**Assignment #3 and Assignment #4**

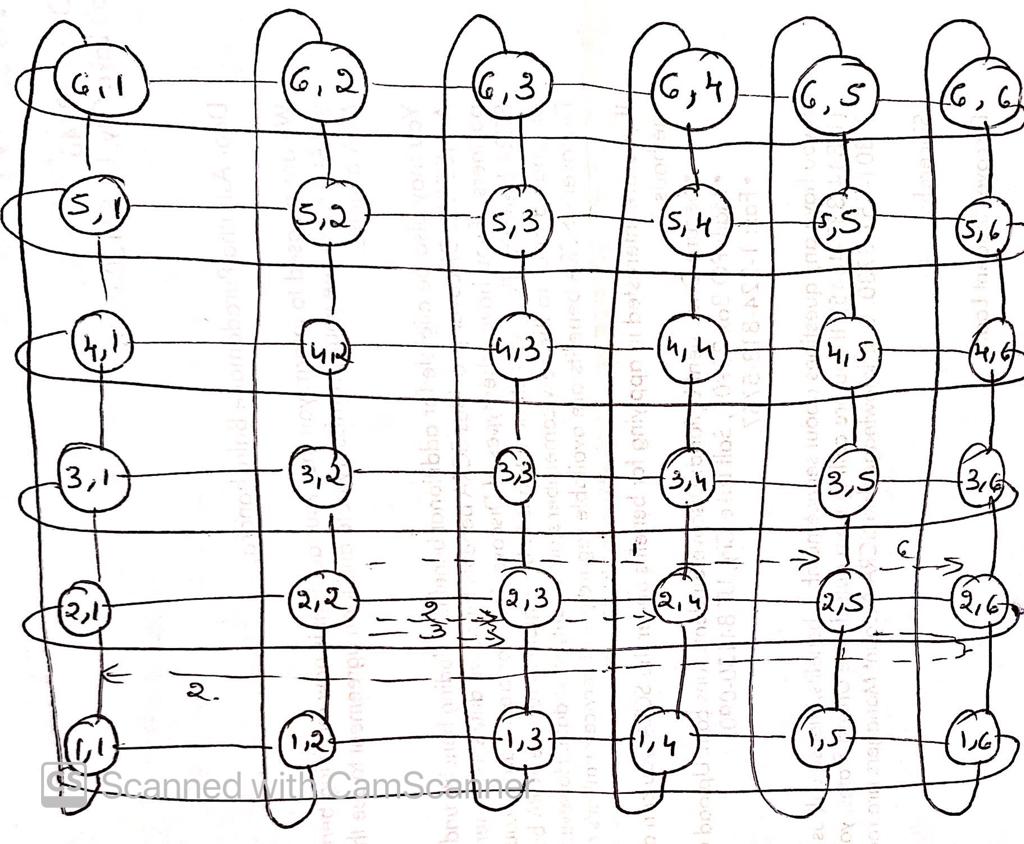
CS4379: Parallel and Concurrent Programming

CS5379: Parallel Processing

Spring 2020

Name: Ashritha Puradamane Balachandra

1. The following diagram shows 6\*6 processor array which are numbered as (i,j) (i, j = 1, 2, 3, 4, 5, 6) for a processor in row *i* and column *j*, arranged as a 2-dimensional torus.



For One-to-all broadcast data from processor (2, 2) to all the processors, there can be two solutions as shown in the above diagrams. In the first phase, the operation is performed along one or all rows by treating the rows as linear arrays. In the second phase, the columns are treated similarly.

Let m = 1000 bytes of data.

