1, and thereafter node 0 and node 1 send their messages to node 2 and 3 respectively, the congestion will occur because of the use of same link for data transfer by both the nodes.

☞ All-to-One Reduction on a Ring

Q. 3.1.3 Explain operation of All-to-one Reduction on a Ring. (Refer section 3.1.1) **(5 Marks)**

- The reduction operation on a linear array is easily performed by reversing the direction and communication sequence as compare to the previous diagram.
- The reduction steps are shown in the Fig. 3.1.4.
- During the reduction operation each odd numbered node sends its buffer to the even number node which is located just before itself.
- For example node 7 and node 5 sends their buffers to nodes 6 and 4 respectively. Similarly things performed in nodes 2 and node 0.
- Now the contents of the two buffers are combined in the next node.
- In the second step the contents of the nodes 0 and 2 are accumulated on node 0 and those on nodes 6 and 4 are accumulated on node 4.
- Finally, node 4 sends its buffer to node 0, which computes the final result of the reduction.

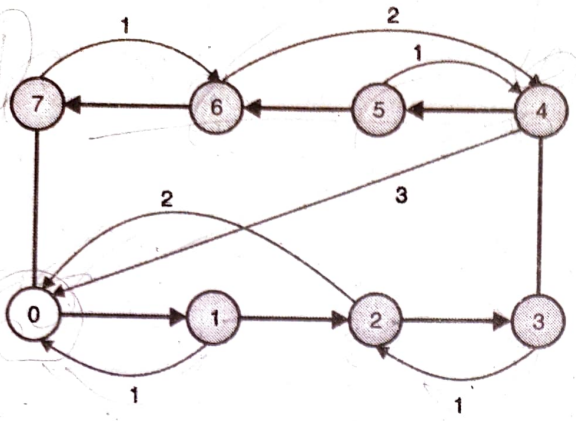
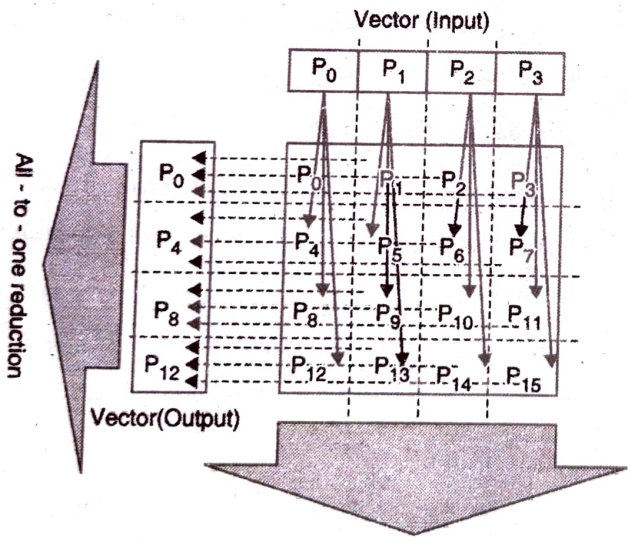


Fig. 3.1.4 : Reduction on an eight-node ring with node 0 as the destination of the reduction

☞ Broadcast and Reduction: Application Example

Consider the problem of multiplying a matrix (n × n) with a vector (n × 1). The n × n matrix is assigned to an n × n (virtual) processor grid. The vector is assumed to be on the first row of processors.

The first step of the product requires a one-to-all broadcast of the vector element along the corresponding column of processors. This can be done concurrently for all n columns.



One - to - all broadcast

Fig. 3.1.5

- The processors compute local product of the vector element and the local matrix entry.
- In the final step, the results of these products are accumulated to the first column using n concurrent all-to-one reduction operations along the columns (using the sum operation).

3.1.2 Mesh

- We can view each row and column of a square mesh of p nodes as a linear array of $\sqrt{P}$ nodes. Broadcast operation on mesh is like broadcast on linear array. Every node on the linear array has the data and broadcast on the columns with the linear array algorithm, in parallel.
- Broadcast and reduction operations can be performed in two steps - the first step does the operation along a row and the second step along each column concurrently. This process generalizes to higher dimensions as well.
- The Fig. 3.1.6 shows the one-to-all broadcast on a 16 - node mesh.

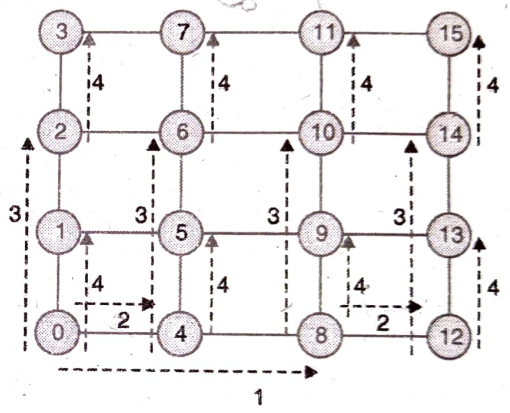


Fig. 3.1.6 : One-to-all broadcast on a 16-node mesh

3.1.3 Hypercube

- We have seen previously that on a two dimensional mesh the operation one - to - all broadcast is performed in two phases.
- The communication steps were performed along a different dimension in each phase. In the similar fashion, the process is carried out in three phases on a three-dimensional mesh.
- A hypercube with 2^d nodes can be regarded as a d-dimensional mesh with two nodes in each dimension. The mesh algorithm can be generalized to a hypercube and the operation is carried out in d (= log p) steps. The Fig. 3.1.7 shows a one-to-all broadcast on a three dimensional eight node hypercube.
- Here node 0 is considered as the source node. The highest dimension that is the most significant bit of the binary representation of a node label is the starting point of the communication. In subsequent steps the communication proceeds along with successively lower dimensions.

- The Fig. 3.1.7 shows one-to-all broadcast operation on a three dimensional hypercube. In this Fig. 3.1.7 the binary representation of nodes are represented in parenthesis.
- As compare to broadcasting steps using a linear array, the broadcasting used on a hypercube steps are almost similar but while using the hypercube, the order in which the dimensions are chosen for communication does not affect the outcome of the procedure.

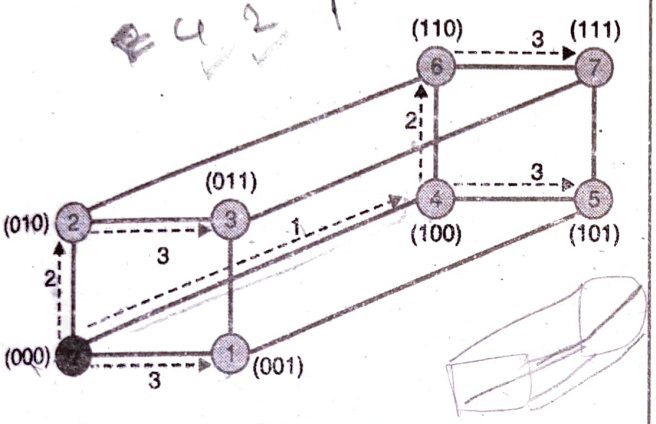


Fig. 3.1.7 : One-to-all broadcast on a three-dimensional hypercube.

- One-to-all broadcast on a three-dimensional hypercube. The binary representations of node labels are shown in parentheses. Reduction is done in reversed order.

3.1.4 Balanced Binary Tree

Consider a binary tree in which processors are (logically) at the leaves and internal nodes are routing nodes. Assume that source processor is the left most node.

- The process of broadcast is done in progressive manner. Reduction is done in reversed order.

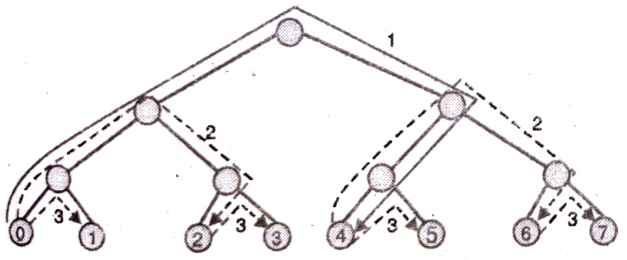


Fig. 3.1.8 : One-to-all broadcast on an eight-node tree

- The balanced binary tree is mapped naturally from the hypercube algorithm for one-to-all broadcast where intermediate nodes are the switching nodes and each leaf nodes are the processing nodes.
- This is depicted in the Fig. 3.1.8 for eight nodes. The Fig. 3.1.8 shows that congestion will not be present on any of the communication links at any time.
- There is different number of switching nodes along different paths when this tree is compared with the hypercube on the basis of communication.

