UNIT-III

Name:- Prof. Ghavu A.N.
Assistant Professor
Computer Dept.
HPC.
8087777708

3. BASIC COMMUNICATION OPERATION

* Introduction to communication operation :

→ In most parallel algorithm, process needs to exchange data with other process. This exchange of data can be significantly impact the efficiency of parallel program by introducing interaction delays during their execution.

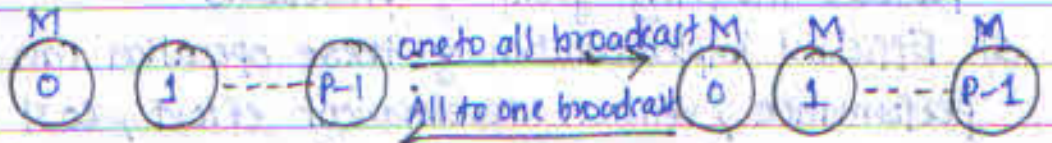
1) Communication operation can be done with help of pattern, related Algorithm and Interconnection network.
2) In Comm, Interconnection n/w can be linear Array, two dimensional mesh and hypercube.
3) The basic pattern of inter process comm are building block of parallel algorithms, which makes their exe. efficient.
4) Many interaction in ||el programs occurs in well-defined patterns involving group of processors.
5) Efficient implementation of these operation can improve performance, reduce development effort, cost and improve software quality.
6) We select descriptive set of architecture to illustrate process of algorithm design.
7) Group communication operation are built using point to point messaging primitives.
8) Comm operation and related time complexities derived for various algorithms. following assumption is to be considered :-
   - Interconnection n/w support cut through routing.
   - The comm time between any node pair is independent of the no of intermediate nodes between them.
   - Communication links are bidirectional. Two directly connected nodes can send the message of size m, to each other in time $t_s + t_w m$.
   - End user does not have control over mapping to process onto processors.

* One to All broadcast and All to One broadcast :-

This type of operation needs for matrix-vector multiplication, shortest path, Gaussian elimination and vector inner product.

- One to All Broadcast :-

1) One to all broadcast is the operation in which a single process send identical data to all other processes.
2) parallel algorithms always needs this operation.
3) Let's consider that data of size m is to be sent to all the processes.
   - Initially always source process has the data.
   - After termination of algorithm there will be copy of initial data with each process.
   - P copies of data will be generated whereas P is the number of processor as shown in fig.



One-to-All and All-to-One Operation.
One to All Broadcast & All to One Reduction.

- All to One Reduction :-

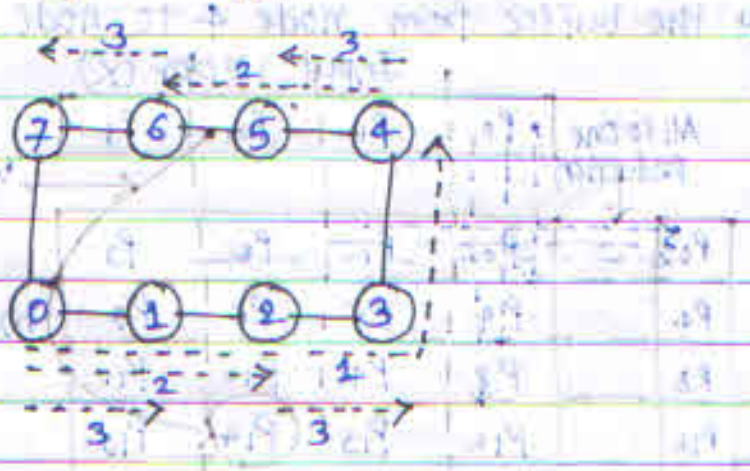
1) All to One reduction is the operation in which data from all processes are combined at single destination process.
2) Various operation like Sum, product, max & min of a set of numbers can be performed by All to one reduction.
3) Each processor P will have buffer M which contains m words.
4) After termination of algorithm, the ith word of final buffer M_i will contain the collected result of Sum, product, maximum & minimum of the ith word of each of the buffer.

above Fig. shows All to One Reduction.

* Implementation of One to All Broadcast and All to One Reduction.
On Interconnection topologies
Ring or Linear Array (1D), Mesh(2D), Hypercub (3D) etc.

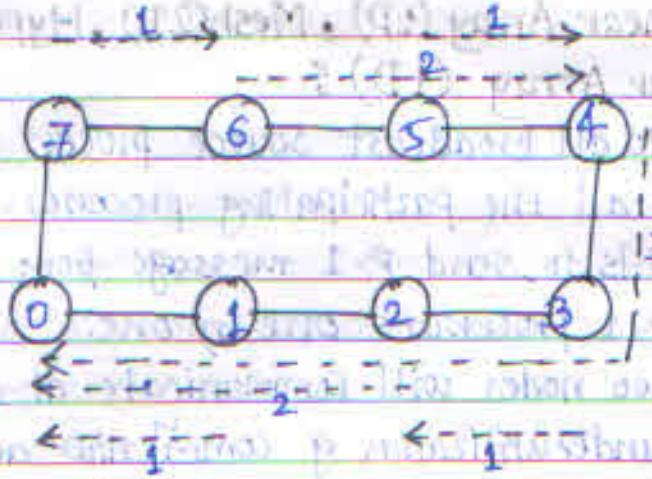
1] Ring or Linear Array (1D) :-

→ 1) In this, One to all broadcast source process sends the copy of data to all the participating processes.
2) Simplest way is to send P-1 message from the source to the other P-1 processors one by one.
3) By this only two nodes will communicate at a time. resulting in underutilization of comm. n/w and also Source process becomes bottlenecks.
So this is not very efficient.
4) This problem is overcame by "Recursive Doubling" broadcast Algo. can be made more efficient by the tech is called as Recursive Doubling.
5) In Recursive doubling techniques, the source process 1st send the menage to another process.
6) Now, there will be two copies of the menage which can be simultaneously sent to two other processes that are still waiting for the message.
This process will be continued till all processes receive data.
7) menage can be broadcast to all the nodes in log P step.
Fig: shaw steps in one to all broadcast on an 8-node linear array or ring. nodes are labelled from 0 to 7.



One-to-All broadcast on a eight node ring.

* Reduction on Linear Array :-



Reduction on 8-node ring with node 0 as dest g Reductia.

1) To perform reduction on a linear array the direction and sequence g communication is to be reversed.
2) As shown in figure, in the 1st step, each odd numbered nodes sends it's buffer to the even numbered node just before itself.
   e.g. 1 to 0, 3 to 2 etc
3) The content g these two buffers is combined into one.
4) As a result, we get 4 buffers on nodes 0, 2, 4 and 6.
5) In the 2nd comm step, the contents g the buffer on node 0 and 2 are accumulated on node 0 and those on node 6 and 4 are accumulated on node 4.
6) At last, final result g reduction will be computed by sending the buffer from node 4 to node 0.

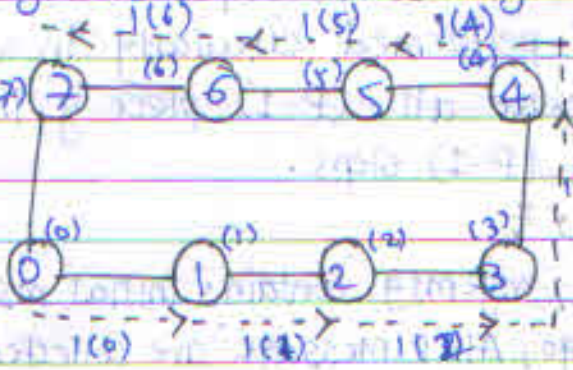
Input vector (x)

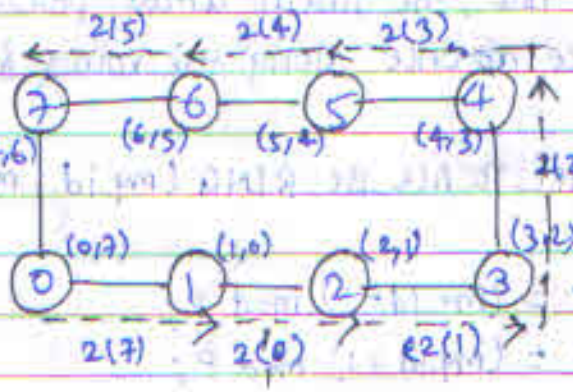
* Linear Array and Ring in All-to-All Broadcast

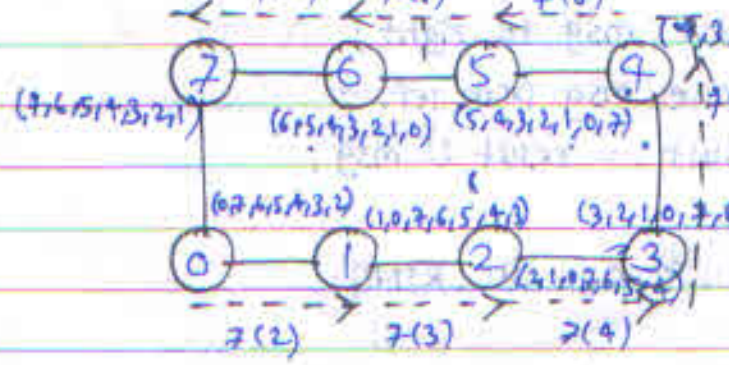
→ 1) While performing all-to-all broadcast on linear Array, all comm link can be kept Busy simultaneously until the oper is complete bcz each node always has some information that it can pass along to it's neighbor.

