

Universidad Nacional Autónoma de México
Facultad de Ciencias
Computación Distribuida
Reporte de Práctica 3

1. Equipo

Integrantes del equipo:

- Bonilla Reyes Dafne - 319089660
- García Ponce José Camilo - 319210536
- Ortega Venegas Rodrigo Aldair - 318036104
- Rivera Mora Jesús Alberto - 313208641

2. Desarrollo

2.1 Descripción del desarrollo de la práctica y su funcionamiento:

a) Configuración:

- 1) Importamos la biblioteca MPI (`#include <mpi.h>`) para poder realizar la programación distribuida.
- 2) Se definen las etiquetas (tags) para las operaciones de comunicación entre nodos como: `TAG_INAC`, `TAG_ELEC`, `TAG_RESP`, `TAG_VICT`, `TAG_ACTI` y `TAG_TERM`.

b) Inicialización:

- 1) El programa espera dos argumentos de línea de comandos: el número total de nodos y el número de nodos inactivos.
- 2) Se inicializa el entorno MPI con `MPI_Init`.
- 3) Obtenemos el número de nodos y el rango del nodo actual.

c) Generación de números aleatorios y envío de estados:

- 1) Se utiliza `srand` para inicializar un generador de números aleatorios con una semilla única basada en la hora actual y el nodo actual.
- 2) Se envía a todos los nodos su respectivo estado.
- 3) Se verifica en el arreglo `nodosActivos[size]` el primer nodo que tenga estado activo 1, se almacena su índice en la variable `pos` y se envía el resultado a todos los nodos.

d) Recibir estado y nodo activo:

- 1) Cada nodo recibe su estado con la etiqueta `TAG_INAC` y almacena ese valor en la variable `activo`
- 2) Todos los nodos reciben el nodo activo que iniciara la ejecución del algoritmo. Este nodo se almacena en la variable `eleccion`. Los nodos que reciben el mensaje tienen la etiqueta `TAG_ACTI`
- 3) Se crea un arreglo `terminados[size]` para cada posición del arreglo se inicializa con 0.

e) Inicio de paso de mensajes con el nodo electo:

- 1) El nodo `eleccion` envía mensajes a los nodos de rango mayor, espera a recibir respuesta de cada nodo.
- 2) Si no recibe respuesta de alguno de los otros nodos, entonces el nodo `eleccion` envía un mensaje a todos los nodos con la etiqueta `TAG_VICT`
- 3) Verificamos que el nodo actual este activo, si es el caso, entonces el nodo actual se proclama como lider.
- 4) El nodo actual envía un mensaje a todos los nodos de que ya termino con la etiqueta `TAG_TERM` y en el arreglo `terminados[rank] = 1`, asignamos el valor 1 para marcar como terminado ese nodo.
- 5) Entramos en un while, los nodos que aún no han terminado reciben un mensaje de terminar, asignamos el valor 1 de que ya termino en el arreglo `terminados[emisor]`, si todos los nodos ya terminaron, entonces salimos del ciclo, en otro caso seguimos esperando mensajes.

*f) Nodo **eleccion** no es el nodo con el rango más alto:*

- 1) Entramos en un ciclo, donde se reciben los mensajes y se guardan en la variable `emisor`.
- 2) Recibimos mensajes de cualquier etiqueta, si la etiqueta es `TAG_VICT`, entonces se actualiza el rango lider, el nodo envía mensajes a los demás nodos que ya termino, esperamos a que todos los nodos terminen, se dejan de recibir mensajes y se sale del ciclo.
- 3) Si recibimos un mensaje con la etiqueta `TAG_TERM`, entonces asignamos a `terminados[emisor] = 1`, si todos los nodos ya terminaron, entonces el nodo actual termina, salimos del ciclo, en cualquier otro caso el nodo sigue esperando mensajes.

g) Nodo actual es diferente a nodo eleccion:

- 1) Inicializamos un variable `esperando = 1` y `eleccion = 0`.
- 2) Entramos es un ciclo como en pasos anteriores hasta que `esperando != 0`.
- 3) Si el nodo actual esta activo, se reciben menssaje de cualquier etiqueta, si la etiqueta es `TAG_ELEC`, entonces verificamos que le nodo actual este activo, enviamos un mensaje todos los nodos con la etiqutea `TAG_RESP`.
- 4) Iniciamos una elección, donde enviamos un mensaje con el tag `TAG_ELEC` a los nodos con rango mayor que el nodo actual.
- 5) Esperamos a que todos los nodos respondan, y verifcamos si el nodo actual es el nodo activo de mayor rango.
- 6) Si el nodo actual es el nodo de mayor rango, se realizan los mismo pasos del punto e) desde el inciso 2)
- 7) Si el nodo actual no está activo, se inicializa una variable `resp = -1` se envian mensajes con el valor de `resp` al emisor.
- 8) Si recibimos un mensaje de victoria con la etiqueta `TAG_VICT`, se relizan los pasos del punto f) inciso 2).
- 9) Si recibimos un mensaje de terminado se procede a realizar los pasos de punto f) inciso 3).