3.1.5 Broadcast and Reduction Algorithms

Q. 3.1.4 Write a pseudo-code for One-to-All Broadcast Algorithm on Hypercube with different cases. (Refer section 3.1.5) **(8 Marks)**

- All of the algorithms described above are adaptations of the same algorithmic template. We illustrate the algorithm for a hypercube, but the algorithm, as has been seen, can be adapted to other architectures.
- The hypercube has 2^d nodes and my_id is the label for a node. An algorithm to broadcast from 0 is simply implemented by utilizing how the address bits map to the recursive construction of the hypercube.
- To support arbitrary source processors we us a mapping from physical processors to virtual processors. We always send from processor 0 in the virtual processor space.
- The XOR operation with the root gives us a idempotent mapping operation (apply once to get from virtual → physical, second time to get from physical → virtual).

```
Algorithm One_To_All_Broadcast_General(d, my_id , source, X)
{
    my_Vid = my_id XOR source; //virtual id assigned
    mask = 2d – 1;
    for(i = d-1; i>= 0 )
    {
        mask = mask XOR 2i; /*set bit i of mask to 0*/
        if (my_Vid AND mask ) == 0)
            if (my_Vid AND 2i ) == 0) {
```

Fig. 3.1.9 : Conted....

```
        virtual_dest =my_Vid XOR 2i ;
        send X to (virtual_dest XOR source);
        /*convert virtual_dest to the label of the
        physical destination */
      }//if
      else
      {
        virtual_source = my_Vid XOR 2i ;
      receive X from (virtual_source XOR source);
        /*convert virtual_dest to the label of the
        physical destination */
      }//else
    }
}
```

Fig. 3.1.9 : One-to-all broadcast of a message X from source on a hypercube

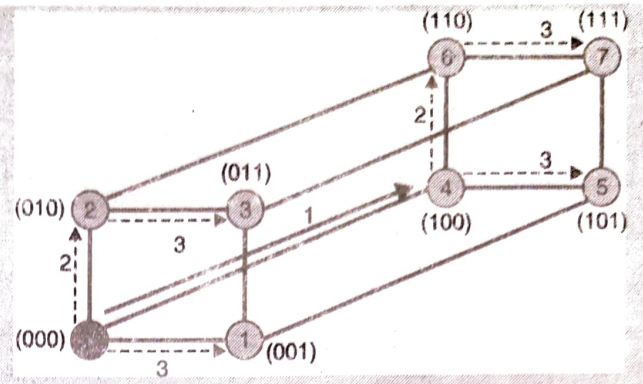
The following algorithm is based on Single-node accumulation on a d-dimensional hypercube. Each node contributes a message X containing m words, and node 0 is the destination.

```
Algorithm All_To_One_Reduce(d, my_id, m,X,sum)
{
  for (j = 0; j<= m-1; j++)
    sum[j] = X[j];
  mask = 0;
  for (i=0; i<= d-1; i++)
  {
    //select nodes whose lower I bits are 0
    if((my_id AND mask) == 0)
      if((my_id AND 2^i) != 0)
      {
        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; j<= m-1)
          sum[j] = sum[j] + X[j];
      }
    mask = mask XOR 2^i;  /*set bit i of mask to 1*/
  }
}
```

One-to-All Broadcast Algorithm on Hypercube (Case : 100: 0 to 4)

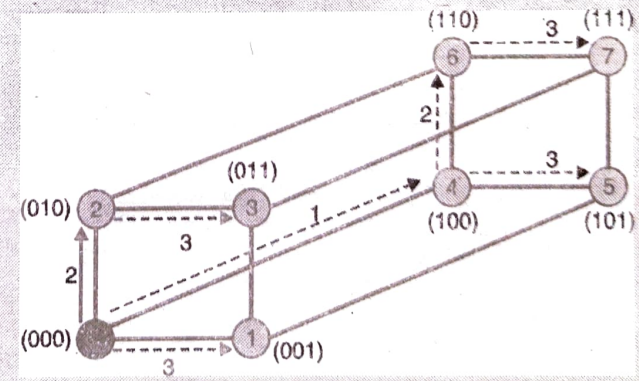
```
Algorithm ONE_TO_ALL_BC(d, my_id, X)
{
  mask:= 2^d - 1; /* Set all d bits of mask to 1 */        (111)
  for (i = d - 1 >= 0)  /* Outer loop */                   i = d – 1 → 2
  {
    mask =mask XOR 2^i; /* Set bit i of mask to 0 */       111 XOR 100 = (011)
    if ((my_id AND mask) == 0) /* If lower i bits of my_id are 0* /
                                                           (000 & 011) → 000
    {
      if ((my_id AND 2^i)==)                               (000 & 100) → 000
(100 & 011) → 000
(100 & 100) → 100
      msg_destination = my_id XOR 2^i;                     (000 XOR 100) → 100
      send (X to msg_destination);
    }
    else                                                   (100 XOR 100) → 000
    {
```

```
            msg_source : = my_id XOR 2^i ;
            receive (X from msg_source)
        }
    }
}
```



☞ **One-to-All Broadcast Algorithm on Hypercube Case : (010: 0 to 2)**

```
Algorithm ONE_TO_ALL_BC(d, my_id, X)
{
    mask:= 2^d - 1; /* Set all d bits of mask to 1 */        (111)
    for (i = d - 1 >= 0)  /* Outer loop */                   i = 1
    {
        mask =mask XOR 2^i; /* Set bit i of mask to 0 */     111 XOR 010 = (101)
        if ((my_id AND mask) == 0) /* if lower i bits of my_id are 0* /   (000 & 101) → 000
        {
            if ((my_id AND 2^i)==)                          (000 & 010) → 000
(010 & 101) → 000
(010 & 010) → 010
            msg_destination = my_id XOR 2^i;                (000 XOR 010) → 010
            send (X to msg_destination);                    0 → 2
        }
        else                                                (010 XOR 010) → 000
        {
            msg_source : = my_id XOR 2^i ;
            receive (X from msg_source)
        }
    }
}
```



☞ **One-to-All Broadcast Algorithm on Hypercube Case : (001 : 0 to 1)**

```
Algorithm ONE_TO_ALL_BC(d, my_id, X)
{
    mask:= 2^d - 1; /* Set all d bits of mask to 1 */        (111)
```

```
for (i = d - 1 >= 0)  /* Outer loop */
{
    mask = mask XOR 2^i; /* Set bit i of mask to 0 */
        if ((my_id AND mask) == 0) /* If lower i bits of my_id are 0* /
        {
            if ((my_id AND 2^i) == 0)
                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)
        }
    }
}
```

i = 0

111 XOR 001 = (110)

(000 & 110) → 000

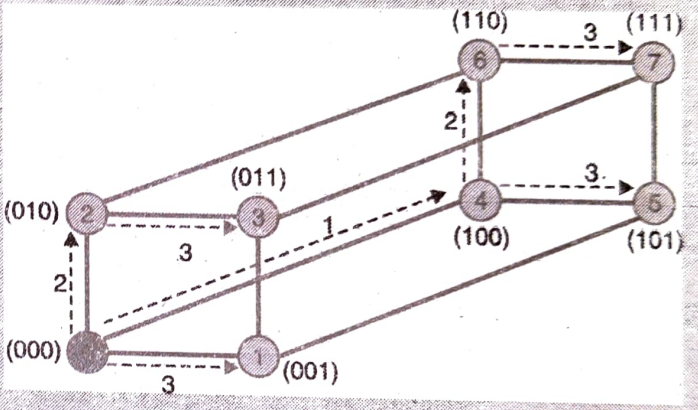
(000 & 001) → 000

(001 & 110) → 000
(001 & 001) → 001

(000 XOR 001) → 001

0 → 1

(001 XOR 001) → 000



Broadcast and Reduction Algorithms

```
Algorithm One_To_All_Broadcast_General(d, my_id, source, X)
{
    my_vid = my_id XOR source ; //virtual id assigned
    mask = 2^d – 1;
    for (i = d-1;i>=0)
    {
        mask = mask XOR 2^i; /*set bit i of mask to 0* /
        if (my_Vid AND mask) == 0)
            if (my_Vid AND 2^i) == 0) {
                virtual_dest = my_vid XOR 2^i ;
                send (X to (virtual_dest XOR source));
                /*convert virtual_dest to the label of the physical destination */
            }//if
            else
```

(001 XOR 001) = 000

(111)

i =2

(111 XOR 100) = (011)

(000 & 011) = (000)

(000 & 100) = (000)

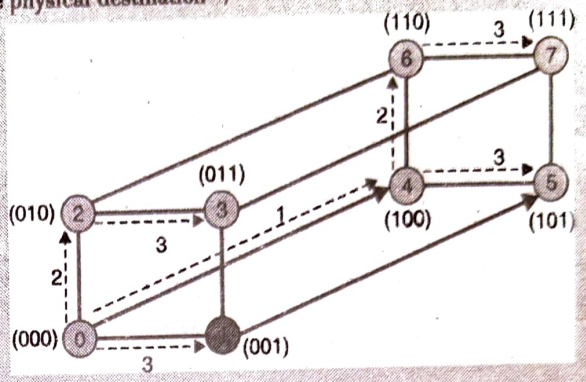
(000 & 100) = (100)

(100 XOR 001) = 101 (5)

```
{
    virtual_source = my_vid XOR 2^i ;
    receive (X from (virtual_source XOR))
    /*convert virtual_dest to the label of the physical destination */
}//else
}
}
```

(100 XOR 100) = 000

(000 XOR 001) = 001

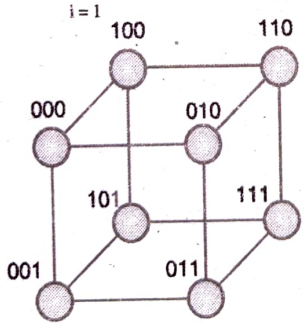


3.1.6 Cost Analysis for One-to-Many Broadcast and Many-to-One Reduction on Hypercube

- Let us consider p processes participate in the operation. Also assume m words contained in the data to be broadcasted and reduced.
- The broadcast or reduction procedure involves **log p** point-to-point simple message transfers, each at a time cost of $t_s + t_w m$ (setup time + transmission time for all words).