2) Each node 1st sends to one of it's neighbors the data it needs to broadcast.

3) In subsequent step, it forward the data received from one of it's neighbors to it's other neighbours.







All-to-All Broadcast on a eight node Ring.

- The initial source of message is identified by the label in parenthesis along with time step.
  e.g. The label on the arrow from node 1 to 2 i.e. 1(1) indicates that in time step 1 the message (1) which is present with the node-1 will be sent to node-2.
- The number (s) in parentheses next to each node are the labels of node from which data has been received before the current comm step.
- As the comm performed circularly in a single direction, each node receive all (P-1) piece of info from all other node in (P-1) steps.

Detailed Algorithm :=

1) At every node, my_msg contains initial msg to be broadcasted
2) At the end of algo, All P message are collected at each node

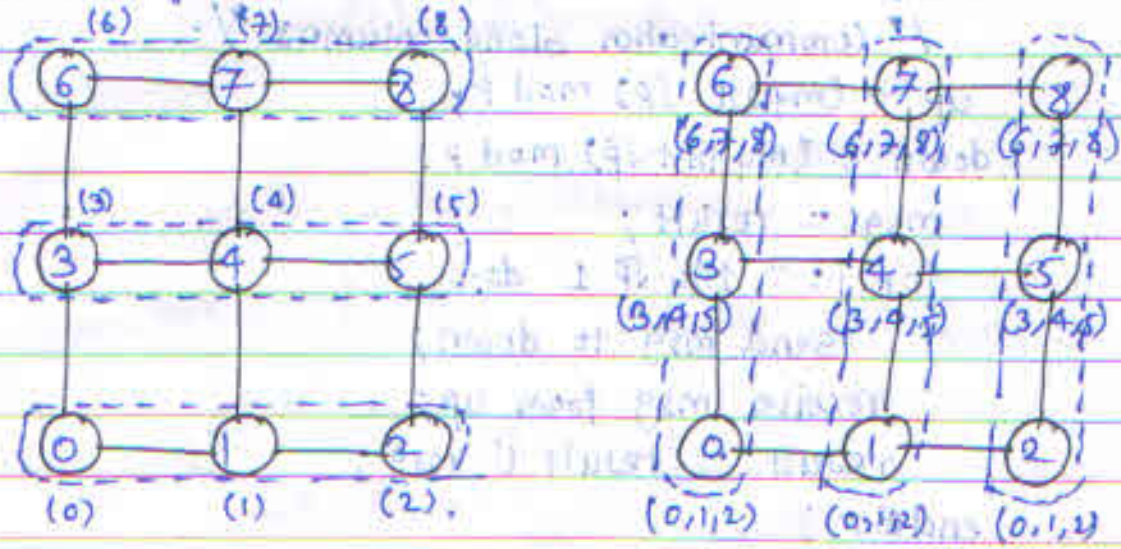
- For mesh comm nw, the linear array procedure has to be carried out twice: once along the rows & once along the columns

```
procedure ALL_TO_ALL_BC_RING (my_id, my_msg, P, result)
  begin
    left := (my_id - 1) mod P;
    right := (my_id + 1) mod P;
    result := my_msg;
    msg := result;
    for i := 1 to P-1 do
      Send msg to right;
      receive msg from left;
      result := result U msg;
    endfor;
  end ALL_TO_ALL_BC_RING.
```

All-to-all broadcast on a P-node ring.

* Mesh (All to All) :

- Similar to one to all broadcast, the All-to-All broadcast algo for the 2-D mesh is based on the linear array algo.
- Comm take place in two phase as shown in fig.
- In the 1st phase, each row of the mesh platforms an all-to-all broadcast using the algorithm for linear array.
- $\sqrt{p}$ message are collected from the $\sqrt{p}$ nodes of respective rows.
- These menage are consolidated into a single msg of size $m\sqrt{p}$.
- In the 2nd comm phase column wise all-to-all broadcast of the consolidated msg is done.
- After completion of 2nd phase, each node obtain all p pieces of m-word data i.e. all node will get (0,1,2,3,4,5,6,7) (i.e. menage from each node)



(a) Initial Data Distribution (b) Data Distribution After rowwise Broadcast

ALL-to-ALL Broadcast on a 3×3 mesh.

```
procedure ALL_TO_ALL_BC_MESH (my_id, my_msg, P, result)
begin
    /* comm along row */
  left := my_id - (my_id mod √P) + (my_id - 1) mod √P;
 right := my_id - (my_id mod √P) + (my_id + 1) mod √P;
  result := my_msg;
   msg := result;
   for i := 1 to √P - 1 do
       Send msg to right;
     receive msg from left;
      result := result ∪ msg;
      endfor;
       /* Communication along columns */
     up := (my_id - √P) mod P;
   down := (my_id + √P) mod P)
      msg := result;
      for i := 1 to √P - 1 do
          send msg to down;
        receive msg from up;
        result := result ∪ msg;
  endfor;
 end ALL_TO_ALL_BC_MESH.
```

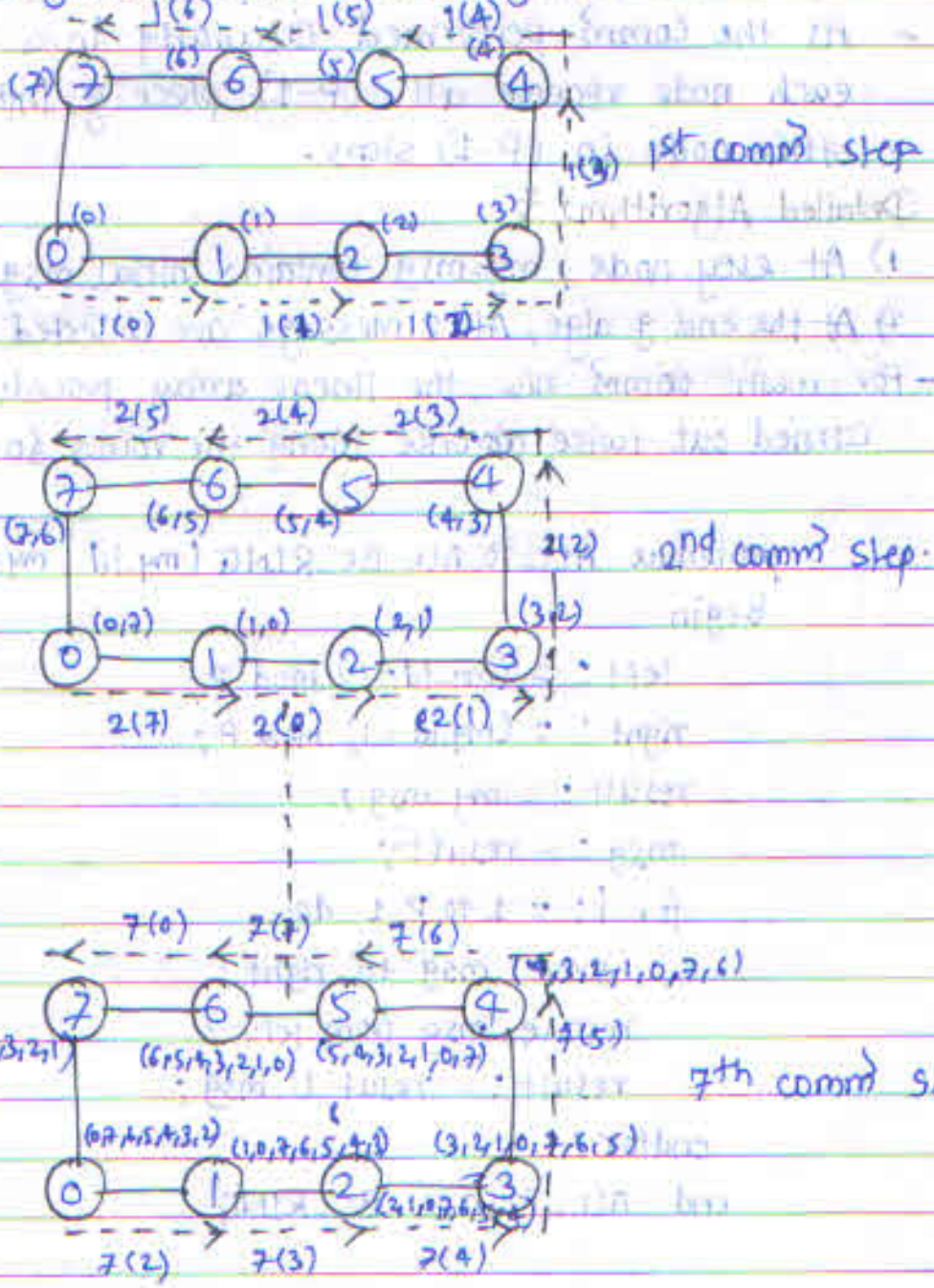
ALL TO ALL Broadcast on a Square Mesh of P nodes.