h) Resultados:

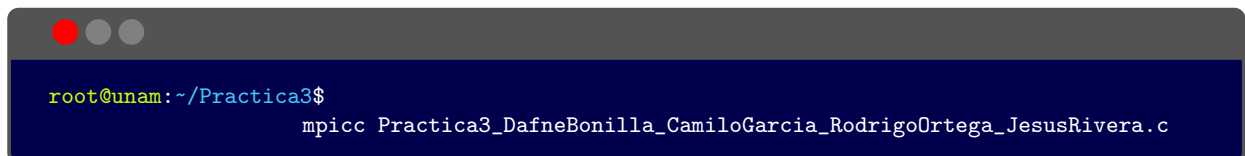
- 1) Cada nodo va imprimir en consola si esta activo o no y también el rank del nodo lider.

i) Finalización:

- 1) El programa MPI finaliza utilizando `MPI_Finalize()`.

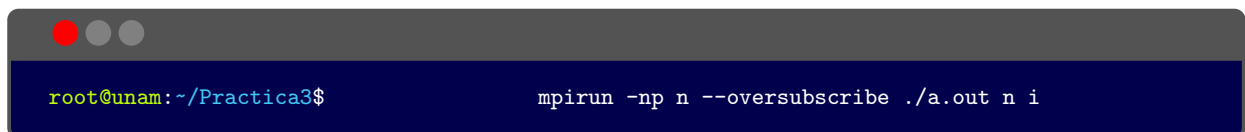
2.2 Compilación y salidas:

- Compilar desde `/Practica3`:



```
root@unam:~/Practica3$
mpicc Practica3_DafneBonilla_CamiloGarcia_RodrigoOrtega_JesusRivera.c
```

- Para ejecutar desde `/Practica3`:



```
root@unam:~/Practica3$ mpirun -np n --oversubscribe ./a.out n i
```

Donde `n` es el número de nodos e `i` el número de nodos inactivos.

2.3 Ejemplos de salidas:

- Ejemplo con 20 nodos y 8 inactivos:

```
root@unam:~/Practica3$ mpirun -np 20 --oversubscribe ./a.out 20 8

Resultado:
Nodo 0 activo, con lider al nodo 17
Nodo 1 activo, con lider al nodo 17
Nodo 2 activo, con lider al nodo 17
Nodo 3 activo, con lider al nodo 17
Nodo 4 activo, con lider al nodo 17
Nodo 5 inactivo, con lider al nodo 20
Nodo 6 activo, con lider al nodo 17
Nodo 7 activo, con lider al nodo 17
Nodo 8 inactivo, con lider al nodo 20
Nodo 9 activo, con lider al nodo 17
Nodo 10 inactivo, con lider al nodo 20
Nodo 11 inactivo, con lider al nodo 20
Nodo 12 inactivo, con lider al nodo 20
Nodo 13 activo, con lider al nodo 17
Nodo 14 activo, con lider al nodo 17
Nodo 15 inactivo, con lider al nodo 20
Nodo 16 activo, con lider al nodo 17
Nodo 17 activo, con lider al nodo 17
Nodo 18 inactivo, con lider al nodo 20
Nodo 19 inactivo, con lider al nodo 20
```

- Ejemplo con 10 nodos y 5 inactivos:

```
root@unam:~/Practica3$ mpirun -np 10 --oversubscribe ./a.out 10 5

Resultado:
Nodo 0 inactivo, con lider al nodo 10
Nodo 1 inactivo, con lider al nodo 10
Nodo 2 activo, con lider al nodo 9
Nodo 3 activo, con lider al nodo 9
Nodo 4 inactivo, con lider al nodo 10
Nodo 5 inactivo, con lider al nodo 10
Nodo 6 activo, con lider al nodo 9
Nodo 7 activo, con lider al nodo 9
Nodo 8 inactivo, con lider al nodo 10
Nodo 9 activo, con lider al nodo 9
```

2.4 Comentarios:

A veces no termina la ejecución, en ese caso simplemente cancelarla y volverlo a correr. Por lo general al volverlo a correr ya funciona, pero en teoría siempre termina.