The total time is therefore given by :

$$T_{comm} = \sum_{i=1}^{\log p} (t_s + t_w m) = (t_s + t_w m) \log p$$



Syllabus Topic : All-to-All Broadcast and Reduction

3.2 All-to-All Broadcast and Reduction

Q. 3.2.1 Explain term of all-to-all broadcast on linear array, mesh and hypercube topologies. **(8 Marks)**

(Refer section 3.2)

- This is a generalized approach of broadcast in which each processor is the source as well as destination. The same m word message is sent by a process to all other processes but different processes can broadcast different messages.
- The matrix operations such as matrix multiplication and matrix - vector multiplication are implemented using the all - to - all broadcast approach.
- The all - to - all reduction is the dual of all - to - all broadcast where every node is the destination of an all-to-one reduction.

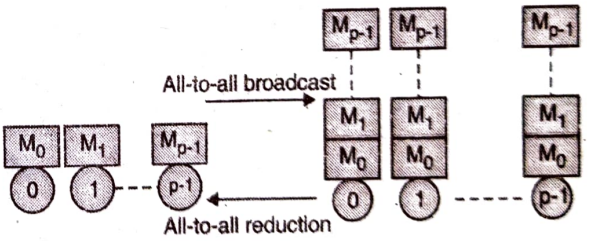


Fig. 3.2.1 : All-to-all broadcast and all-to-all reduction

- The all – to – all broadcast is handled if performed naively; it may take up to p times as long as a one-to-all broadcast (for p processors). Possible to concatenate all messages that are going through the same path (reduce time because fewer t_s).
- One way to perform an all-to-all broadcast is to perform p one-to-all broadcasts, one starting at each node. If performed naively, on some architecture this approach may take up to p times as long as a one-to-all broadcast.

- It is possible to use the communication links in the interconnection network more efficiently by performing all p one-to-all broadcasts simultaneously so that all messages traversing the same path at the same time are concatenated into a single message whose size is the sum of the sizes of individual messages.
- The descriptions about all – to – all broadcast on linear array, mesh and hypercube topologies are given in the following sections.

3.2.1 All - to - all Broadcast and Reduction on a Ring

- In its simplest approach it performs **p** one-to-all broadcasts. This is actually not the most efficient way.
- In this method each node first sends message to one of its neighbors the data it needs to broadcast. In subsequent steps, it forwards the data received from one of its neighbors to its other neighbor. The algorithm terminates in **p-1** steps. The Fig. 3.2.2 shows the steps in All-to-all broadcast on an eight-node ring.

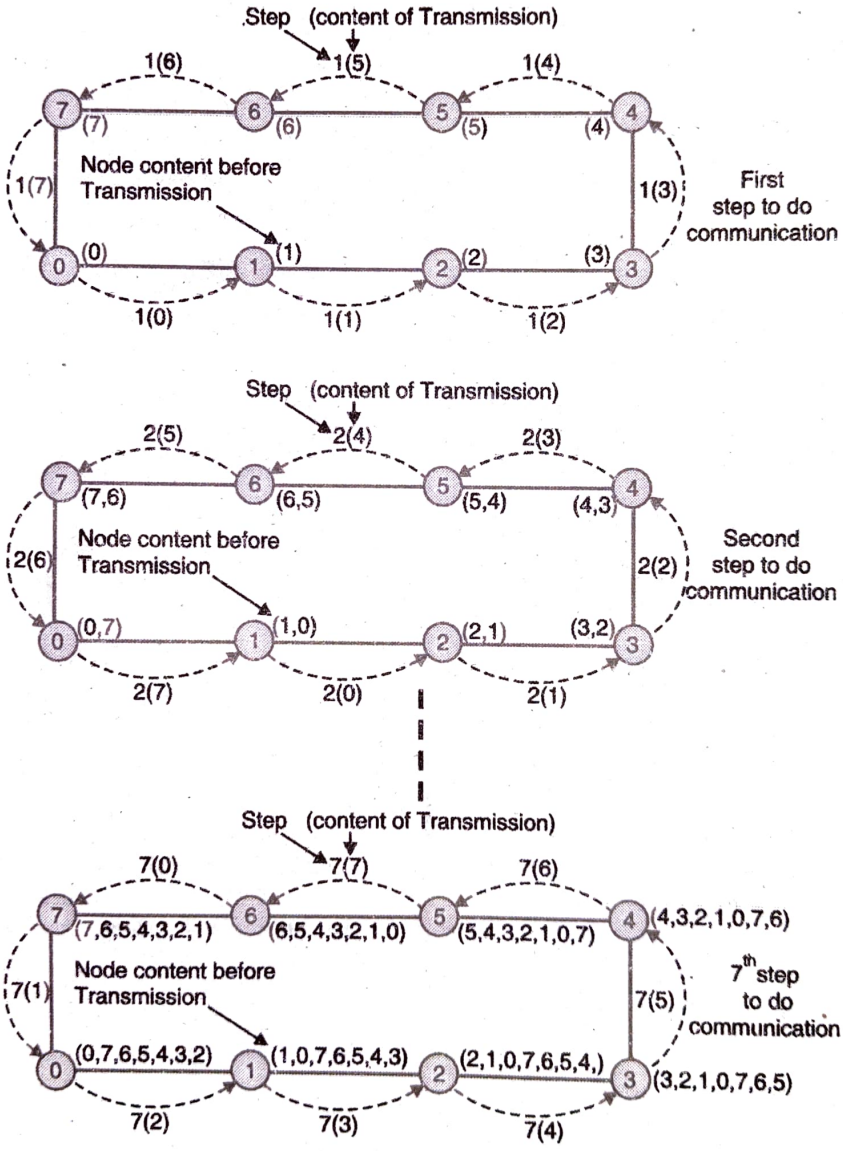


Fig. 3.2.2 : All-to-all broadcast on an eight-node ring

```
Algorithm All_to_All_Broadcast_Ring(my_id, my_msg, p, result)
{
   left = (my_id – 1) mod p ;
   right = (my_id + 1) mod p ;
   result = my_msg ;
   msg = result ;
   for (i = 1 ; k=p-1)
   {
      send (msg to right) ;
      receive (msg from left) ;
      result = result ∪ msg ;
   }
}
```

Ring: mod p

Receive & send point - to - point

Initialize the loop

Forward msg

Accumulate result

Fig. 3.2.3 : All-to-All Broadcast Algorithm

```
Algorithm All_to_All_Reduction_Ring(my_id, my_msg, p, result)
{
   left = (my_id – 1) mod p ;
   right = (my_id + 1) mod p ;

   recv = 0 ;
   for (i = 1; i<=p-1)
   {
      j =(my_id + i) mod p ;
      temp = msg[j] + recv ;
      send (temp to left);
      receive (recv from right);
   }
   result = msg [my_id] + recv;
}
```

Accumulate and forward

Last message for my_id

Fig. 3.2.4 : All-to-All Reduce Algorithm

☞ Analysis of ring all-to-all broadcast algorithm

– The algorithm does p-1 steps and in each step it sends and receives a message of size m. Therefore the communication time is :

$$T_{\text{all-to-all-ring}} = \sum_{i=1}^{\log p} (t_s + t_w m) = (t_s + t_w m)(p - 1)$$

- Note that the bisection width of the ring is 2, while the communication pattern requires the transmission of p/2 pieces of information from one half of the network to the other. Therefore the all-to-all broadcast cannot be faster than O(p) for a ring. Therefore this algorithm is asymptotically optimal.

3.2.2 All - to - all Broadcast on a mesh

- The all - to - all broadcast algorithm for the 2-D mesh is similar to the one - to - all broadcast based on the linear array algorithm. Here rows and columns of the mesh are treated like linear arrays. This is also based on the approach in which communication takes place in two phases.
- The procedure for the linear array is used in the first phase in which each row of the mesh performs an all – to –all broadcast. Here $\sqrt{p}$ messages corresponding to the $\sqrt{p}$ nodes of their respective rows are collected by all nodes. This information is consolidated by each node into a single message of size $m\sqrt{p}$.
- The second communication phase is a column-wise all-to-all broadcast of the consolidated messages. When second phase ends all p pieces of m – word data are obtained by each node that originally resides on different nodes.

