Computação em Larga Escala

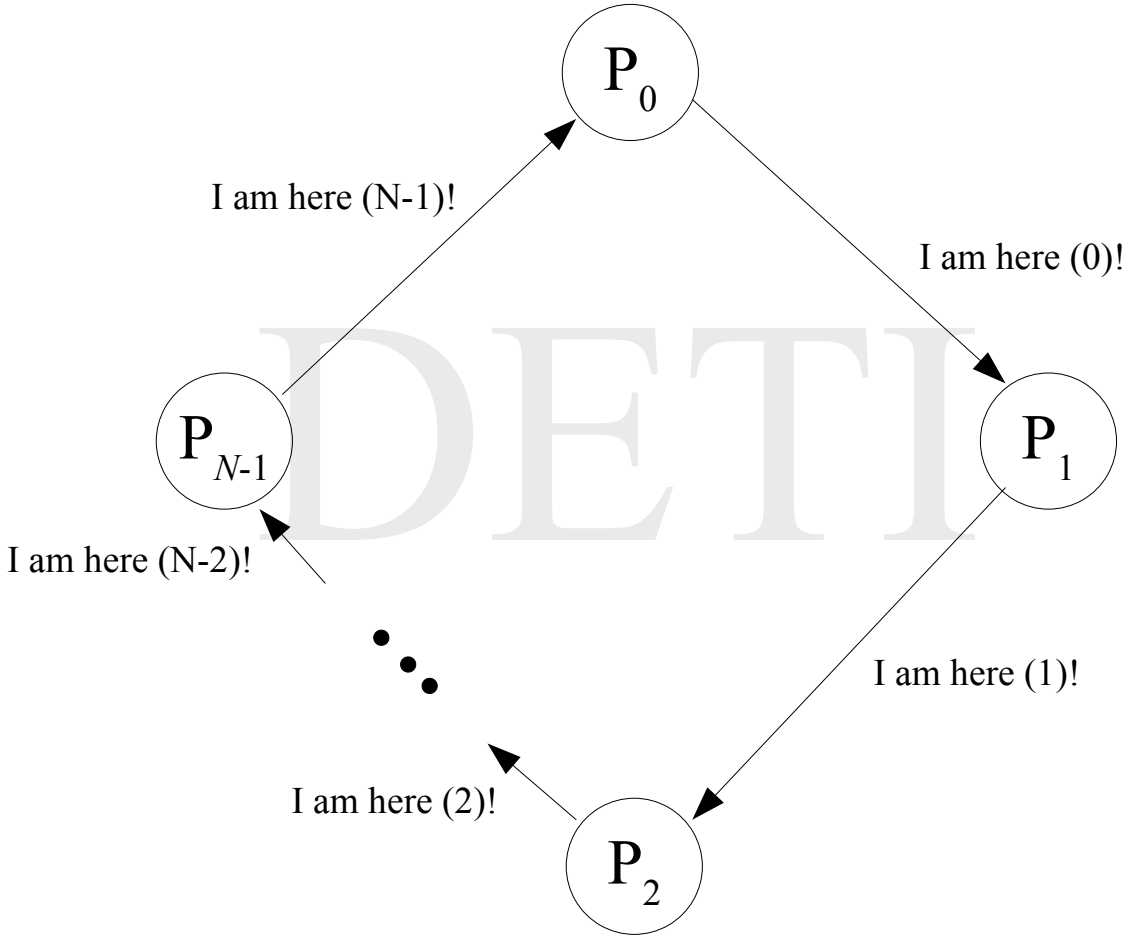
MPI Problems 1 – Algorithmic analysis

António Rui Borges

Summary

- *Ring communication*
- *One-to-all and all-to-one communication*

Ring communication - 1



Ring communication - 2

- process 0 sends a message to next process in the group, waits for a message from process (N-1) % N and terminates
- all other processes i, with $0 < i < N$, wait for a message from process i-1, send a message to process (i+1) % N and terminate