* Linear Array and Ring in All-to-All Broadcast

→ 1) While performing all-to-all broadcast on linear Array, all comm link can be kept Busy simultaneously until the operation is complete bcz each node always has some information that it can pass along to it's neighbor.

2) Each node 1st sends to one of it's neighbors the data it needs to broadcast.

3) In subsequent step, it forward the data received from one of it's neighbors to it's other neighbours.



All-to-All Broadcast on a eight node Ring.

- The initial source g message is identified by the label in parenthesis along with time step.
e.g. The label on the arrow from node 1 to 2.
i.e. 1(1) indicates that in time step 1 the message (1) which is present with the node-1 will be sent to node-2.
- The number (s) in parentheses next to each node are the labels g node from which data has been received before the current comm step.
- As the comm performed circularly in a single direction, each node receive all (P-1) piece g. infor from all other node in (P-1) steps.

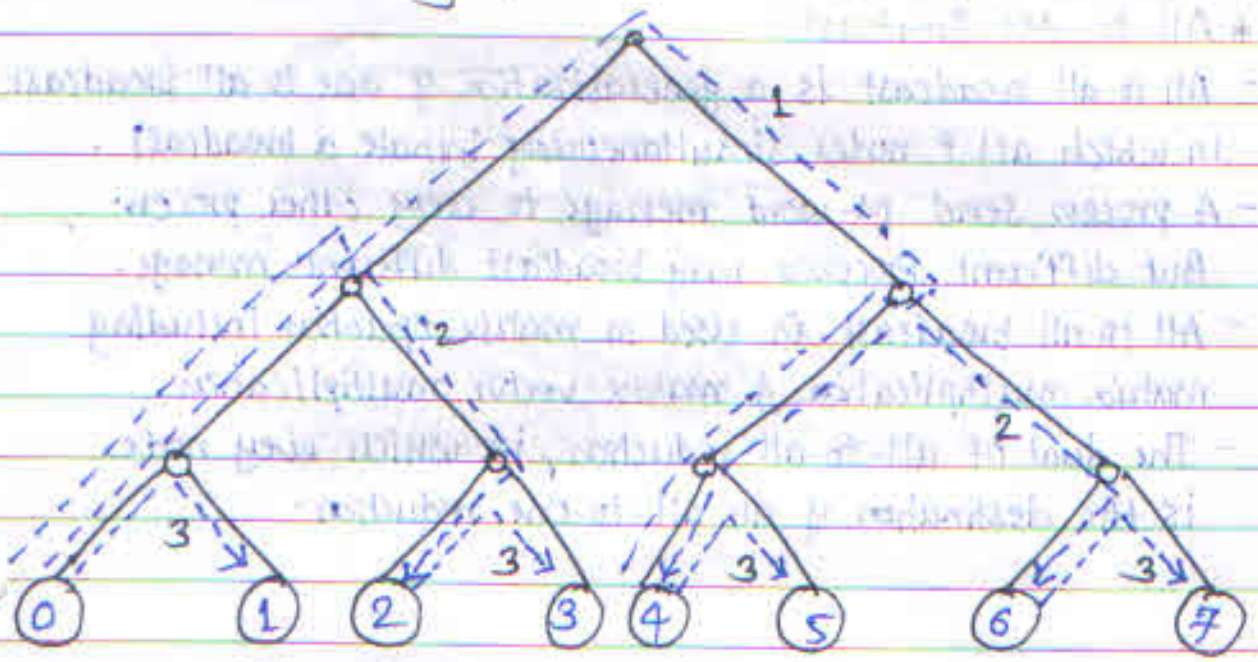
Detailed Algorithm:

1) At every node, my_msg contains initial msg to be broadcasted
2) At the end g algo, All P message are collected at each node
- For mesh comm nw, the linear array procedure has to be carried out twice: once along the rows & once along the columns

```
procedure ALL_TO_ALL_BC_RING (my_id, my_msg, P, result)
begin
    left := (my_id - 1) mod P;
    right := (my_id + 1) mod P;
    result := my_msg;
    msg := result;
    for i := 1 to P-1 do
        send msg to right;
        receive msg from left;
        result := result ∪ msg;
    endfor;
end ALL_TO_ALL_BC_RING
```

All-to-all broadcast on a P-node ring.

* Balanced Binary Tree :-



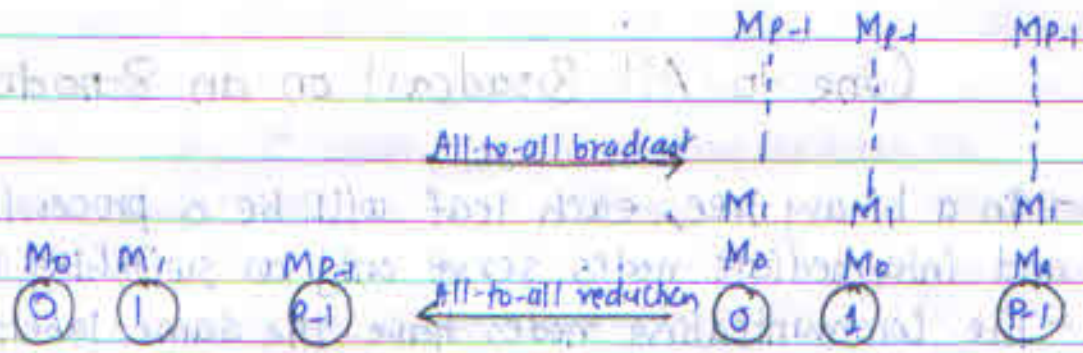
One to All Broadcast on an 8-node tree.

- In a binary tree, each leaf will be a processing node. and intermediate nodes serve only as switching units.
- The communicating nodes have the same labels as in the hypercube algorithms.
- The communication pattern will be the same as that of hypercube algorithms.
- There will not be any congestion on any of the communication links at any time
- On diff. paths, different number of switching nodes will be. there making it's communication different from hypercube.

* All to All Broadcast and Reduction ∴

→ * All to All Broadcast :

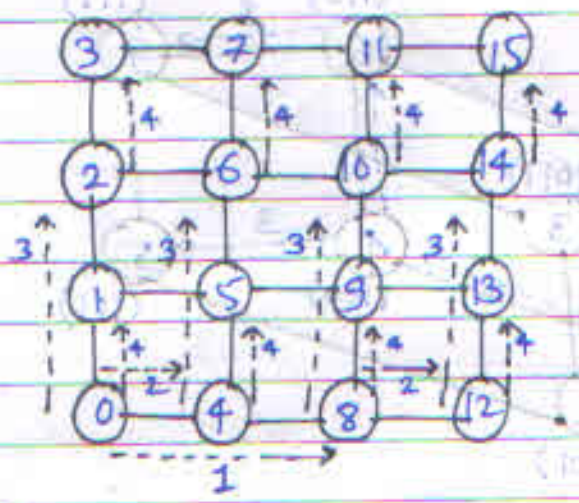
- All to all broadcast is a generalization of one to all broadcast in which all P nodes simultaneously initiate a broadcast.
- A process send M-word message to every other process, But different processes may broadcast different message.
- All to all broadcast is used in matrix operation Including matrix multiplication & matrix-vector multiplication.
- The dual of all-to-all reduction, in which every node is the destination of an all-to-One reduction.



All-to-All Broadcast & All-to-All Reduction.

- One approach to perform all to all broadcast is performing P one-to-all broadcast.
- This approach may take P times to complete comm.
- Comm. link can be used more efficiently by simultaneously performing all P one-to-all broadcast.
- By this, there will be contention of all messages traversing the same path at the same time into a single message.