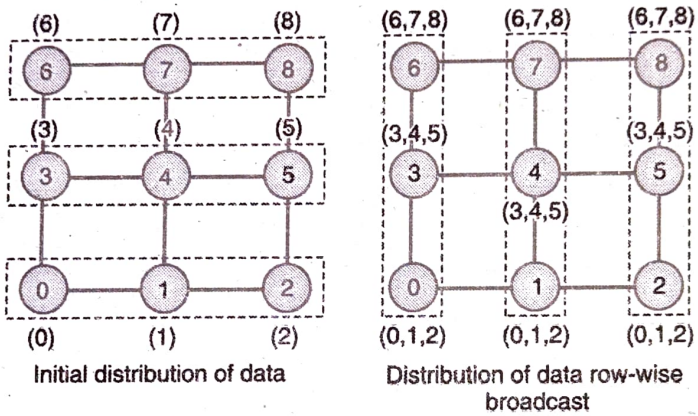


Fig. 3.2.5 : All-to-all broadcast on a 3 × 3 mesh. The groups of nodes communicating with each other in each phase are enclosed by dotted boundaries. By the end of the second phase, all nodes get (0,1,2,3,4,5,6,7) (that is, a message from each node).

```
Algorithm All_to_All_Broadcast_mesh(my_id, my_msg, p, result)
{
        /*communication along rows*/                          add the offset
Translate   left = my_id – (my_id mod sqrt (p)) + (my_id – 1) mod sqrt (p);
to left end
        right = my_id – (my_id mod sqrt(p)) + ((my_id + 1) mod sqrt(p));
        result = my_msg;
        msg = result;
        for (i = 1; k=sqrt(p) – 1 ; i++)
        {
            send (msg to right);
            receive (msg form left);
            result = result ∪ msg;
        }
```

Fig. 3.2.6 : Conted...

```
/*communication along columns
up = (my_id – sqrt(p)) mod p ;
down = (my_id + sqrt(p)) mod p ;
msg = result;
for (i = 1; i <=(sqrt(p) – 1); i++)
{
    send(msg to down);
    receive (msg from up);
    result = result ∪ msg;
}
}
```

(0 – 3) % 9 → 6

(0 + 3) % 9 → 3

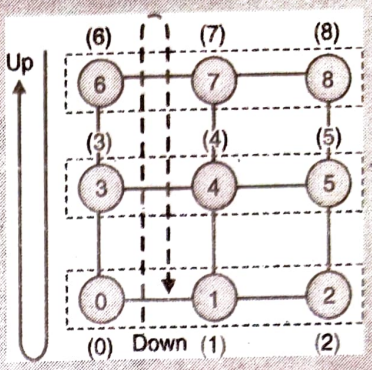


Fig. 3.2.6 : All-to-all broadcast on a square mesh of p nodes

☞ Mesh based All-to-All broadcast Analysis

- Algorithm proceeds in two steps:
  1) Ring broadcast over rows with message size = m, then
  2) Ring broadcast over columns with message size = $\sqrt{p}$ m.

Time for communication

$$T_{comm} = \overbrace{(t_s + t_w m)(\sqrt{p} - 1)}^{\text{Step I}} + \overbrace{(t_s + t_w \sqrt{p} m)(\sqrt{p} - 1)}^{\text{Step II}}$$

$$T_{comm} = 2t_s(\sqrt{p} - 1) + t_w m(p - 1)$$

- Due to single-port assumption, all-to-all broadcast cannot execute faster than O(p) time since each processor must receive p-1 distinct messages. Therefore this algorithm is asymptotically optimal.

3.2.3 All - to - all Broadcast on a Hypercube

- The all - to - all broadcast on a hypercube algorithm is an extension of the mesh algorithm to log p dimensions. In each step communication is done along a different dimension of the hypercube. A pair of nodes exchange their data in each step and the overall data becomes double in size that to be transmitted onto the next step. In this stage the received message is concatenated with the current data of the receiver.
- The Fig. 3.2.7 shows the steps for an eight node hypercube with bidirectional communication channels.

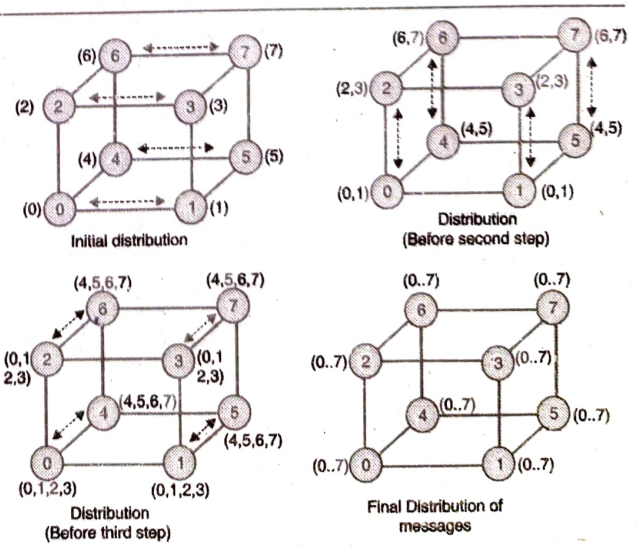


Fig. 3.2.7 : All-to-all broadcast on an eight-node hypercube

- **Algorithm :** At every step we have a broadcast on sub-cubes. The size of the sub-cubes doubles at every step and all the nodes exchange their messages.

```
Algorithm All_to_All_Broadcast_HC(my_id, my_msg, d,
result)
{
    result = my_ msg;
    for(i = 0; i<=d-1) /*loop on the dimensions*/
    {
        partner = my_id XOR 2^i ;
        send(result to partner);
        receive(msg from partner)  /*exchange messages*/
        result = result ∪ msg;  /*forward (double size)*/
    }
}
```

Fig. 3.2.8 : All to all broadcast on a d – dimensional hypercube

The communication with diagrammatic representation for all - to - all broadcast on a d – dimensional hypercube is given in Fig. 3.2.9.

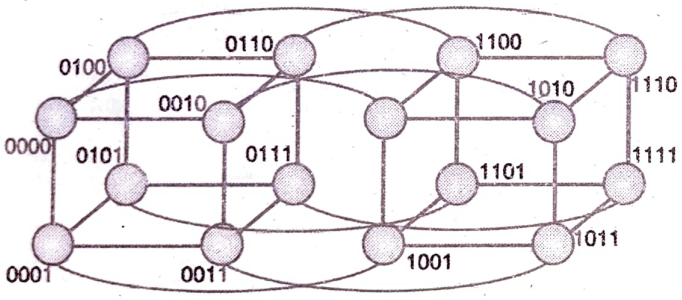


Fig. 3.2.9 : All-to-all broadcast on a d-dimensional hypercube

Syllabus Topic : All-Reduce and Prefix-Sum Operations

3.3 All-Reduce and Prefix-Sum Operations

- This operation is the extension of all - to - all broadcast communication pattern. This is variation of the reduce operation in which each node starts with a buffer of size m and the final result is the same combination of all buffers on every node.
- This operation is syntactically similar to the combined technique of all - to - one reduce and one - to - all broadcast. This operation differs in working as compare to all - to - all reduce.
- The operation is depicted in the Fig. 3.3.1.

Fig. 3.3.1

- This operation acts as a barrier synchronization on a message – passing computer when each node has a single word message.
- The reduction operation cannot be finished by any node before contributing a value by each node when this operation is implemented and executed as a parallel program.
- The all – reduce operation can be performed in a simple way if two operations all - to - one reduction and one - to - all broadcast is used but first all - to - one reduction is performed thereafter one - to - all broadcast is applied.
- The drawback of this approach is that it is not performed in most efficient ways because it is performed effectively by using the communication pattern of all - to - all broadcast. The only difference is that message size does not increase here. Time for this operation is $(t_s + t_w m) \log p$.
- This operation is different from all-to-all reduction, in which p simultaneous all-to-one reductions take place, each with a different destination for the result.

The Prefix-Sum Operation

- Finding prefix - sum operation is also called as **scan operation**. This is an important problem which can be solved by the use of communication pattern supported in all - to - all broadcast and all - reduce operations.
- Given p numbers $n_0, n_1, \ldots, n_{p-1}$ (one on each node), the problem is to compute the sums $s_k = \sum_{i=0}^{k} n_i$
- For all k between 0 and p-1. For example, consider one sequence in which numbers are originally arranged as <3,1,4,0,2>, now the sequence of prefix sum will be <3,4,8,8,10>.
- Initially, n_k resides on the node labeled k, and at the end of the procedure, the same node holds S_k.