**Solution 1:**

First phase – row wise:

Step 1: from (2,2) to (2,5)

Step 2: from (2,2) to (2,4) and from (2,5) to (2,1)

Step 3: from (2,2) to (2,3) and from (2,5) to (2,6)

Second phase – column wise:

Step 1: from (2,2) to (5,2)

Step 2: from (2,2) to (4,2) and from (5,2) to (1,2)

Step 3: from (2,2) to (3,2) and from (5,2) to (6,2)

Communication cost = total time = 2 \*(( ts + twm + 3 th)+ ( ts + twm + 2 th) + ( ts + twm + th))

= 2 \*((10+0.01\*1000+3\*2) + (10+0.01\*1000+2\*2) + (10+0.01\*1000+2))

= 2\*(26+24+22)

= **144 microseconds**

**Solution 2:**

First phase – row wise:

Step 1: from (2,2) to (2,5)

Step 2: from (2,2) to (2,3) and from (2,5) to (2,6)

Step 3: from (2,3) to (2,4) and from (2,6) to (2,1)

Second phase – column wise:

Step 1: from (2,2) to (5,2)

Step 2: from (2,2) to (3,2) and from (5,2) to (6,2)

Step 3: from (3,2) to (4,2) and from (6,2) to (1,2)

Communication cost = total time = 2 \*(( ts + twm + 3 th)+ ( ts + twm + th) + ( ts + twm + th))

= 2 \*((10+0.01\*1000+3\*2) + (10+0.01\*1000+2) + (10+0.01\*1000+2))

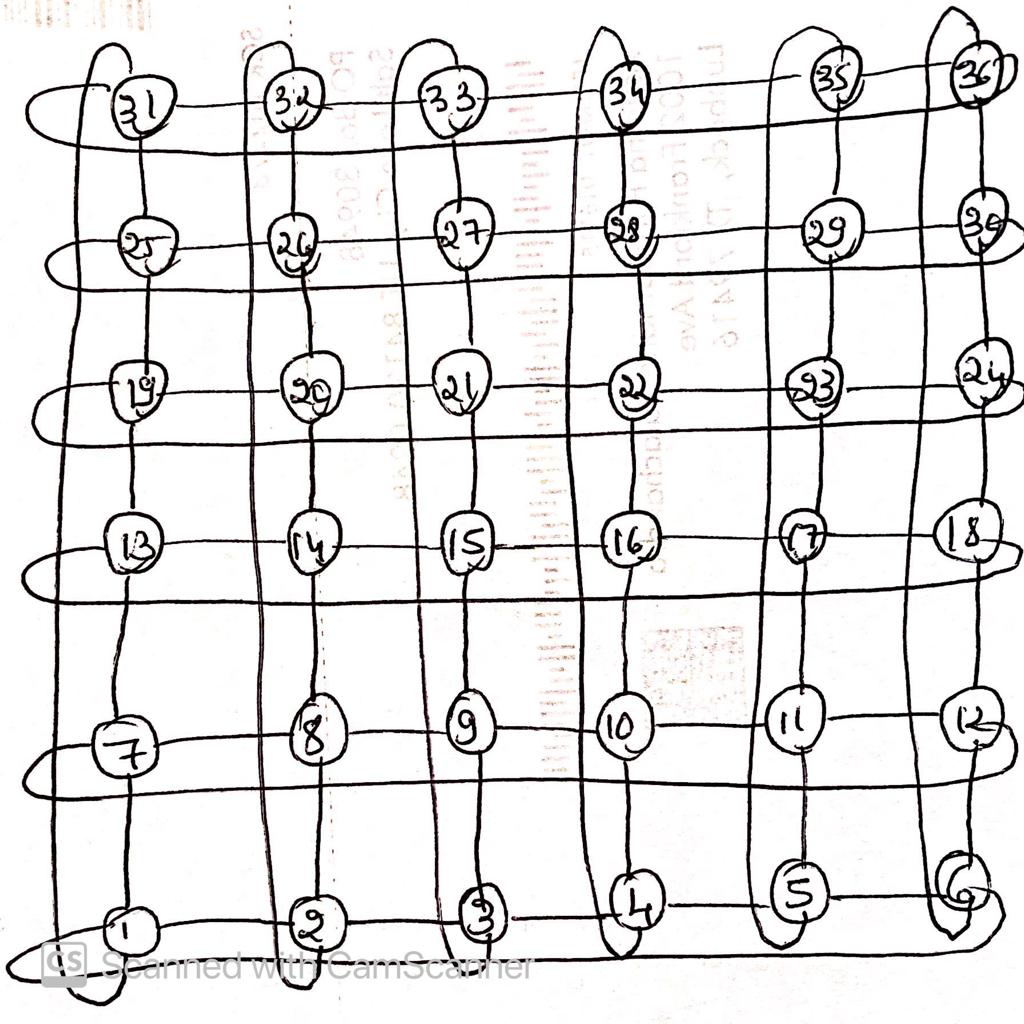
= 2\*(26+22+22)