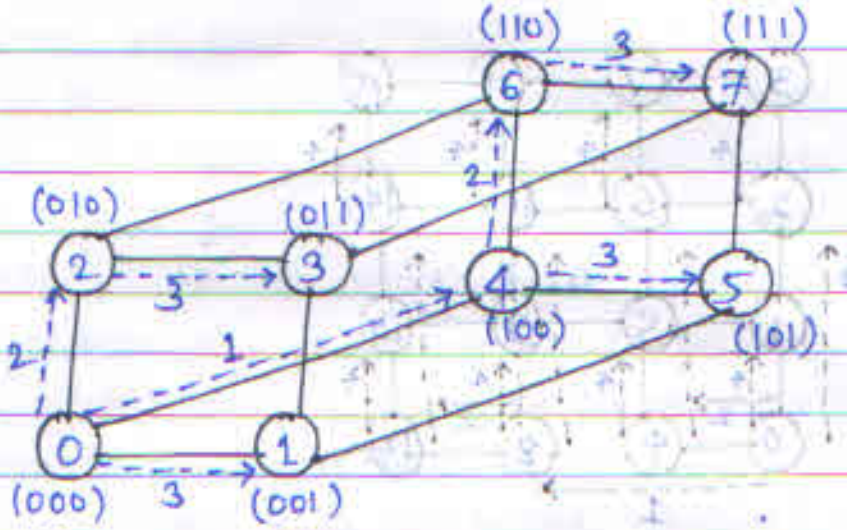
* Mesh :-



One to All broadcast on a 16-node mesh.

- Each row and col of square mesh of P node can be considered as a linear Array of $\sqrt{P}$ nodes.
- There will be 2 phases of comm
  1) 1st step, oper is performed along one or all rows by considering the row as linear arrays
  2) 2nd step, same process will be carried out for column.
- For example, Consider 2-D square mesh with $\sqrt{P}$ row & $\sqrt{P}$ col. for one to all broadcast operation
- 1st ly, the data is sent to remaining all th $(\sqrt{P}-1)$ node in row by one to all broadcast by the source.
- 2nd phase, the data will be sent to respective column by One to all broadcast.
- Thus, each node of the mesh will have the copy of Initial message by the end of 2nd phase.
- Consider the example of 16 node mesh: the source node 0 & data is to be sent to all nodes.
- In diag, 1st phase is shown by 1 & 2 on directed arrows. At end of this phase, all nodes (0,4,8,12) in 1st row will have data.
- 2nd phase, data will be sent to respective col shown by step 3 & 4.
- Similar process for one to all broadcast on 3-D mesh
- Reduction process in linear array can be carried out on two and 3-D meshes as well by reversing the direction & order of message.

* Hypercube



One to All broadcast on 3-D Hypercube.

- In 2D mesh, comm take place in 2 phase for 2 diff dimensions
- In 3D mesh, the same process will be carried out in 3-D.
- Hypercube with 2^d nodes can be considered as a d-dimensional mesh with two nodes in each dimension.
- mesh comm algo can be implemented by carrying out d steps, one in each dimension.
- Consider example of one-to-all broadcast on eight-node (3-D) hypercube with node 0 as source.
- In 8-node hypercube each node is identified by a unique label of three digits.
- Comm starts along the highest dimension specified by MSB of node label.

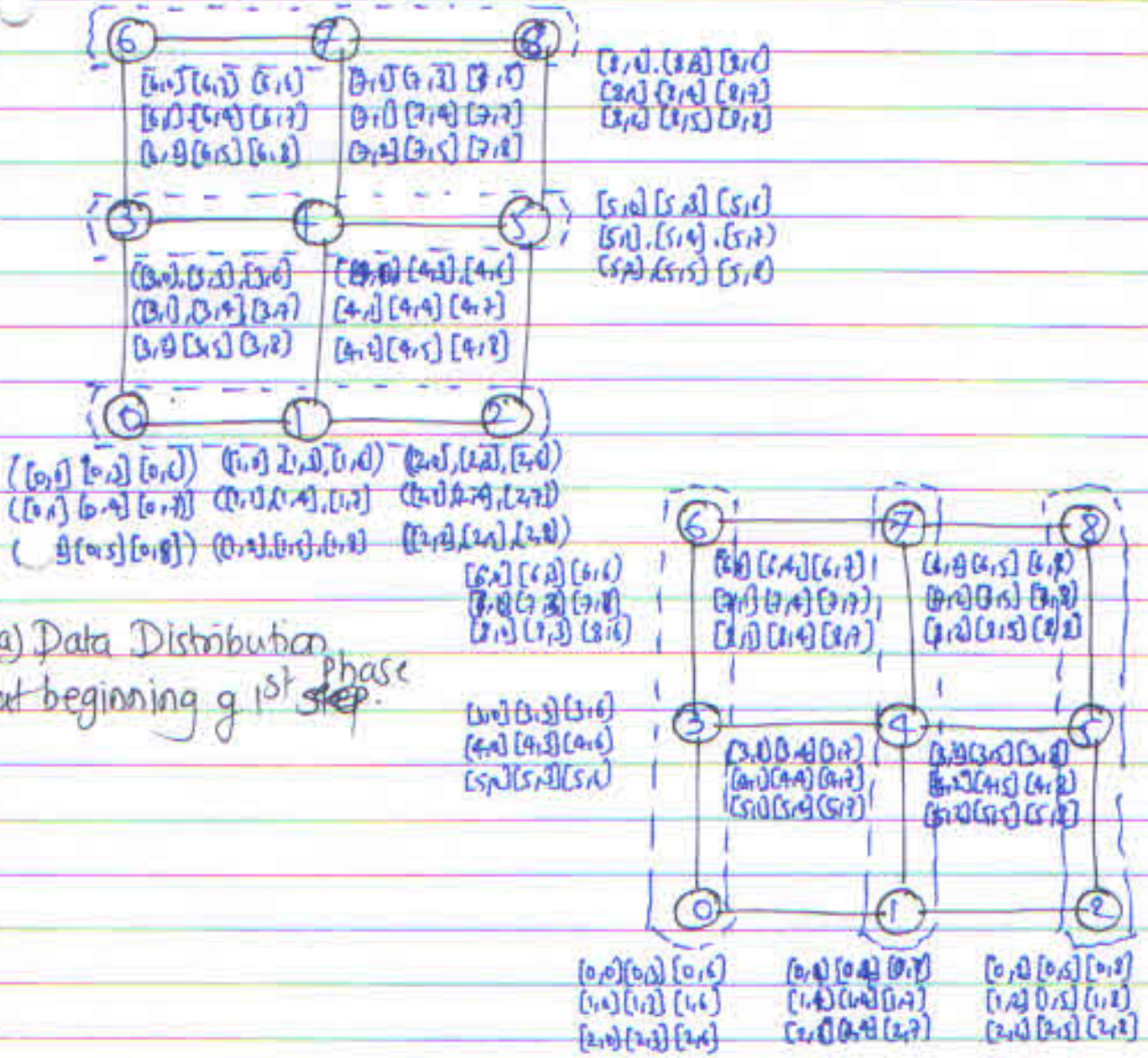
For example, in step 1 node 0 (000) will send data to node4(100) with higher dimension.

- In next steps comm will be done for the lower dimension.
- The source and the destination nodes in three communication steps of algo are similar to nodes in the broadcast algo on linear array.
- Hypercube broadcast will not suffer from congestion if node 0 sends message to node 1 in the first step, followed by node 0 and 1 sending message to nodes 2 and 3, respectively & finally nodes 0, 1, 2 & 3 sending message to nodes 4, 5, 6, 7 respectively.

* Mesh (All-to-All Broadcast) *

→ - For all-to-all personalized comm on a mesh $\sqrt{p} \times \sqrt{p}$, at each node group of p msg is formed considering columns of destination nodes.

- As shown in fig, for 3×3 mesh, each node will have nine m-word messages one for each node.
- For each node three groups of three msg are formed.
- The 1st group contains the msg for destination node labeled 0, 3 and 6; second group contains the msg for node 1, 4 & 7. and last group has msg for node labeled 2, 5 and 8.



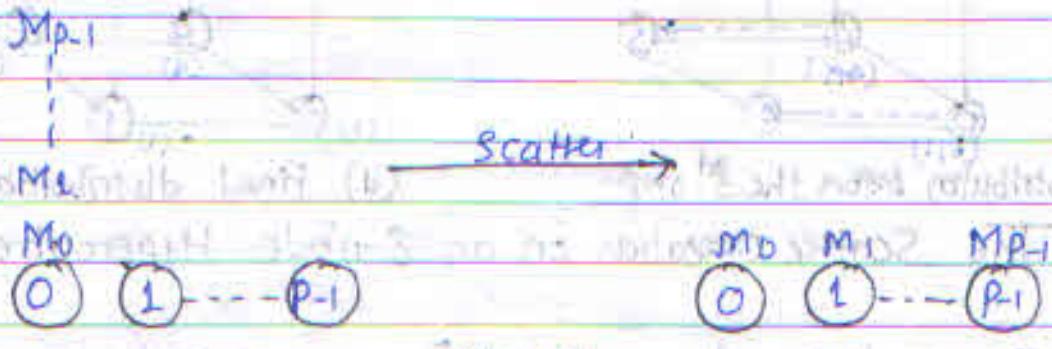
(a) Data Distribution at beginning of 1st phase

Data Distribution at the beginning of second phase

All to All personalized comm on 3×3 mesh.