3.3.1 Prefix-Sum Algorithm

Q. 3.3.1 Implement Prefix-Sum Algorithm with an example. (Refer section 3.3.1) **(5 Marks)**

The following algorithm gives a procedure to solve the prefix sums problem on a d-dimensional hypercube.

```
Algorithm Prefix_sums_HC(my_id, my_num, d, result)
{
    result = my_num;
    msg = result;
    for (i = 0; i <= d-1)
    {
        partner = my_id XOR 2^i;
        send(msg to partner);
        receive(number from partner);
        msg = msg + number;
                                        All-reduce
        if(partner < my_id)
            result = result + number ;
                                        Prefix-sum
    }
}
```

The Fig. 3.3.2 gives an example of 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.

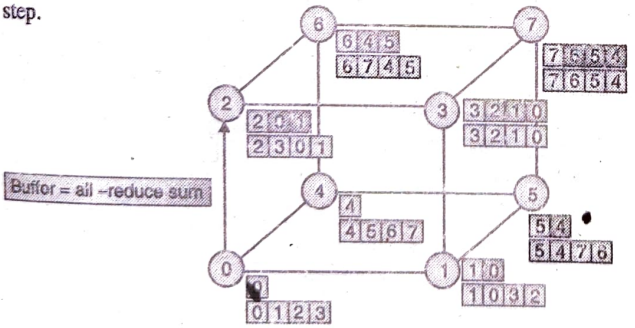


Fig. 3.3.2 : Prefix sums on an eight-node hypercube

Syllabus Topic : Scatter and Gather

3.4 Scatter and Gather

Q. 3.4.1 Explain scatter and gather operation. (Refer section 3.4) **(5 Marks)**

- In scatter operation a node sends a unique message to every other node – unique per node. The scatter operation works on the fact that, a unique message of size m is send to every node by a single node. This is also called as **one – to – all personalized communication operation.**
- The one – to – all broadcast and one – to – all personalized communication operations are different with each other. In one – to – all personalized communication operations the source node starts with p unique messages, where one message is destined for each node.
- In this method of communication the problem of data duplication does not involved.
- The gather operation is a dual operation but the target node does not combine the messages into one. The gather operation is the dual of the scatter operation. In fact this is a concatenation operation in which a single node collects a unique message from each node.
- The gather operation differs from all – to – one reduce operation because it does not involve any combination or reduction of data.

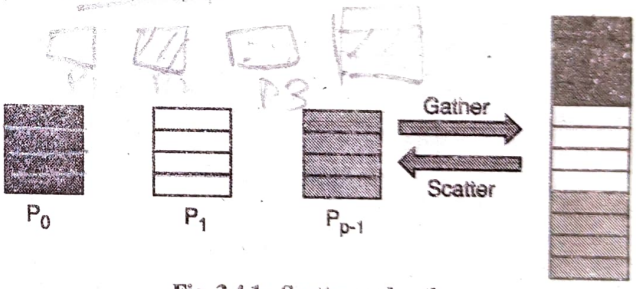


Fig. 3.4.1 : Scatter and gather

- The scatter operation is very much similar to the broadcast operation though it differs semantically from one - to - all broadcast operation. The communication steps for the scatter operation on an eight – node hypercube is shown on the Fig. 3.4.2.
- The pattern of communication is identical with one-to-all broadcast but the size and the content of the messages are different. Scatter is the reverse operation. This algorithm can be applied for other topologies.
- In Fig. 3.4.2 all messages are contained initially by the source node that is node 0. The messages contains destination node labels through which they are identified. The source transfers half of the messages to one of its neighbors in the first step of communication.

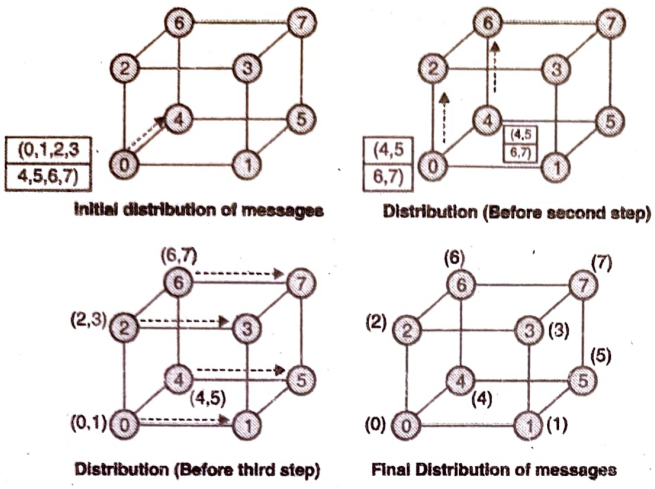


Fig. 3.4.2

- In subsequent steps, each node that has some data transfers half of it to a neighbor that has yet to receive any data. There is a total of log p communication steps corresponding to the log p dimensions of the hypercube.

Cost Analysis

Number of steps required in the communication = log p.

Size transferred : pm/2, pm/4,…,m

Geometric sum →

$$p + \frac{p}{2} + \frac{p}{4} + \ldots + \frac{p}{2^n} = p\frac{1 - \frac{1}{2^{n+1}}}{1 - \frac{1}{2}}$$

$$\frac{p}{2} + \frac{p}{4} + \ldots + \frac{p}{2^n} = 2p\left(1 - \frac{1}{2^{n+1}}\right) - p$$

$$= 2p\left(1 - \frac{1}{2p}\right) - p$$

$$= p - 1$$

$(2^{n+1} = 2^{1+\log p} = 2p)$

$$\text{Cost } T = t_s \log p + t_w m (p - 1)$$

The term t_w m(p-1) is a lower bound for any topology because the message of size m has to be transmitted to p-1 nodes, which gives the lower bound of m(p-1) words of data.

Syllabus Topic : All-to-All Personalized Communication

3.5 All-to-All Personalized Communication

Q. 3.5.1 What is total exchange method ? (Refer section 3.5) (2 Marks)

Q. 3.5.2 Write short note on :

(i) Total Exchange on a Ring

(ii) Total Exchange on a Mesh

(iii) Total Exchange on a Hypercube

(Refer sections 3.5.1, 3.5.2 and 3.5.3) (12 Marks)

In this communication method each node sends a message to every other node. The message to be transferred are of size m and each message is distinct with other. This method differs from all - to - all broadcast approach in which the same message is transferred to all other nodes. All - to - all personalized communication is also called as **total exchange method** (a transpose).

This method is used in various parallel algorithms like, Fourier transform, matrix transpose, some parallel database join operations etc. The result of all – to – all communication method is similar to transpose of the input if it is seen as a matrix.

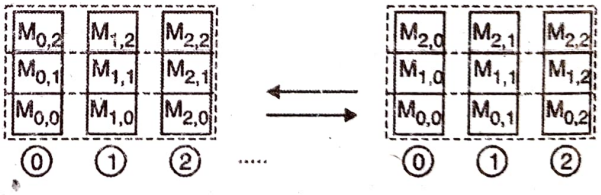


Fig. 3.5.1 : All – to – all personalized communication

Example : Matrix transposition

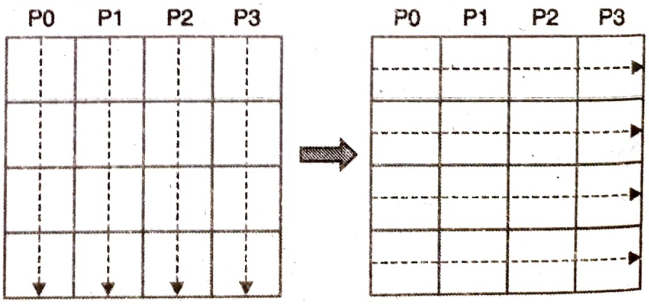


Fig. 3.5.2 : All - to - all personalized communication in transposing a 4 × 4 matrix using 4 processes

- Suppose we have N× N matrix and want to perform transpose in parallel. One full row of the matrix is assigned to each processor when the matrix is mapped onto N processors.
- This mappings results that the processor P_i initially contains the elements of the matrix with indices $[i, 0]$, $[i, 1]$, ..., $[i, n - 1]$.
- The different sets of elements belong to different processor after transposition as P0 contains element [I,0], P1 contains element [I,1] and so on.
- This operation can be described in general as element [I,j] moves to Pj during the transposition which was initially residing on Pi.
- The Fig. 3.5.2 is an example of the all - to - all personalized communication and shows the data – communication pattern of the procedure for a 4 × 4 matrix mapped onto four processes using one dimensional row – wise partitioning. In this Fig. 3.5.2 a distinct set of elements of the matrix is given to every other procedure.