Ring communication - 3

```c
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main (int argc, char *argv[])
{
   int rank, size;
   char *sndData, *recData;
   MPI_Init (&argc, &argv);
   MPI_Comm_rank (MPI_COMM_WORLD, &rank);
   MPI_Comm_size (MPI_COMM_WORLD, &size);
   if (rank == 0)
      { sndData = malloc (100);
        sprintf (sndData, "I am here (%d)!", rank);
        printf ("I, %d, am going to transmit the message: %s\n", rank, sndData);
        MPI_Send (sndData, strlen (sndData) + 1, MPI_CHAR, (rank + 1) % size, 0, MPI_COMM_WORLD);
        recData = malloc (100);
        MPI_Recv (recData, 100, MPI_CHAR, size - 1, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        printf ("I, %d, received the message: %s\n", rank, recData);

      }
      else { recData = malloc (100);
             MPI_Recv (recData, 100, MPI_CHAR, rank - 1, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
             printf ("I, %d, received the message: %s\n", rank, recData);
             sndData = malloc (100);
             sprintf (sndData, "I am here (%d)!", rank);
             printf ("I, %d, am going to transmit the message: %s\n", rank, sndData);
             MPI_Send (sndData, strlen (sndData) + 1, MPI_CHAR, (rank + 1) % size, 0, MPI_COMM_WORLD);
           }
   MPI_Finalize ();
   return EXIT_SUCCESS;
}
```

Ring communication - 4

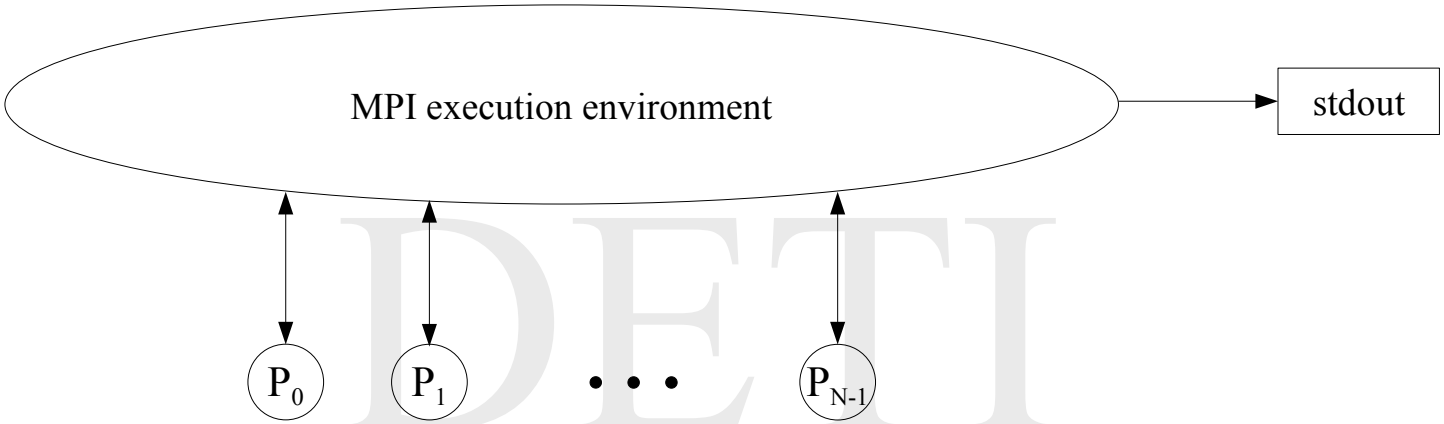
```
[ruib@ruib-laptop exercises-1]$ mpiexec -n 8 ./ringSendRecData
I, 0, am going to transmit the message: I am here (0)!
I, 1, received the message: I am here (0)!
I, 1, am going to transmit the message: I am here (1)!
I, 2, received the message: I am here (1)!
I, 2, am going to transmit the message: I am here (2)!
I, 3, received the message: I am here (2)!
I, 3, am going to transmit the message: I am here (3)!
I, 4, received the message: I am here (3)!
I, 4, am going to transmit the message: I am here (4)!
I, 5, received the message: I am here (4)!
I, 5, am going to transmit the message: I am here (5)!
I, 6, received the message: I am here (5)!
I, 6, am going to transmit the message: I am here (6)!
I, 7, received the message: I am here (6)!
I, 7, am going to transmit the message: I am here (7)!
I, 0, received the message: I am here (7)!

[ruib@ruib-laptop exercises-1]$ mpiexec -n 2 ./ringSendRecData
I, 0, am going to transmit the message: I am here (0)!
I, 0, received the message: I am here (1)!
I, 1, received the message: I am here (0)!
I, 1, am going to transmit the message: I am here (1)!
```


There is an apparent contradiction in the second run.

It looks like that causality is violated: process 0 receives the message from process 1 before it has been sent!

How to explain what is happening?