* Scatter and Gather op

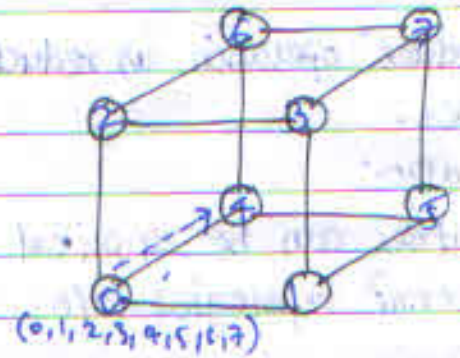
- In one-to-all broadcast src node sends the same data to all the P nodes, resulting in duplication of data.
- In scatter opn, a single node sends a uniques msg of size m to every node.
  This is also called as One-to-all personalized Comm.
- Gather opn or Concatenation is the dual scatter operation.
- In Gather opn, a single node collect a unique message from each node.
- Gather is different from all-to-One reduce operation as reduction or combination of data does not take place.
- Scatter algo. is similar to broadcast algo.
- The hypercube algo for scatter & gather can be applied to linear array and mesh interconnn topologies w/o any increase in the comm time.
- Consider e.g of 8-node hypercube.
- The comm patterns of all-to-all broadcast & scatter are identical, the only difference in size & content of msg.
- As shown in below fig. Initially src node 0 will have all the message



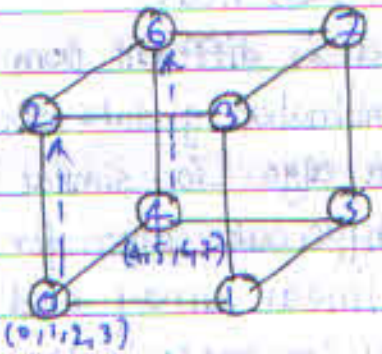
SCATTER & GATHER OPERATION

- In 1st Comm step, node 0 transfers half of the msg to one of it's neighbors (node 4).
- In next step, if any node has some data, it transfer half of the to one of it's neighbors who has not received any data upto now.

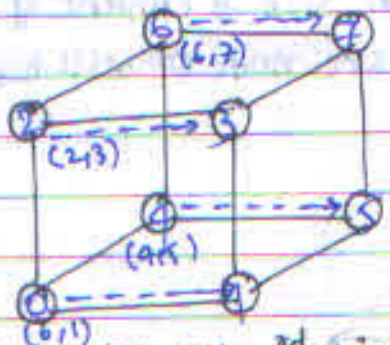
- This process involves log p comm step for log p dimension of the hypercube.
- The gather operation is reverse of scatter operation.
- Every node will have m word message
- In the 1st step, each odd numbered node send its buffer to an even numbered neighbor behind it.



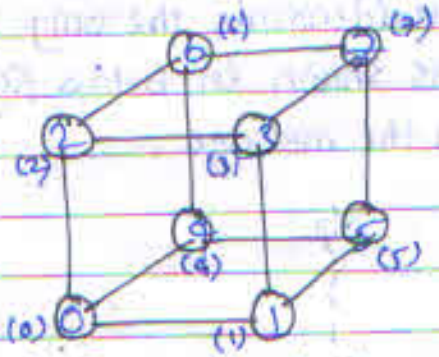
(a) Initial Distribution of message.



(b) Distribution before the 2nd step.



(c) Distribution before the 3rd step.



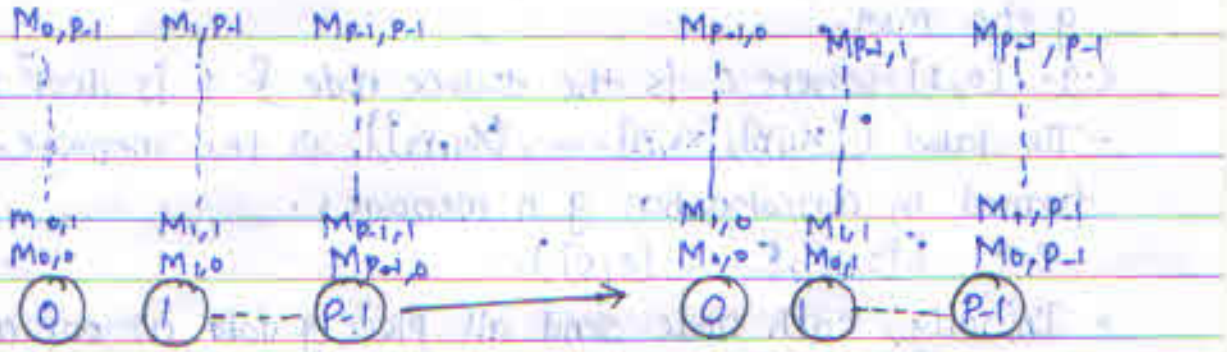
(d) Final distribution of msg.

The Scatter operation on an 8-node Hypercube.

- The neighbor node concatenates the received message with its own buffer.
- In next comm step, only even numbered nodes participate in comm.
- The node with multiple of 4 labels gather more data and double the size of their data.
- This process is continued till node 0 gather all

* All-to-All Personalized Communication:

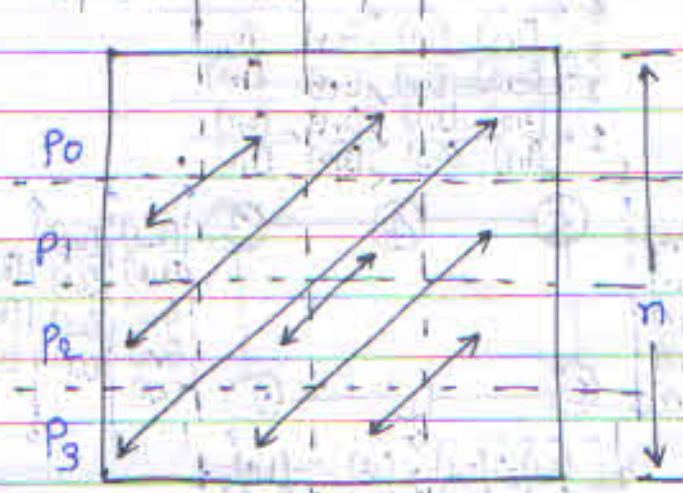
- All to all personalized Comm opn can be applied in variety of ||el algo such as fast Fourier transform, matrix transpose, sample sort and some IM d/b join operation.



ALL TO ALL Personalized Communication.

- In all-to-all broadcast every node sends same message to all the nodes.
- In contrast in all-to-all personalized Comm every node sends a distinct msg of size of m to every other node.
- So it is called as total exchange.
- This opn is similar to transposing a 2-D array of data distributed among P processes using one-Dimensional array partitioning.

* Matrix transposition Example:



All-to-One Personalized Comm in transposing a 4x4 matrix using 4 processes

- Fig. shows the example of 4x4 matrix mapped onto 4 processes using 1-D rowwise partitioning.

* Ring (All to All personalized comm)

→ - As shown in fig, every node sends P-1 piece of data, each of size m.

- These piece are identified by label (x,y), x is label of node that originally own the msg & y is label of final destination of the msg.

e.g: [0,1] where 0 is the source node & 1 is dest node.

- The label [[x1,y1],[x2,y2]---,[xn,yn]] is the menage formed by concatenation of n menages.

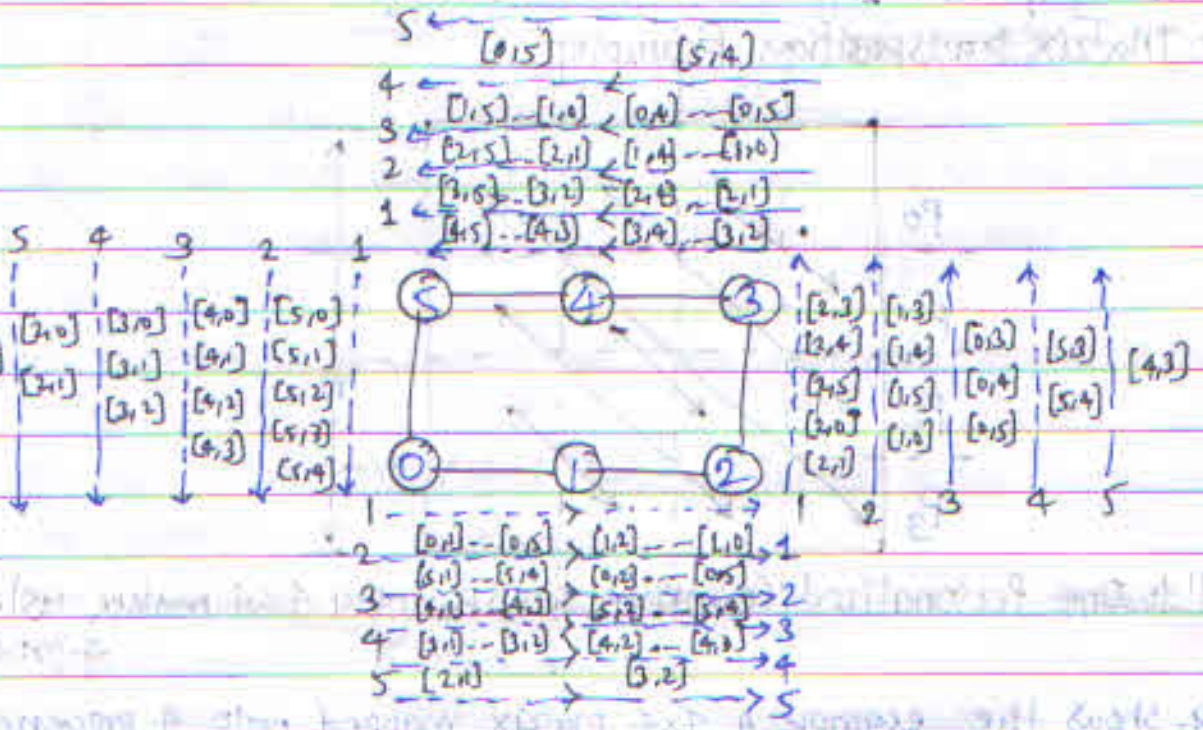
e.g: ([0,1],[0,2]---[0,5]).