3.5.1 Total Exchange on a Ring

- When we consider to apply total exchange operation on a ring, each node sends all pieces of data as one consolidated message of size m(p – 1) to one of its neighbors.
- 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.
- The algorithm terminates in p – 1 steps. At each step a message is reduced by the size of m. The operation is depicted in the Fig. 3.5.3.
- In this approach every node sends (p – 1) pieces of data with size m during the operation.

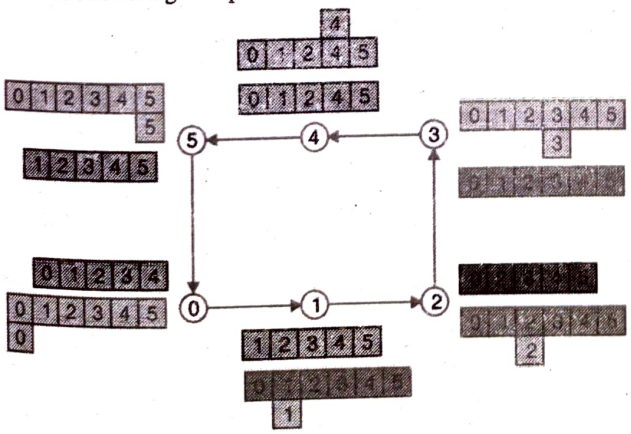


Fig. 3.5.3 : Contd..

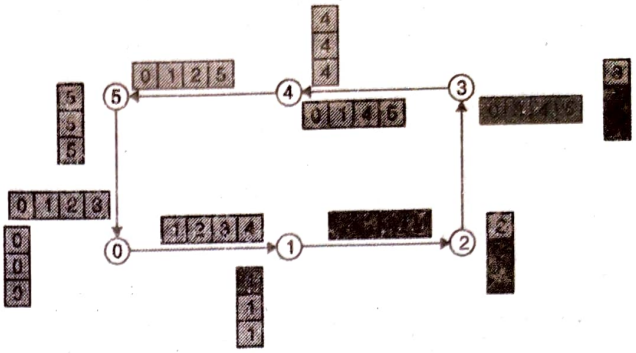


Fig. 3.5.3 : Total exchange on ring

☞ Cost analysis on ring

- All – to – all communication on a ring requires (p-1) communication steps in total. The total time required to perform this operation is represented by the following equation, since in the i^{th} step the size of the messages transferred is m(p-i).

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

$$= t_s (p - 1) + \sum_{i=1}^{p-1} i t_w m$$

$$= (t_s + t_w mp / 2) (p - 1)$$

- The t_w term in this equation can be reduced by a factor of 2 by communicating messages in both directions.
- In average we transmit mp/2 words, whereas the linear all-to-all transmits m words. If we make this substitution, we have the same cost as the previous linear array procedure. To really see optimality we have to check the lowest possible needed data transmission and compare it to T.
- Average distance a packet travels = p/2. There are p nodes that need to transmit m(p-1) words.
- Total traffic = m(p-1)*p/2*p. Number of link that support the load = p, to communication time ≥ t_wm(p-1)p/2.

3.5.2 Total Exchange on a Mesh

- In total exchange on a mesh, each node first groups its p messages according to the columns of their destination nodes. All-to-all personalized communication is performed independently in each row with clustered messages of size $m\sqrt{p}$.

- At this stage messages in each node are sorted again, this time according to the rows of their destination nodes. All-to-all personalized communication is performed independently in each column with clustered messages of size $m\sqrt{p}$.
- The Fig. 3.5.4 shows a 3 × 3 mesh. Initially every node has nine messages of size m – words. Here one message is meant for each node. Each node assembles its data into three groups of three 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.
- Thereafter all - to - all personalized communication is performed independently in each row with clustered messages of size $m\sqrt{p}$. In one cluster the information for all $\sqrt{p}$ nodes of a particular column is contained. The distribution of data among the nodes at the end of this phase of communication is shown in Fig. 3.5.4(b).

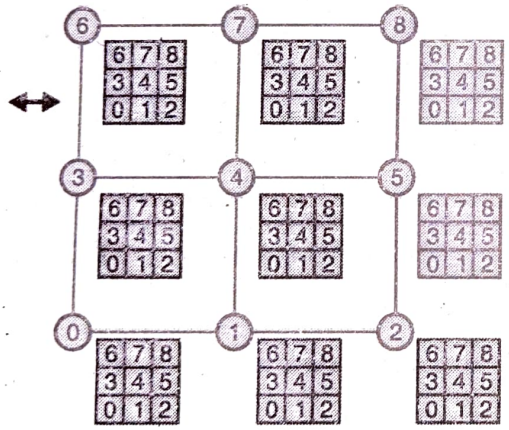


Fig. 3.5.4(a)

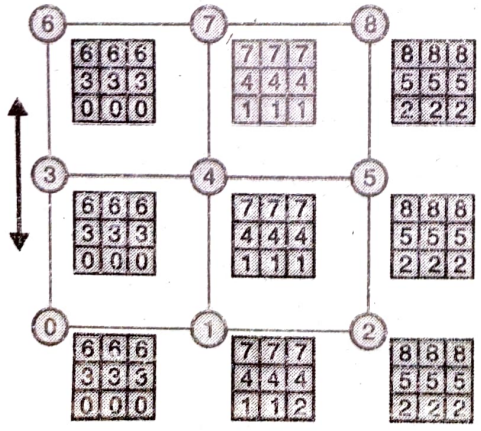


Fig. 3.5.4(b)

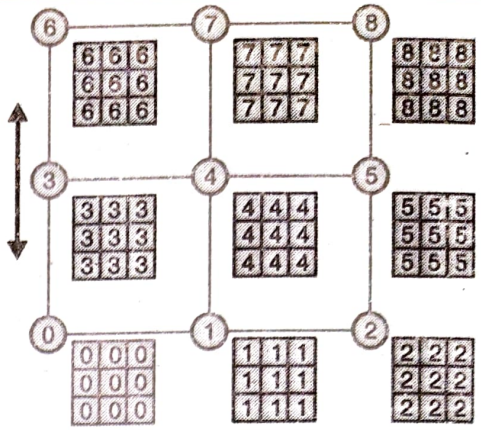


Fig. 3.5.4(c)

☞ Cost analysis of All-to-All Personalized Communication on a Mesh

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

$$T = (2t_s + t_w mp)(\sqrt{p} - 1)$$

- We have $p(\sqrt{p} - 1)m$ words transferred, looks worse than lower bound in $(p - 1)m$ but no congestion. Notice that the time for data rearrangement is not taken into account. It is almost optimal.

3.5.3 Total Exchange on a Hypercube

- All – to – all personalized communication on a p – node hypercube is performed by extending the two – dimensional mesh algorithm to log p dimensions. At any stage in all-to-all personalized communication, every node holds p packets of size m each.
- Every node sends p/2 of these packets which is consolidated as one message during the communication in a particular dimension. While performing total exchange on hypercube a node must rearrange its messages locally before each of the log p communication steps.
- The operation is shown in Fig. 3.5.5. The communication takes log p steps as shown in the Fig. 3.5.5. In each step, pairs of nodes exchange data in a different dimension.
- Every node contains p packets with size of each packet is m at any stage of the communication in personalized communication process. The destinations of these packets are the nodes of the other sub-cube connected by the links in current dimension.