= **140 microseconds**

1. Given that at beginning, each processor has 6\*6\*1,000 bytes of data. At the end, each processor also has 6\*6\*1,000 bytes of data, but messages are scattered among 6\*6 processors and totally exchanged.

Let m = 1000 bytes of data.

Number of processors = 6\*6 = 36.



In all-to-all personalized communication, each node first groups its *p* messages according to the columns of their destination nodes. Every node initially has 36*m* data. Each node assembles its data into 6 groups. The first group contains the messages destined for nodes labeled 1,7,13,19,25 and 31 (all nodes in 1st column). The second group contains the messages for nodes labeled 2,8,14,20,26,32 (all nodes in 2nd column). Similarly, all other groups contains its respective column nodes.

Once again, communication takes place in two phases.

Phase 1 – row wise

In the first phase, each row of the torus performs an all-to-all broadcast using the procedure for the linear array. In this phase, all nodes collect messages corresponding to the nodes of their respective rows. Each node consolidates this information into a single message of size 6m, and proceeds to the second communication phase.

Step 1 : from 1st group to other 5 groups - with message size 30m

Step 2: from 2st group to other 5 groups - with message size 24m

Step 3: from 3st group to other 5 groups - with message size 18m

Step 4: from 4st group to other 5 groups - with message size 12m

Step 5: from 5st group to other 5 groups - with message size 6m

Phase 2 – Column wise

The second communication phase is a column wise all-to-all broadcast of the consolidated messages. Before the second communication phase, the messages in each node are sorted again, this time according to the rows of their destination nodes; then communication similar to the first phase takes place in all the columns. By the end of this phase, each node receives a message from every other node. At the end of the second phase, node *i* has messages ({1, *i*}, …, {36, *i*}), where 1<=i<=36.

Step 1 : with message size 30m

Step 2: with message size 24m

Step 3: with message size 18m

Step 4: with message size 12m

Step 5: with message size 6m

Communication cost = total time = 10 ts + 2\*(30+24+18+12+6)\*m tw+10th

= 10\*10+2\*(90)\*1000\*0.1+20

= 100+1800+20

**= 1920 microseconds**

1. Considering the given All-to-all broadcast on given balanced binary tree, we have p = 8.

Step 1: the communication starts to exchange ‘m’ data size between (0,4), (1,5), (2,6) and (3,7) pairs of processors. For this communication, it has to visit the root node and hence it takes (ts + twm\*4) time.

Step 2: the communication starts to exchange ‘2m’ data size between (0,2), (1,3), (4,6) and (5,7) pairs of processors. If we observe the communication path, then 2 messages take the same path in same direction and hence this iteration takes (ts + tw2\*m\*(2)) = (ts + twm\*(4)) time.

Step 3: the communication starts to exchange ‘4m’ data size between (0,1), (2,3), (4,5) and (6,7) pairs of processors and it also takes (ts + twm\*(4)) time.