- Initially, each node send all piece of data as one consolidated msg of size m(P-1) to one of its neighbor.

e.g. In time step 1, node 0 sends consolidated menage ({0,1},---{0,5}) to node 1.

- From received menage size m(P-2) only 1-m word packet which belongs to it will be kept by the neighbor node. Remaining (P-2) piece will be forwarded to next node.

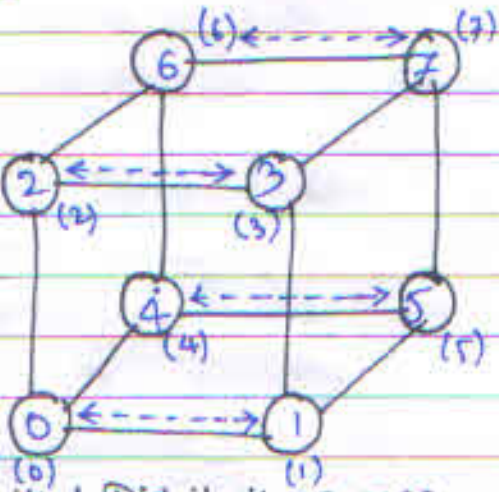
e.g. In time step 1, node 1 will keep the msg from node 0 & forward the remaining packet ({1,2}---{1,0}) to the next neighbors.



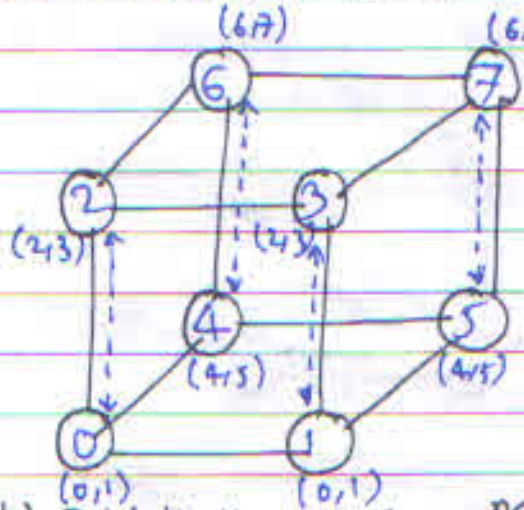
All-to-All personalized Comm on 6-node Ring

* Hypercube (ALL TO ALL Broadcast)

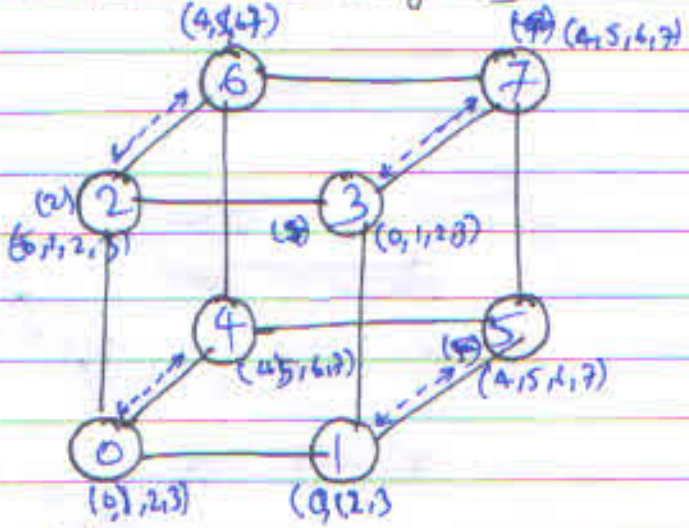
→ - All to all broadcast operation can be performed on hypercube by implementing mesh algo. to $\log p$ dimension.
- In each step, for different dimension comm^n is carried out. for e.g.,
As shown in fig, comm^n is carried out in each row in 1st step. in fig (b) comm^n happens columnwise in 2nd step.
- As shown in fig. pair of nodes exchange data in each step.
- Received msg is concatenated with the current data in each step.
- Thus, size of msg gets double which will be transmitted in next step.
- Hypercube with bidirectional comm^n channel is considered.



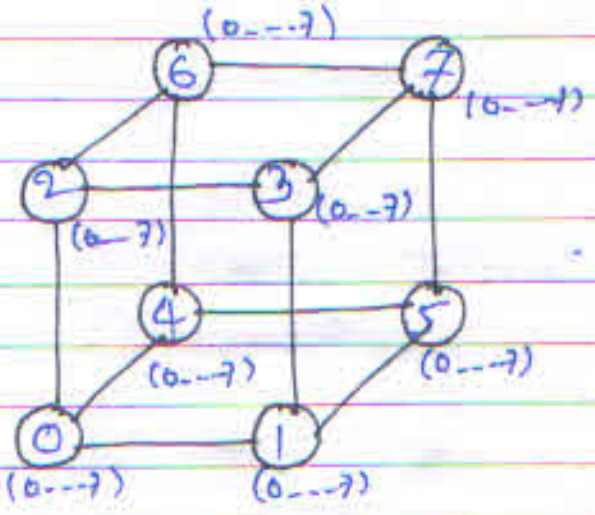
(a) Initial Distribution of msg



(b) Distribution before 2nd step.



(c) Distribution before the 3rd step

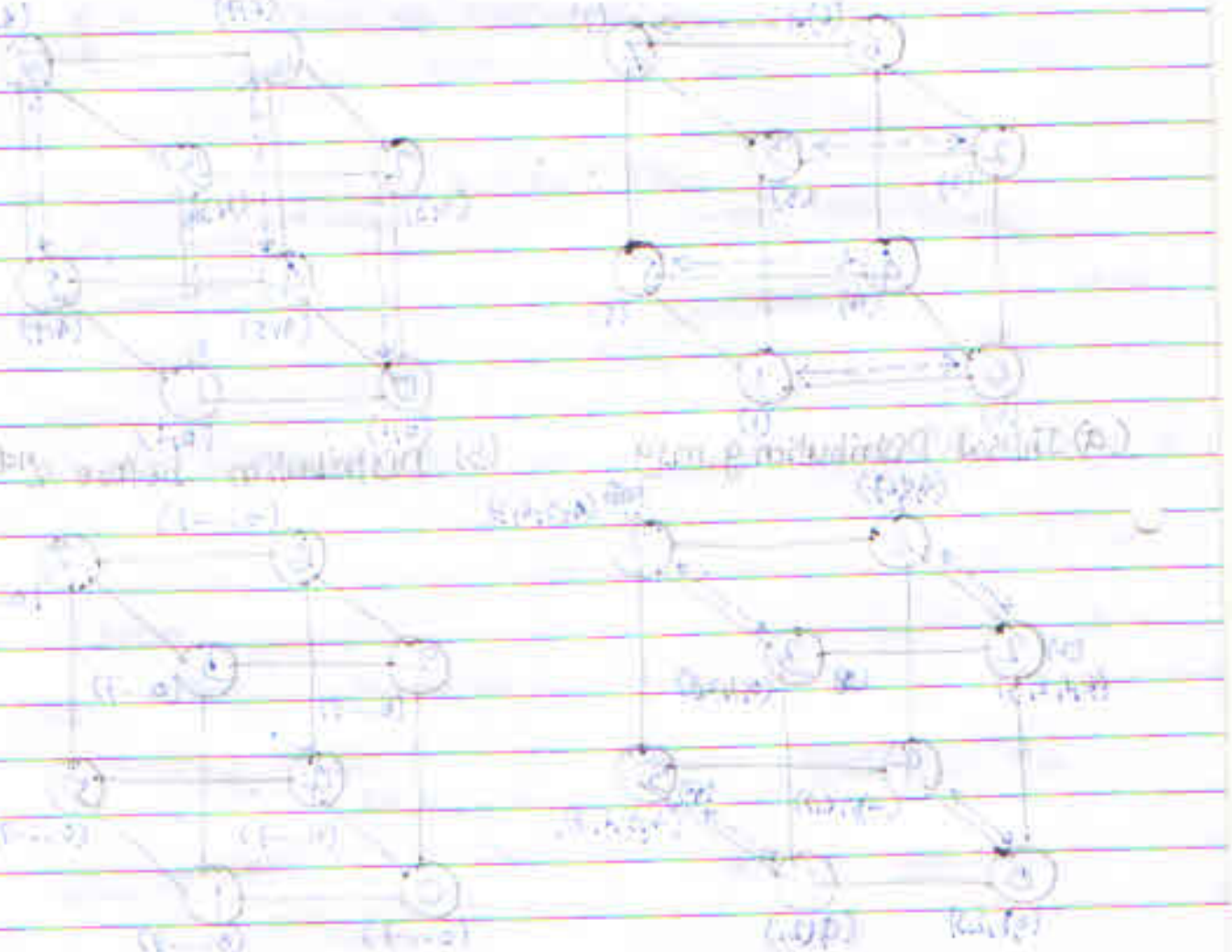


(d) Final Distribution of Message

ALL-TO-ALL Broadcast on an 8-node Hypercube.