Ring communication - 5



Access to the standard output is controlled by the MPI execution environment.
When concurrent outputs take place, their ordering may not be chronological.
Only outputs from the same process are printed in chronological order.

Ring communication - 6

```
[ruib@ruib-laptop exercises-1]$ mpiexec -n 1 ./ringSendRecData
I, 0, am going to transmit the message: I am here (0)!
```

Deadlock!

The program is not completely general.

If executed with a single process, it blocks. Process 0 can not be at the same time sending and receiving a message!

A possible solution to the problem is to turn the *send* operation from process 0 from blocking to non-blocking.

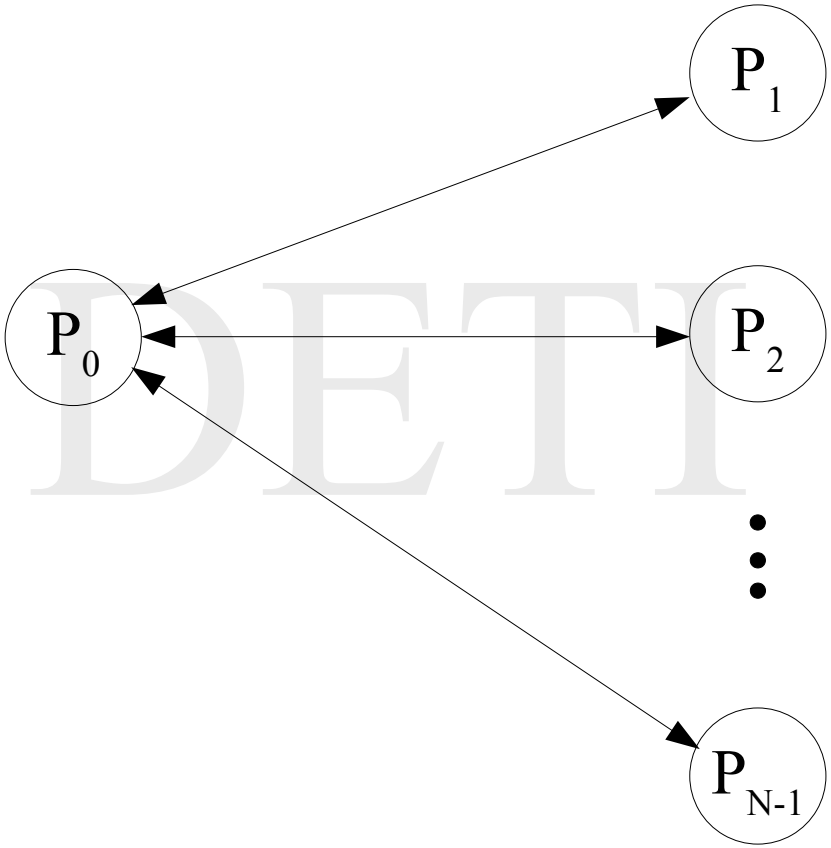
Ring communication - 7

```c
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main (int argc, char *argv[])
{
   int rank, size;
   char *sndData,
        *recData;
   MPI_Init (&argc, &argv);
   MPI_Comm_rank (MPI_COMM_WORLD, &rank);
   MPI_Comm_size (MPI_COMM_WORLD, &size);
   if (rank == 0)
      { MPI_Request req;
        MPI_Status stat;
        sndData = malloc (100);
        sprintf (sndData, "I am here (%d)!", rank);
        printf ("I, %d, am going to transmit the message: %s\n", rank, sndData);
        MPI_Isend (sndData, strlen (sndData) + 1, MPI_CHAR, (rank + 1) % size, 0, MPI_COMM_WORLD,
                   &req);
        recData = malloc (100);
        MPI_Recv (recData, 100, MPI_CHAR, size - 1, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        printf ("I, %d, received the message: %s\n", rank, recData);
        MPI_Wait (&req, &stat);
      }
      else { recData = malloc (100);
             MPI_Recv (recData, 100, MPI_CHAR, rank - 1, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
             printf ("I, %d, received the message: %s\n", rank, recData);
             sndData = malloc (100);
             sprintf (sndData, "I am here (%d)!", rank);
             printf ("I, %d, am going to transmit the message: %s\n", rank, sndData);
             MPI_Send (sndData, strlen (sndData) + 1, MPI_CHAR, (rank + 1) % size, 0, MPI_COMM_WORLD);
           }
   MPI_Finalize ();
   return EXIT_SUCCESS;
}
```

