


National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	Parallel and Distributing Computing	Course Code:	CS3006
	Degree Program:	BS (CS)	Semester:	Spring 2023
	Exam Duration:	60 Minutes	Total Marks:	35
	Paper Date:	08/04/23	Weight	12.5
	Exam Type:	Mid II	Page(s):	5

Student : Name: SOLUTION Roll No. _____ Section: _____

Question # 1 a: [6 marks, CLO # 1]

From the given options, select the best answer.

- i. With distributed systems, we will have:
 - a. Processors with a shared logical address space
 - b. Processors with a disjoint logical address space
 - c. Processors that interact with each other by message passing
 - d. **Both (b) and (c)**
- ii. Failover clusters are also known as:
 - a. **High availability clusters**
 - b. Load balancing clusters
 - c. High performance clusters
 - d. None of the given option
- iii. For a 2*8 mesh, using a naïve solution, how many messages would you expect the sending process to send, for a 1-to-all broadcast?
 - a. Less than 8
 - b. **More than 12**
 - c. Less than 4
 - d. Any one of the given option
- iv. Google App Engine is an example of
 - a. Infrastructure as a service
 - b. **Platform as a service**
 - c. Software as a service
 - d. None of the above
- v. The transparency that enables access to local and remote resources using identical operations is
 - a. **Access transparency**
 - b. Location transparency
 - c. Computation transparency
 - d. Scaling transparency
- vi. With the naïve solution for one-to-all broadcast on a ring, we would expect to send _____ messages to the other _____ processes, and this may lead to an _____ of the communication network
 - a. p; p-1; overutilization
 - b. **p-1; p-1; underutilization**
 - c. p-1; p-1; overutilization

d. $(p-1)^2$; $p-1$; underutilization

Question # 1 b: [3 marks, CLO # 1]

True/False

- I. The purpose of community cloud is to combine different clouds, e.g. private and public cloud.
 - a. True
 - b. **False**
- II. The total exchange in a linear ring with p nodes would take p steps.
 - a. True
 - b. **False**
- III. Grid Computing is a form of Utility Computing.
 - a. **True**
 - b. False

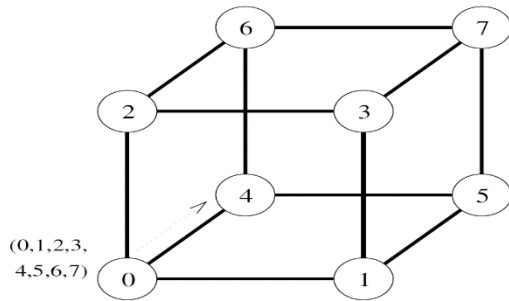
Question # 1 c: [6 marks, CLO # 1]

Provide short answers for the given questions.

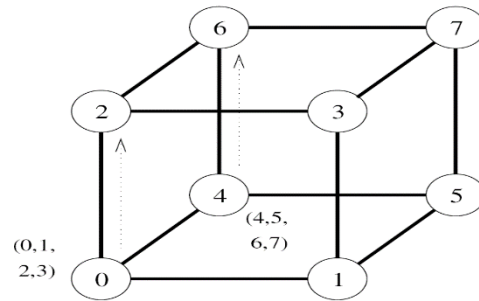
- I. What should be the total time for sending a message of size m across n hops?
$$t = t_s + (t_w m + t_h + t_r) n$$
- II. Which form of Operating System completely hides the distribution and presents a single system image?
Middleware
- III. Which form of Operating System is better suited for tightly coupled OS for multiprocessors and homogenous multi-computers?
Distributed OS
- IV. Which form of Operating System is better suited for loosely coupled OS for heterogeneous multi-computers?
Network OS
- V. eScience applications are better suited to run on which type of computing?
Grid Computing
- VI. How is the gather operation different from all-to-one reduction?
The size of the message doubles at each step in the gather operation whereas in the reduction operation the size of message remains the same.

Question # 2: [3+3+4 marks, CLO # 3]

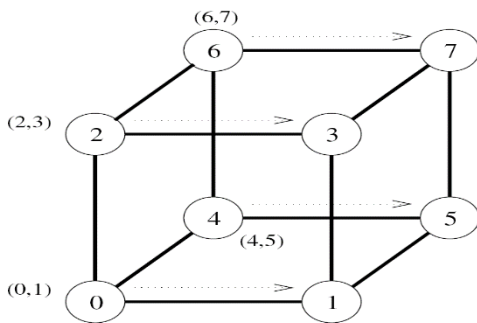
a) Explain the scatter communication operation on a hypercube with 8 nodes.



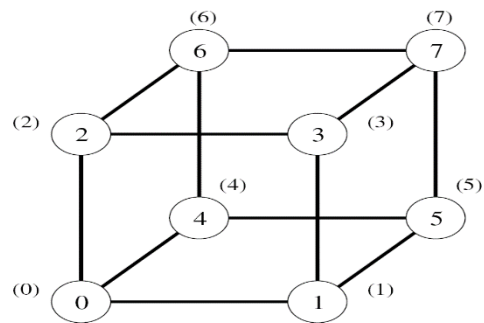
(a) Initial distribution of messages



(b) Distribution before the second step



(c) Distribution before the third step



(d) Final distribution of messages

Figure 4.15 The scatter operation on an eight-node hypercube.

b) Provide total cost estimation for this operation. You have to consider the size of each message!

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

At first step, the size of the message is 4 then 2 and finally 1 thus amounting to $4+2+1 = 7$ which is equivalent to $p - 1$.

- c) What modification would be required in the following code to perform scatter operation instead of one to all broadcast?

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

The only difference is the message to be sent and received so in this case X is an array of P messages or 2^d messages.

Now at sender end, we need to keep the first half of the array and send second half to the receiver.

Int arraysize = 2^d; //declare a new variable

If (I am a participant)

```
{
    arraysize = arraysize/2;
```

If (I am the sender)

Send &X[arraysize], arraysize

else

Receive X, arraysize

```
}
```

Question # 3: [6+4 marks, CLO # 2]

- a) Write the output for the following piece of code assuming that there are 4 MPI processes. Assume there is no syntax error.

```
#include <mpi.h>
#include <stdio.h>

int main (int argc, char** argv) {
    MPI_Init (NULL, NULL);
    MPI_Status status;

    int p, b, my_rank;
    MPI_Comm_size(MPI_COMM_WORLD, &p);
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
    int a = my_rank;
    int sTag = a;
    int rTag = (a-1+p) % p;
    int next = (my_rank + 1) % p;
    int prev = ((my_rank - 1 + p) % p);
    MPI_Sendrecv(&a,1,MPI_INT,next,sTag, &b,1,MPI_INT,prev,rTag, MPI_COMM_WORLD, &status);
    printf("I am %d: Got:%d from %d and Sent:%d to %d\n ", my_rank, b, prev, a, next);
    MPI_Finalize();
}
```

I am 0: Got: 3 from 3 and Sent: 0 to 1

I am 3: Got: 2 from 2 and Sent: 3 to 0

I am 2: Got: 1 from 1 and Sent: 2 to 3

I am 1: Got: 0 from 0 and Sent: 1 to 2

The order may differ (vertically)

- b) What is the one extra parameter used in MPI_Recv() that is not used in MPI_Send(), and in what circumstances would it be of greater importance?

The status parameter (of MPI_Status type) is the extra parameter. It may be of greater importance when we call the MPI_Recv() using wildcards for either source or tag, or possibly both. This parameter provides us with information on the (i) source, (ii) tag, and (iii) error from the incoming message from another process.