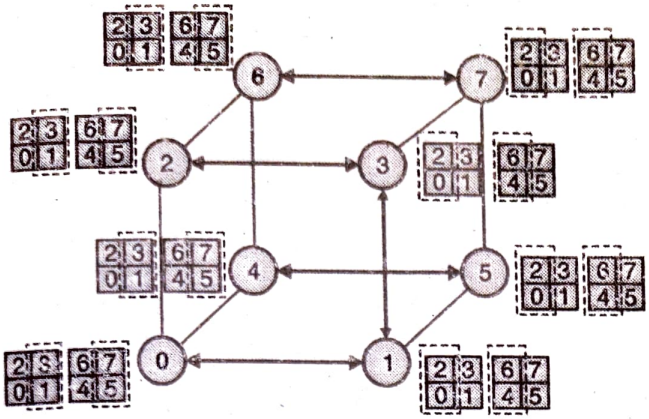


Fig. 3.5.5

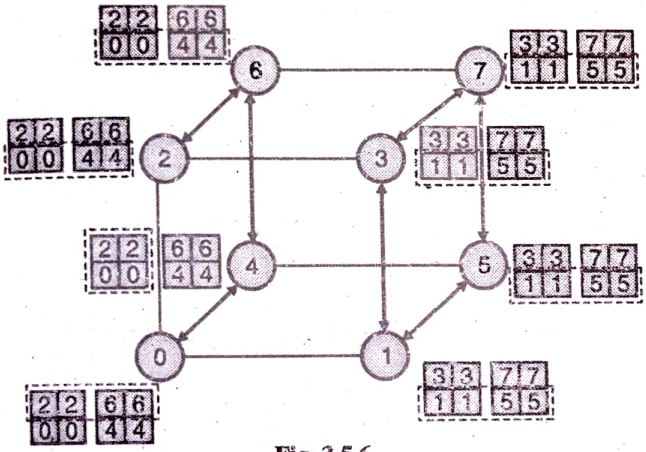


Fig. 3.5.6

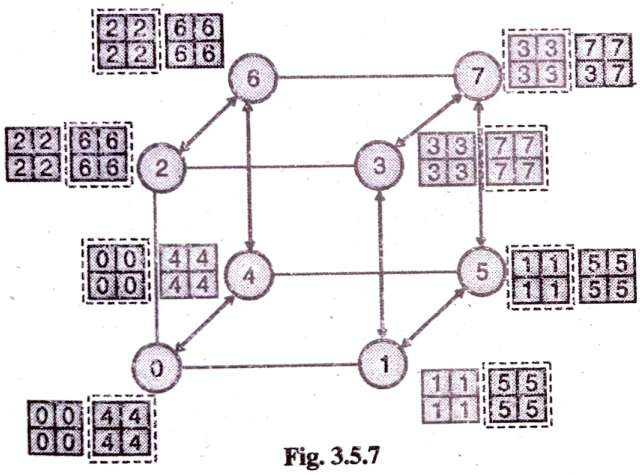


Fig. 3.5.7

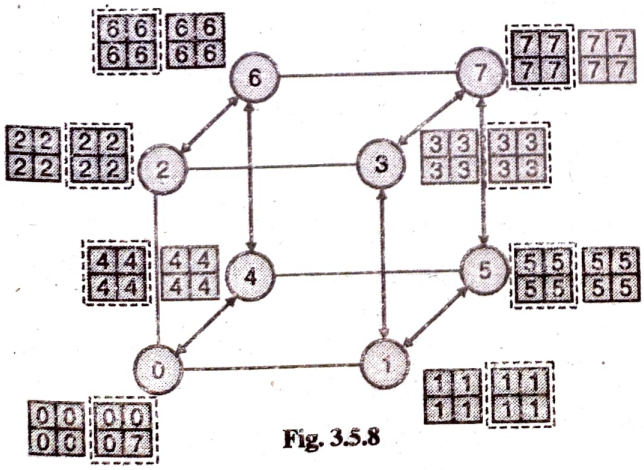


Fig. 3.5.8

Cost Analysis

- Total mp/2 words of data are exchanged along the bidirectional channels in log p steps in the hypercube algorithm for all – to – all personalized communication. The total communication time is given by the following equation:

  $T=(t_s + t_w mp/2) \log p$

- The hypercube algorithm is not optimal as compare to linear array and mesh algorithm for all – to – all personalized communication.
- Here each of the p nodes sends and receive m(p-1) words of data and the average distance between any two nodes on a hypercube is (log p)/2. This implies the total traffic on the network is $p \times m(p-1) \times (\log p)/2$.
- The total lower bound of the total exchange time is represented by the following equation because the hypercube contains total (p log p)/2 links.

$$T = \frac{t_w\, pm\,(p-1)\,(\log p)/2}{(p \log p)/2} = t_w\, m\,(p-1)$$

3.5.4 All-to-All Personalized Communication on a Hypercube: Optimal Algorithm

Q. 3.5.3 With the help of diagram explain All-to-All Personalized Communication on a Hypercube. (Refer section 3.5.4) **(8 Marks)**

- An all – to – all personalized communication is also called as total exchange algorithm because all pairs of nodes exchange some data in communication. This operation effectively performed on a hypercube if every pair of nodes communicates directly with each other.
- In this way each node exchange m words of data with a different node in every step and performs p-1 communication step. The hypercube links do not suffer with congestion problem because a communication partner of a node is chosen by itself in every step.
- The Fig. 3.5.9 shows pairwise exchange of data in a three dimensional hypercube. According to this Fig. 3.5.9, in the j^{th} communication step, node i exchanges data with node (i XOR j). In this schedule, all paths in every communication step are congestion-free, and none of the bidirectional links carry more than one message in the same direction.