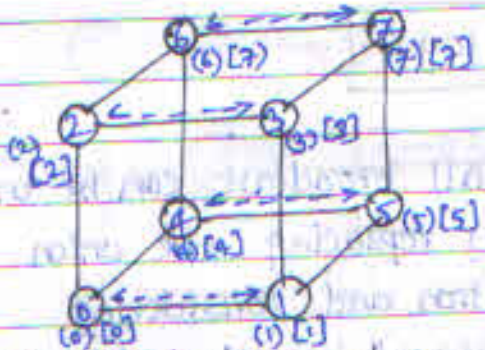
Let's consider algo. All to All broadcast in Hypercube :-

```
procedure ALL_TO_ALL Broadcast_HCUBE (myid, my_msg, d, result)
begin
  result := my_msg;
  for i := 0 to d-1 do
    partner := myId XOR 2^i;
    Send result to partner;
    receive msg from partner;
    result := result U msg;
  endfor;
end ALL-TO-ALL-BC-HCUBE.
```

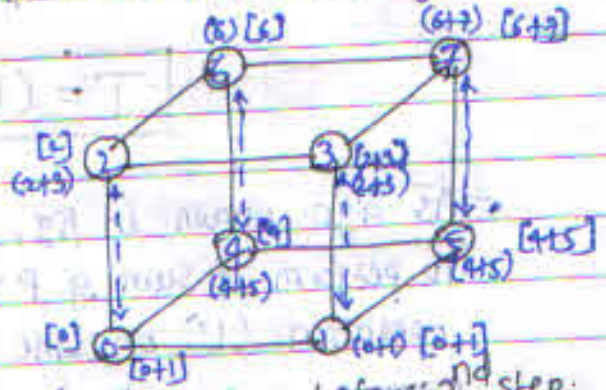


* All-Reduce and Prefix Sum Operation :

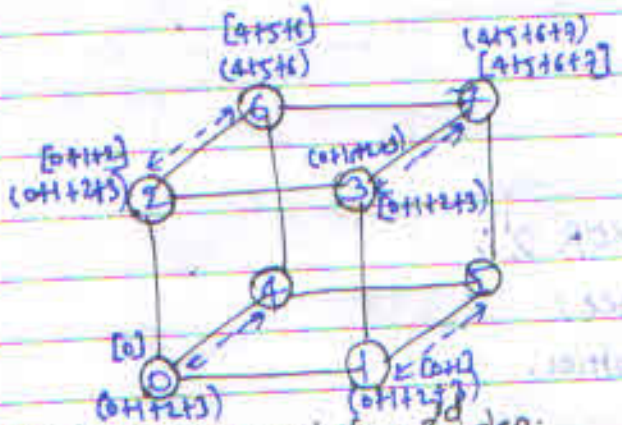
- An all reduce oper with single word message on each node can be implemented as barrier synchronization on a message-passing computer as reduction cannot be completed before contribution of the value from each node.
- In All reduce operation, each node will have buffer of size m.
- After oper identical buffer of size m are formed by combining the original P buffer using an associative operator at each node.
- It is similar to performing an all-to-one reduction followed by one to all broadcast result.
- It is different from All-to-All Reduction.
- It can be made faster by using comm pattern of all-to-all broadcast.
- Consider the algo for 8-node hypercube as shown in fig.



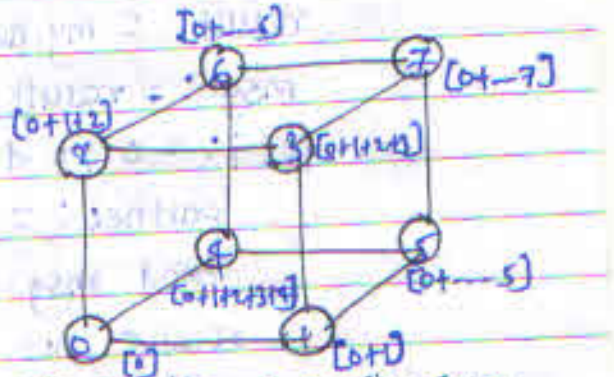
(a) Initial Distribution of value



(b) Distribution of sum before 2nd step.



(c) Distribution of sum before 3rd step.



(d) Final Destination of prefix sum.

Prefix Sum Computation for an 8-node Hypercube.

- Each node will have integral label & integer in parentheses denotes a number to be added which is residing at the node for e.g. 5[5] first 5 shows label and 5 in parentheses shao no to be added
- Each msg which will be transferred in the reduction oper has only one word
- Comm step g the all to all broadcast algo. one used but instead g concatenating two messages, two nos are added
- when algo. terminates, each node will hold sum(0+1+2--7)
- In each step, size g msg is not doubled as instead g concatenation the message are added
- Total Comm time for all log p step is,

$$T = (t_s + t_{wm}) \log p$$

- As algo. shown in fig, for all-to all broadcast can be used to perform a sum g P numbers by replacing the union operation (U) on Line 8 by addition and considering my_msg, msg and result are numbered instead g message.

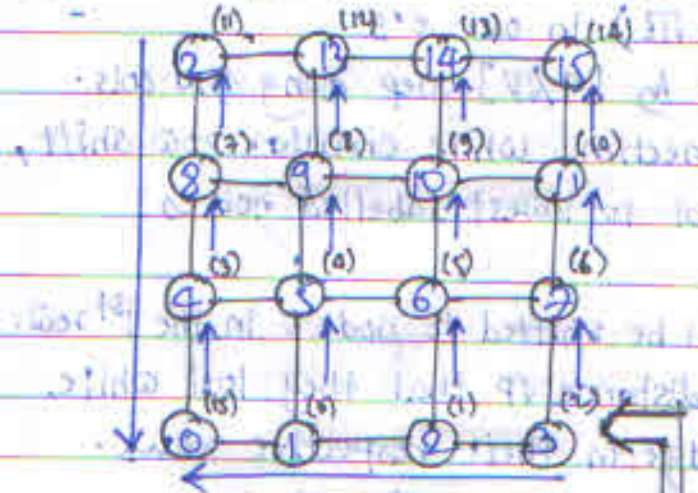
```
procedure PREFIX_SUM_HCUBE (my_id, my_number, d, result)
begin
    result := my_number;
    msg := result;
    for i := 0 to d-1 do
        partner := my_id XOR 2^i;
        send msg to partner;
        receive nos from partner.
        msg := msg + nos.
        if (partner < my_id) then result := result + number;
endfor;
end PREFIX_SUM_HCUBE;
```

* Circular Shaft :-

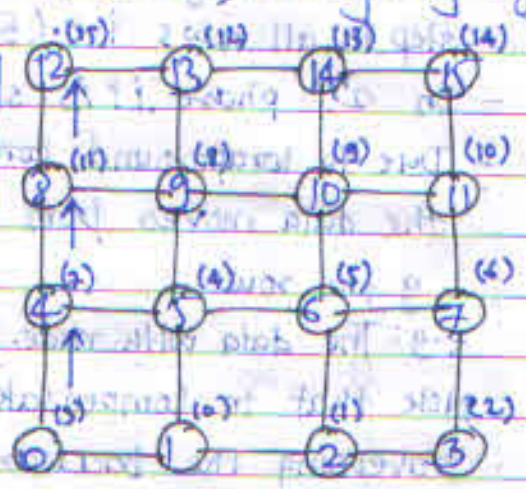
- Circular shaft can be applied in some matrix computations and in string and image pattern matching.
- It is a member of broader class of global Comm operation.
- So in circular q-shift node i sends data to node $(i+q) \bmod p$ in a group of p node where $(0 < q < p)$. known as permutations.
- In permutation, every node sends a message of word to a unique code.