Ring communication - 8

```
[ruib@ruib-laptop exercises-1]$ mpiexec -n 1 ./ringSendNBRecData
I, 0, am going to transmit the message: I am here (0)!
I, 0, received the message: I am here (0)!

[ruib@ruib-laptop exercises-1]$ mpiexec -n 2 ./ringSendNBRecData
I, 0, am going to transmit the message: I am here (0)!
I, 0, received the message: I am here (1)!
I, 1, received the message: I am here (0)!
I, 1, am going to transmit the message: I am here (1)!

[ruib@ruib-laptop exercises-1]$ mpiexec -n 8 ./ringSendNBRecData
I, 0, am going to transmit the message: I am here (0)!
I, 1, received the message: I am here (0)!
I, 1, am going to transmit the message: I am here (1)!
I, 2, received the message: I am here (1)!
I, 2, am going to transmit the message: I am here (2)!
I, 3, received the message: I am here (2)!
I, 3, am going to transmit the message: I am here (3)!
I, 4, received the message: I am here (3)!
I, 4, am going to transmit the message: I am here (4)!
I, 5, received the message: I am here (4)!
I, 5, am going to transmit the message: I am here (5)!
I, 6, received the message: I am here (5)!
I, 6, am going to transmit the message: I am here (6)!
I, 7, received the message: I am here (6)!
I, 7, am going to transmit the message: I am here (7)!
I, 0, received the message: I am here (7)!
```

One-to-all and all-to-one - 1



One-to-all and all-to-one - 2

- process 0 sends a message to all other processes in the group, waits for a message from each of them and terminates
- all other processes i, with $0 < i < N$, wait for a message from process 0, send a message back to it and terminate

One-to-all and all-to-one - 3

```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main (int argc, char *argv[])
{
   int rank, size;
   char *sndData,
        *recData;
   MPI_Init (&argc, &argv);
   MPI_Comm_rank (MPI_COMM_WORLD, &rank);
   MPI_Comm_size (MPI_COMM_WORLD, &size);
   if (rank == 0)
      { int i;
        MPI_Request req;
        MPI_Status stat;

        sndData = malloc (100);
        recData = malloc (100);
        sprintf (sndData, "I am alive (%d)!", rank);
        printf ("I, %d, am going to transmit the message: %s to all other processes in the group\n",
                rank, sndData);
        for (i = (rank + 1) % size; i < size; i++)
        { MPI_Isend (sndData, strlen (sndData) + 1, MPI_CHAR, i, 0, MPI_COMM_WORLD, &req);
          MPI_Recv (recData, 100, MPI_CHAR, i, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
          printf ("I, %d, received the message: %s\n", rank, recData);
          MPI_Wait (&req, &stat);
        }
      }
      else { recData = malloc (100);
             MPI_Recv (recData, 100, MPI_CHAR, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
             sndData = malloc (100);
             sprintf (sndData, "I am alive (%d)!", rank);
             MPI_Send (sndData, strlen (sndData) + 1, MPI_CHAR, 0, 0, MPI_COMM_WORLD);
           }
   MPI_Finalize ();
   return EXIT_SUCCESS;
}
```

One-to-all and all-to-one - 4

```
[ruib@ruib-laptop exercises-1]$ mpiexec -n 1 ./oneToAllSendNBRecData
I, 0, am going to transmit the message: I am alive (0)! to all other processes in the group
I, 0, received the message: I am alive (0)!

[ruib@ruib-laptop exercises-1]$ mpiexec -n 2 ./oneToAllSendNBRecData
I, 0, am going to transmit the message: I am alive (0)! to all other processes in the group
I, 0, received the message: I am alive (1)!

[ruib@ruib-laptop exercises-1]$ mpiexec -n 4 ./oneToAllSendNBRecData
I, 0, am going to transmit the message: I am alive (0)! to all other processes in the group
I, 0, received the message: I am alive (1)!
I, 0, received the message: I am alive (2)!
I, 0, received the message: I am alive (3)!

[ruib@ruib-laptop exercises-1]$ mpiexec -n 8 ./oneToAllSendNBRecData
I, 0, am going to transmit the message: I am alive (0)! to all other processes in the group
I, 0, received the message: I am alive (1)!
I, 0, received the message: I am alive (2)!
I, 0, received the message: I am alive (3)!
I, 0, received the message: I am alive (4)!
I, 0, received the message: I am alive (5)!
I, 0, received the message: I am alive (6)!
I, 0, received the message: I am alive (7)!
```