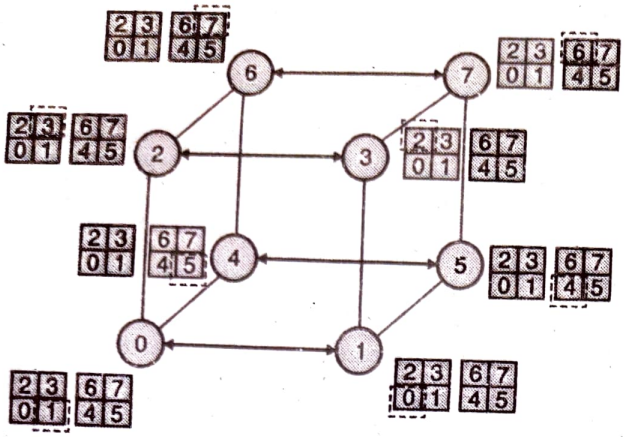


Fig. 3.5.9

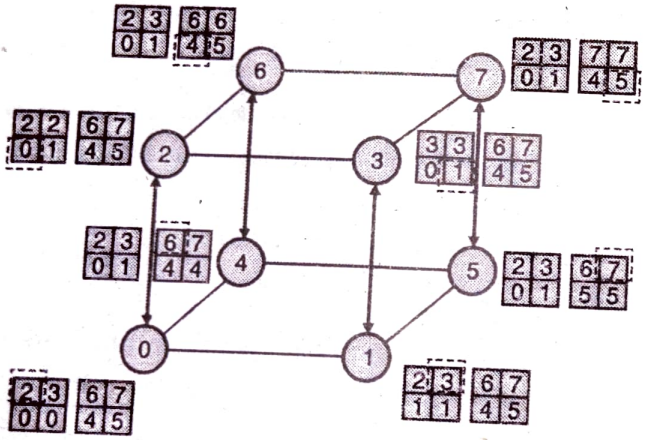


Fig. 3.5.10

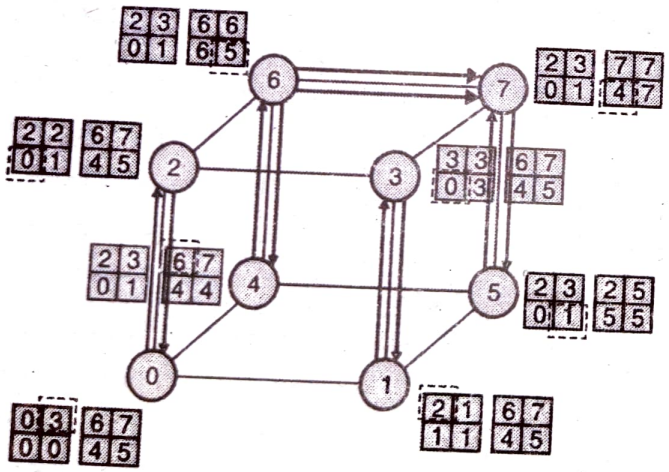


Fig. 3.5.11

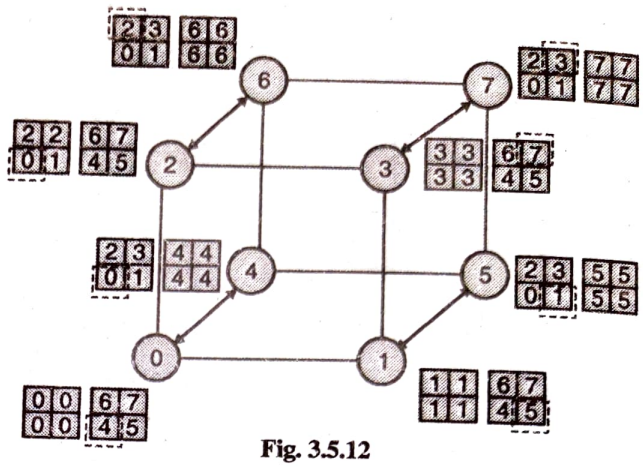


Fig. 3.5.12

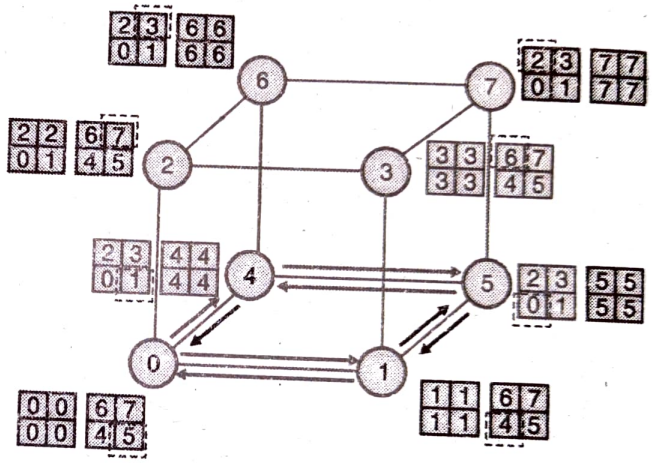


Fig. 3.5.13

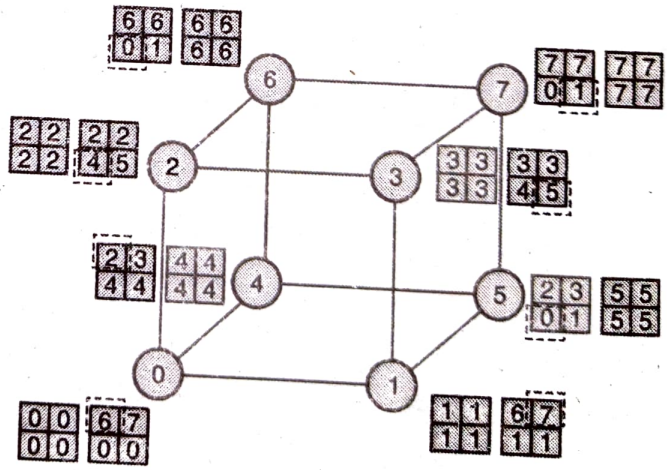


Fig. 3.5.14

☞ All-to-All Personalized Communication on a Hypercube : Optimal Algorithm

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.

```
Algorithm All_To_All_Personal (d, my_id)
{
    for (i = 1; i<=2^d – 1;i++)
    {
        partner=my_id XOR i ;
        send(M_{my_id, partner} to partner);
        receive(M_{partner,my_id} from partner);
    }
}
```

☞ **Cost Analysis of Optimal Algorithm**

- There are **p – 1** steps and each step involves non-congesting message transfer of m words. A message transfer between node i and node j requires $t_s + t_w m$ time for the communication.
- It takes this much communication time because there is no contention between any other message travelling in the same direction along the link between nodes i and j.

Therefore the total time is given by :

$$T = (t_s + t_w m)(p - 1).$$

This is asymptotically optimal in message size.

Syllabus Topic : Circular Shift

3.6 Circular Shift

Q. 3.6.1 What is circular shift ? (Refer section 3.6) **(2 Marks)**

Q. 3.6.2 Write short note on :
(i) Circular Shift on a Mesh
(ii) Circular Shift on a Hypercube
(Refer sections 3.6.1 and 3.6.2) **(6 Marks)**

Circular shift is a particular permutation operation. A permutation is a simultaneous one – to – one redistribution operation. In permutation each node sends a packet of m words to a unique node. A circular – q – shift is defined as an operation where node i sends a data packet to node (i + q) mod p in a p node ensemble (0 < q < p). Circular shift is useful in some matrix operations and pattern matching.

3.6.1 Circular Shift on a Mesh

- The implementation on a ring is rather intuitive. It can be performed in min{q, p – q} neighbour communications. Mesh algorithms follow from this as well. We shift in one direction (all processors) followed by the next direction.
- A circular – q – shift can be performed on a p – node square wraparound mesh in two steps in the case where the nodes of the mesh have row – major labels.
- This operation is described in the Fig. 3.6.1 for a circular 5 – shift on a 4 × 4 mesh. In the first step the entire set of data is shifted simultaneously by (q mod $\sqrt{p}$) steps along the rows.
- Thereafter it is shifted by $\lfloor q / \sqrt{p} \rfloor$ steps along the columns. During the circular row shifts, some of the data traverse the wraparound connection from the highest to the lowest labelled nodes of the rows.

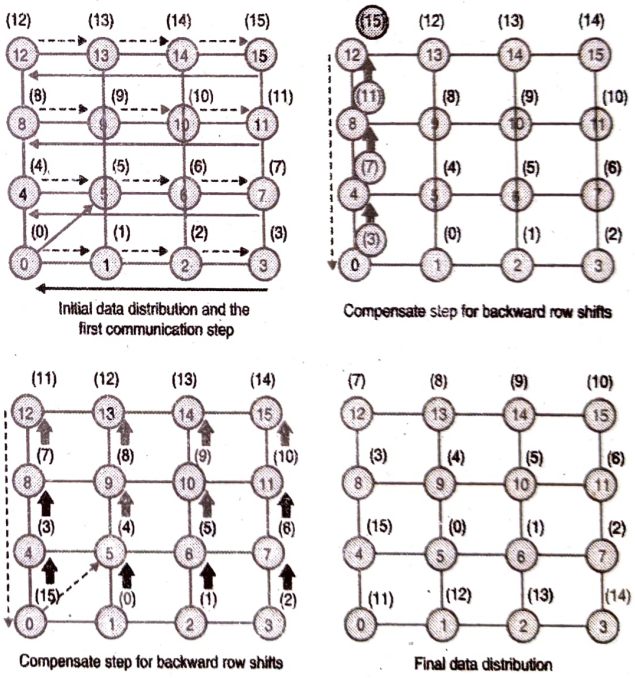


Fig. 3.6.1

3.6.2 Circular Shift on a Hypercube

- A linear array with 2^d nodes map onto a d – dimensional hypercube for developing a hypercube algorithm for a shift operation. For example, expand q shift as a sum of powers of 2 (e.g. 5-shift = 20+22).

– The Fig. 3.6.2 depicts this mapping for eight nodes. The significant property based on this mapping is that, any two nodes at a distance of $2i$ on the linear array are separated by exactly two links on the hypercube. The time for this is upper bounded by : $T = (t_s + t_w m)(2 \log p - 1)$. If E-cube routing is used, this time can be reduced to $T = t_s + t_w m$.

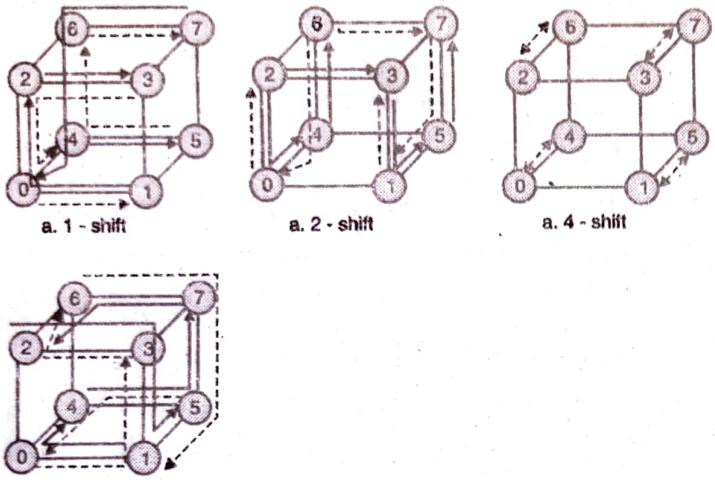


Fig. 3.6.2 : Circular q-shifts on an 8-node hypercube for $1\ q < 8$

Syllabus Topic : Improving the Speed of Some Communication Operations

3.7 Improving the Speed of Some Communication Operations

Q. 3.7.1 Explain different approaches of communication operations. (Refer section 3.7) **(6 Marks)**

Q. 3.7.2 Write short note on
(i) One-to-all Broadcast
(ii) All-to-one reduction
(iii) All Reduce
(Refer section 3.7) **(6 Marks)**

The various approaches we have discussed under communication operations the messages is not divided into several parts. The original message is send as it is and also we have considered each node had a single port for sending and receiving data. Here we will discuss the impact of relaxing these assumptions on some of the communication operations.

3.7.1 One-to-all Broadcast

If the message can be split into p parts, a one-to-all broadcast can be implemented as a scatter operation followed by an all-to-all broadcast operation.

In this approach if we split the message m into p parts one – to – all broadcast becomes as :

scatter + all-to-all broadcast of messages of size m/p

The time for this operation is given as follows:

$$T = 2 \times \left(t_s \log p + t_w (p-1) \frac{m}{p} \right)$$

$$= 2 \times (t_s \log p + t_w m)$$

3.7.2 All-to-one reduction

This operation is a dual of one-to-all broadcast. This implies that an algorithm for all-to-one reduction can be obtained by reversing the direction and the sequence of communication in one-to-all broadcast. This means all-to-one reduction can be performed by performing all-to-all reduction (dual of all-to-all broadcast) followed by a gather operation (dual of scatter).

In this approach if we split the message m into p parts all - to - one reduction becomes as:

all-to-all reduce + gather of messages of size m/p

3.7.3 All Reduce

Since an all-reduce operation is semantically equivalent to an all-to-one reduction followed by a one-to-all broadcast, the asymptotically optimal algorithms for these two operations can be used to construct a similar algorithm for the all-reduce operation. The intervening gather and scatter operations cancel each other. Therefore, an all-reduce operation requires an all-to-all reduction and an all-to-all broadcast.

In this approach if we split the message m into p parts all reduce becomes as:

all-to-all reduction + all-to-all broadcast of messages of size m/p