MESH :-

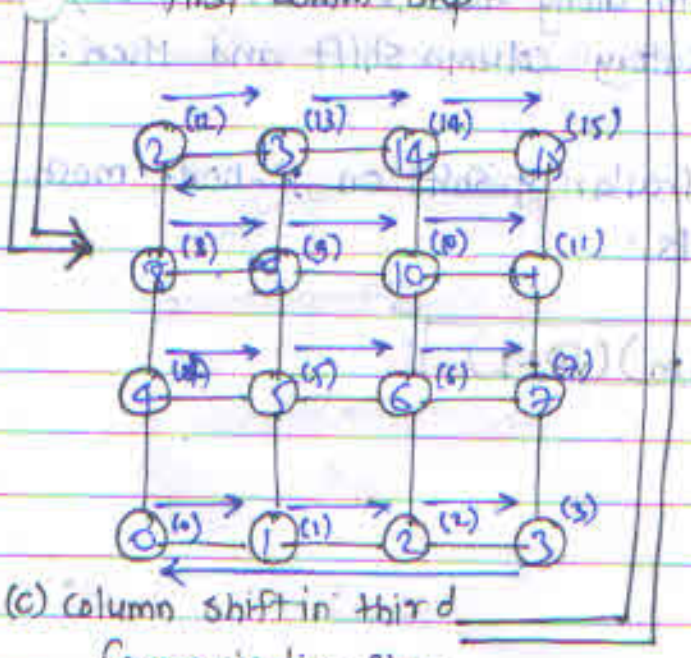
- mesh algo for circular shift can be derived by using Ring Algorithm



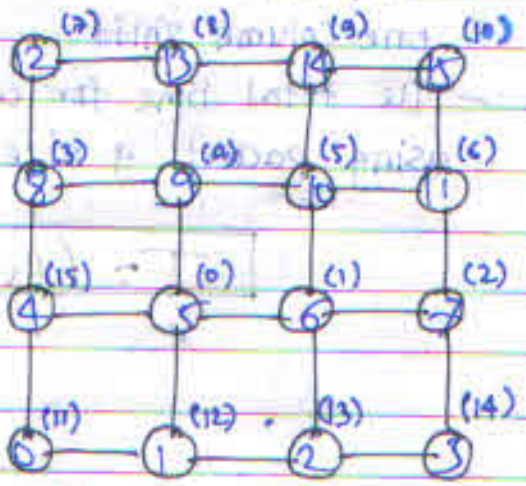
(a) Initial Data Distribution and the first Comm step.



(b) Step to compensate for backward row shifts.



(c) Column shift in third Communication Step.



(d) Final Distribution of data.

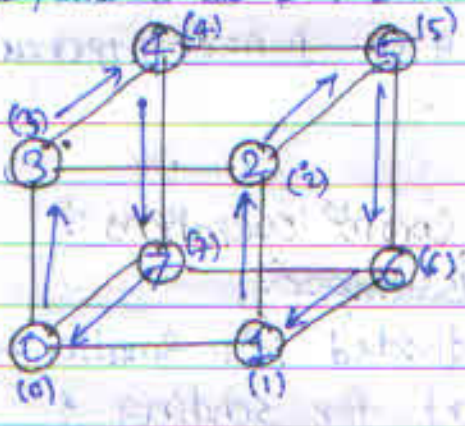
The Communication steps in a Circular 5 shift on a 4×4 mesh.

- Note that as shown in fig. wrap around connections are considered in mesh i.e. in a row of 4 nodes 0, 1, 2, 3, node 3 can communicate and send data to node 0.
- Implementation can be performed by $\min\{q, p-q\}$ neighbor-to-neighbor comm in one direction, where p is number of node and q is the no. of shift to be performed.
- In a p-node square wraparound mesh, for nodes with row major labels, a circular q-shift is performed in 2 stages.
- Consider example for a circular 5-shift shown in fig where $q=5$ and $p=16$ (4×4 mesh)
- In the 1st stage the data is shifted simultaneously by $(q \bmod \sqrt{p})$ step in all rows i.e. $(5 \bmod \sqrt{16})$ in our e.g.
- In 2nd phase it is shifted by $\lfloor q/\sqrt{p} \rfloor$ step along the cols.
- Due to wraparound connection, while circular row shift, the data moves from highest to lowest labelled nodes of the rows.
e.g. The data with node 3 will be shifted to node 0 in the 1st row.
- Note that to compensate for distance $\sqrt{p}$ that they lost while traversing the backward edge in their respective rows, the data packet must be shifted by an additional step.
- As shown in e.g., the data along rows is shifted first, next, there will be compensatory column shift and then one column shift.
- The total time for any circular q-shift on p-node mesh using packet of size m is:

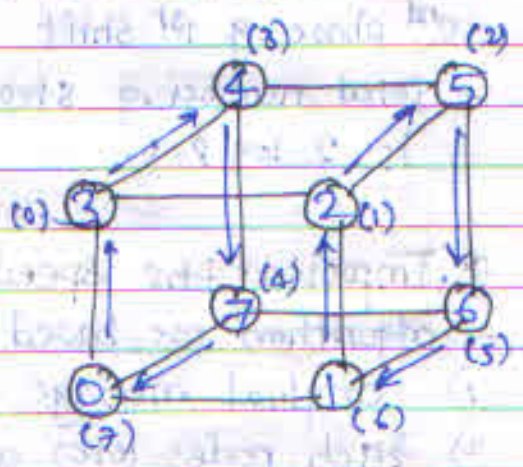
$$T = (t_s + t_w m)(\sqrt{p} + 1)$$

* HYPERCUBE (Circular shift) :-

- For shift operation on hypercube linear array with 2d nodes is mapped onto d-dimensional hypercube.
- Node i of the linear array is assigned to node j of the hypercube where j is d-bit binary Reflected Gray Code (RGC) of i.
- Consider 8-nodes hypercube shown in fig.
- As shown in fig., any two node at distance 2^i are separated by exactly two links.
- for i = 0 nodes are directly connected so this is the exception as only one hypercube link seperates two nodes.
- for q shift operation, q is expanded as a sum of distinct powers of 2. For e.g. the number 5 can be expanded as $2^2 + 2^0$.

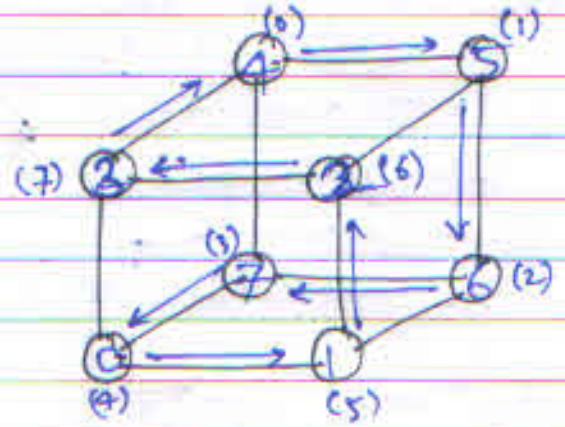


First Comm step of the 4-shift

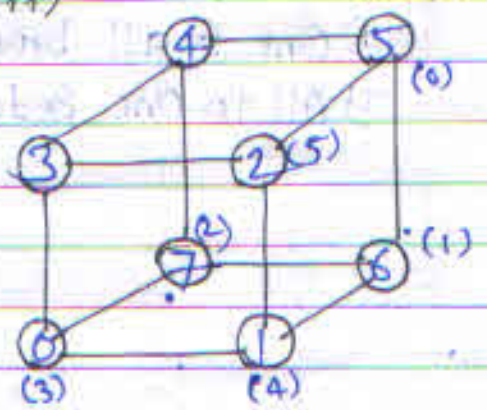


Second Comm step of the 4-shift

(a) First phase (a-4shift)



(b) The 2nd phase (a-1shift)



(c) Final data distribution after the 5-shift.

The mapping of 8-node linear array onto 3-D hypercube to perform a circular 5-shift as combination of a-4-shift & a 1-shift.

- Note that, number of term in sum = number of 1's in binary representation of q.

For. e.g. For number 5 (101) two term will be there in the sum corresponding to bit 2 and bit 0 i.e. $(2^2 + 2^0)$.

- Circular q-shift on a hypercube is performed in s-phase, where s is distinct power of 2.
- In each comm^n phase, move closer to the destination by power of 2.
- For e.g., 5 shift operation is performed by 4 shift (2^2) followed by 1 shift (2^0).
- Each step will have 2 comm^n steps. only 1-shift will have a single step.

for. e.g., the 1st phase of 4-shift consist of 2-steps and the 2nd phase of 1st shift consist of one step.

- Total number of step for any q in p-node hypercube is $2 \log p - 1$.

* Improving the speed of some Comm^n operations *

- algorithms are based on foll. assumption

1) original message can't be divided into smaller part.
2) Each nodes uses a single port for sending & receiving data.

explain -

1) One to all broadcast
2) All to One Reduction.