|  |  |
| --- | --- |
|  | |
| ASIGNATURA: | Computación Distribuida |
| NOMBRE: | MOROCHO DARWIN  PEREZ ESTEBAN |
| FECHA DE ENTREGA: | 09-12-2015 |
|  | |

**Elección de líder**

|  |
| --- |
|  |

1. Algoritmo de elección de líder

1. Cuando el proceso p se da cuenta que todos los miembros del grupo coordinador se haya loqueado inicia el algoritmo de elección.

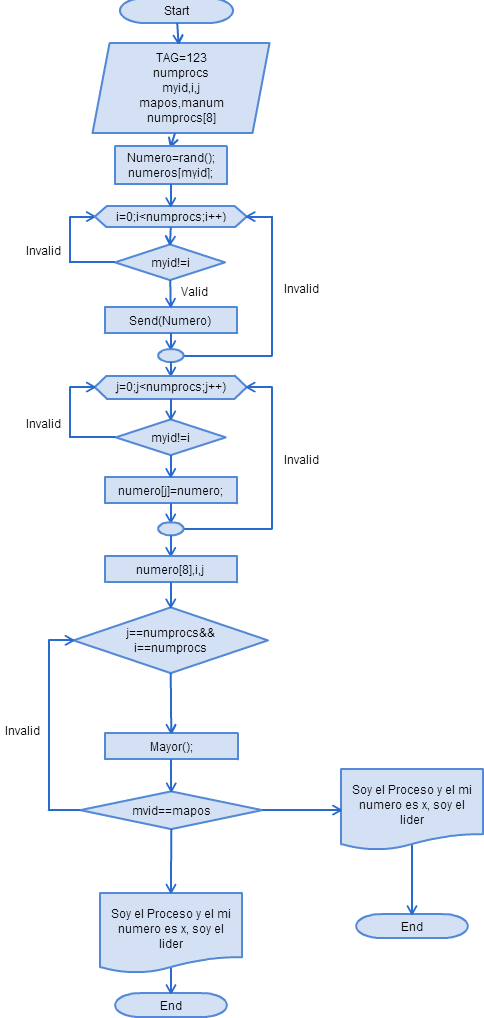
2. Cuando el proceso p encuentra que el coordinado fue “estrellado” envía un mensaje de elección a los otros procesos con mayor número de prioridad.

3. Cada proceso que recibe los mensajes de elección envía un mensaje OK con su número de prioridad.

4. (a): Si no hay ninguna respuesta al proceso P, entonces este es seleccionado como coordinador, y envía un mensaje a los procesos con menor número de prioridad para la selección de otros miembros del grupo de coordinación y espera 2d tiempo. Después de recibir todos los ids selecciona k-1 el proceso más alto cuando los miembros de la coordinación se agrupan y transmitirá un mensaje de coordinador a todos los procesos declarando al coordinado socio del grupo.

(b): Si algún proceso responde al proceso p: el proceso p seleccionará el coordinador del grupo con mayor número de prioridad y k procesos superiores como alternativa. Y a continuación se envía el nuevo coordinador mensaje de concesión.

5. El coordinador transmitirá un mensaje a los otros procesos que contienen su número de prioridad alterntava1, alterntiva2….k números alternativos.



**8**

**8**

**8**

**8**

**8**

**1**

**8**

**8**

**8**

**Calculo de la complejidad algorítmica**

**O(n)=8+8+8+8+8+8+1+8+8**

**O(n)=65**

1. Implementación del algoritmo de elección de líder. Creamos un archivo con el nombre **líder.c** con el siguiente código.

#include <stdio.h>

#include <mpi.h>

#include <string.h>

#include <stdlib.h>

#include <time.h>

void mayor(int cadena[],int n,int \*pos,int \*num);

int main(int argc, char\* argv[])

{

int TAG=123;

int numprocs, myid, i, j, numero, mapos, manum;

numprocs=8;

int numeros[numprocs];

MPI\_Status stat;

MPI\_Init(&argc, &argv);

MPI\_Comm\_size(MPI\_COMM\_WORLD,&numprocs);

MPI\_Comm\_rank(MPI\_COMM\_WORLD,&myid);

//numero randomico

srand(time(0)+myid);

numero=rand()% 37;

numeros[myid]=numero;

for(i=0;i<numprocs;i++)

{

if(myid!=i)

{

MPI\_Send(&numero, 1, MPI\_INT, i, TAG, MPI\_COMM\_WORLD);

//printf("Soy P%d envio el mensaje a P%d\n",myid,i);

}

}

for(j=0;j<numprocs;j++)

{

if(myid!=j)

{

MPI\_Recv(&numero, 1, MPI\_INT, j, TAG, MPI\_COMM\_WORLD, &stat);

//printf("Soy P%d y P%d Calculo %d\n",myid,j,numero);

numeros[j]=numero;

}

}

if((j==numprocs)&&(i==numprocs))

{

mayor(numeros,numprocs,&mapos,&manum);

if(myid==mapos)

{

printf("Soy el P %d y mi numero es: %d, Yo soy el lider\n",myid,numeros[myid]);

}

else

{

printf("Soy el P %d y mi numero es: %d, Yo no soy el lider\n",myid,numeros[myid]);

}

}

MPI\_Finalize();

return 0;

}

void mayor(int cadena[],int n,int \*pos,int \*num){

int i,j;

\*pos=0;

\*num=cadena[0];

//comparar un vector de numeros

for (i=0;i<n-1;i++)//numero1 a comparar

{

for (j=i+1;j<n;j++)//numero2 a comparar

{

if (\*num<cadena[j]){

\*pos=j;

\*num=cadena[j];

i=j-1;

j=n;

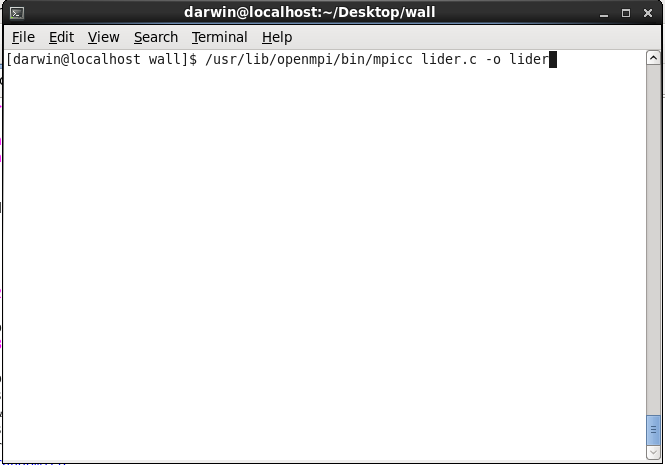
}

}

}

}

1. Compilamos el archivo con la siguiente sentencia



1. Ejecutamos el compilado con la siguiente sentencia.