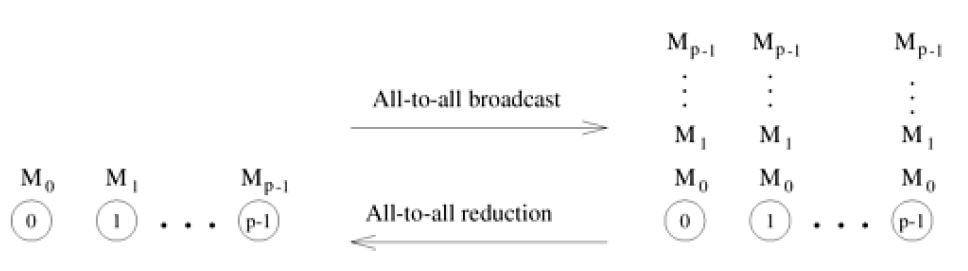
Therefore, for the given tree all-to-all broadcast takes (ts + twm\*(4)) 3 time.

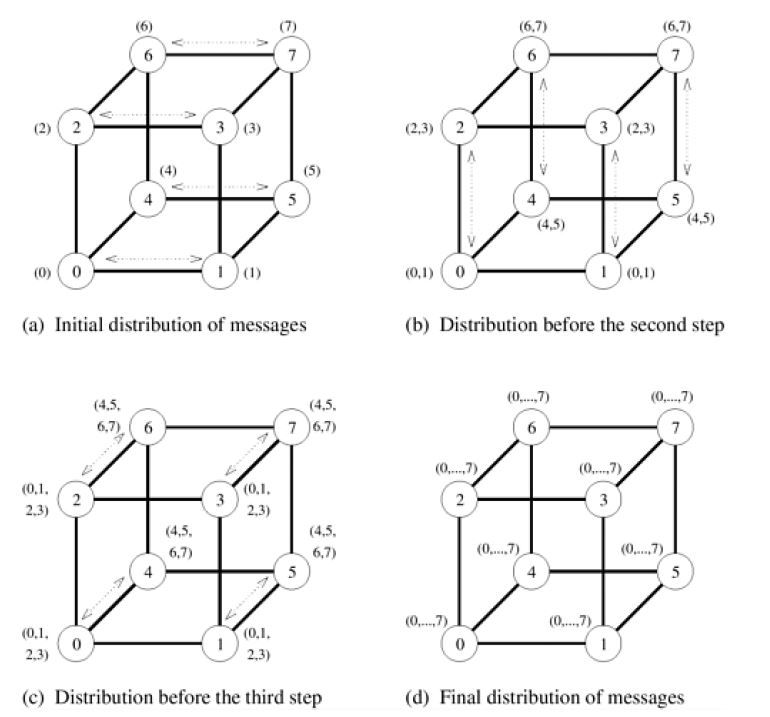
In general, we can say that for any balanced tree, all-to-all broadcast that takes

**Time = (ts + tw m p/2)log p** for m-word messages on p nodes.

The communication cost will increase if we try to change the order of the above steps. If we embed a logical ring onto the tree, it results in higher ts value but lower tw value and all-to-all broadcast on a p-processor tree takes (ts + tw m)(p-1) time. In general, it takes ts + tw mk if the communication channel is shared by k simultaneous messages.

1. In all-to-all reduction, the dual of all-to-all broadcast, each node starts with *p* messages, each one destined to be accumulated at a distinct node. All-to-all reduction can be performed by reversing the direction and sequence of the messages. In all-to-all reduction, each processor starts with p distinct messages but ends up with a single message.





All-to-all broadcast on an 8-node hypercube is shown in the above diagram.

If we take the above example, in 1st step, the processor 0 sends the messages destined for processors 4, 5, 6, and 7 to processor 7, and processor 7 sends the messages destined for processors 0, 1, 2, and 3 to processor 0. After this 1st step, each processor adds mp/2 numbers received to the m\*p/2 numbers already residing on them. The size of the data communicated and added in an iteration is half of that in the previous iteration.

Thus, we can say that in ith iteration (0 < i ≤ log p) of all-to-all reduction algorithm, processors communicates (log p − i + 1)th dimension of the hypercube and processor j communicates with processor j +2(logp-i) or j -2(logp-i).

Total time can be derived as /2i ) = **ts log p +m(tw + tadd)(p-1)**.