

Midterm: CS685-002, Fall 1999

October 21, 1998, 12:30–1:45PM

Note that all communication links are bidirectional. A processor can send and receive messages at the same time. But cannot send (or receive) more than one message at any given time.

1. (10 points) What are the major advantages and disadvantages of the crossbar switching networks and the bus-based networks? What are the major differences between the message passing architecture and the shared memory architecture?
2. (10 points) What does it mean to be a symmetric multiprocessor architecture? What are the uniform and nonuniform memory access computers? Do a symmetric multiprocessor architecture provide uniform or nonuniform memory access? why?
3. (10 points) Assuming cut-through-routing scheme is used, give the best procedure and communication time for one-to-all personalized communication of m -word messages on a p processor linear array (without wraparound) architecture. The source processor must be at one end of the linear array. (Detailed descriptions are required.)
4. (10 points) Figure 1 is a 4 dimensional hypercube architecture. (a) Label the processors using binary representation and calculate the Hamming distance between the processors 2 and 13. (b) Describe a noncongestion optimal procedure and compute the corresponding communication time for processors 0 and 15 to exchange m -word messages using store-and-forward routing with the E-cube routing scheme.

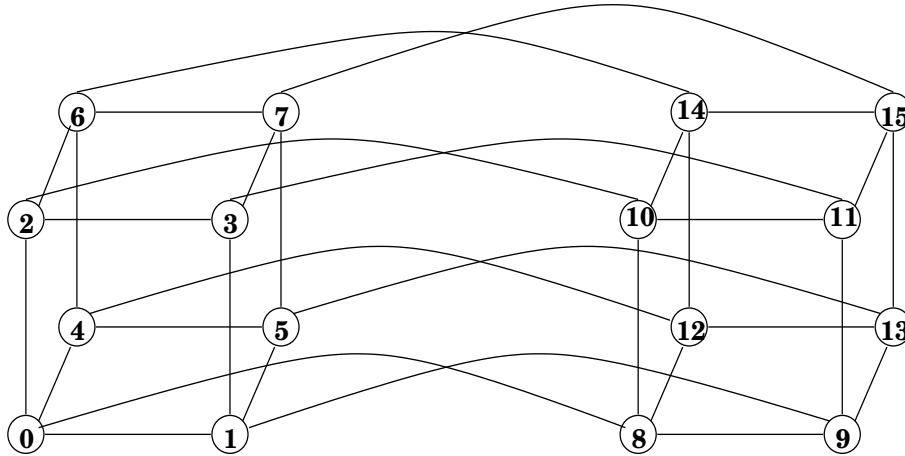


Figure 1: A 4 dimensional hypercube architecture for Problem 4.

5. (10 points) The following MPI code is incorrect. Find out and correct, as much as possible, the incorrect parts of this MPI code? Explain why you think that part is incorrect. You can do this by adding comments to the original code.

```

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

#define MAXSIZE 20

main(int argc, char *argv[])
{
    int myid, numprocs, localresult, result[MAXSIZE];
    int i, tag, final;

    MPI_Status status;

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);

    if ( numprocs > MAXSIZE )
        { if ( myid == 0 )
            printf("The number of processes must be less than %d\n",
                MAXSIZE);
            exit(0);
        }
}

```

```

    }

    final = 0;

    for(i = 0; i < numprocs; i++)
        result[i] = i;

    MPI_Scatter(result, numprocs, MPI_INT, &localresult, 1, MPI_INT,
               0, MPI_COMM_WORLD);

    if ( myid )
    { localresult = localresult/myid;

      MPI_Reduce(&localresult, &final, 1, MPI_INT, MPI_SUM, 1,
                MPI_COMM_WORLD);
    }

    if ( final )
    { tag = myid;
      MPI_Send(&final, 1, MPI_INT, 0, tag, MPI_COMM_WORLD);
    }

    if ( myid == 0 )
        MPI_Recv(&final, 1, MPI_INT, MPI_ANY_SOURCE, MPI_ANY_TAG,
                MPI_COMM_WORLD, &status);

    if( myid == 0 )
        printf("The final result is: %d \n", final);

    MPI_Finalize();
    return(0);
}

```