Universidad Nacional Autónoma de México

Facultad de Ciencias

Computación Distribuida

Práctica 01

Introducción a la Computación Distribuida

Profesores:

Fernando Michel Tavera
Luis Mario Escobar Rosales
Brenda Ayala Flores
David Ortega Medina

Fecha de entrega

Martes, 9 de septiembre del 2025.

Lineamientos

Los alumnxs deberán hacer y entregar las prácticas siguiendo todos los puntos mencionados,si alguno llegase a ser incumplido la penalización puede variar desde puntos menos a su calificación hasta la nulificación total de éstas.

1. Las prácticas deben ser realizadas en parejas , no se calificarán prácticas individuales, así como prácticas de equipos con más de 2 miembros.
2. Las prácticas se deberán entregar a través de classroom, en un archivo .zip.Solo un miembro del equipo debe entregar la tarea con los archivos, pero ambos deben marcar como entregado.
3. El nombre del archivo .zip, debe empezar con el número de la práctica, seguido de los nombres de los integrantes.

   Practica1_FernandoMichel_DavidOrtega.zip

4. Se deberán usar únicamente las bibliotecas permitidas para dicha práctica.Queda prohibido el uso de cualquier biblioteca externa en el código.
5. Se deberá realizar un reporte en un archivo pdf que debe llevar lo siguiente :
   - Una descripción general de como se desarrolló la práctica.
   - Se debe hacer un análisis detallado en como se implementaron los algoritmos solicitados.Mencionando todas las consideraciones, etc.
   - Cualquier otro comentario o aclaración que consideren pertinente.

   Este pdf debe ir al mismo nivel que la carpeta principal de sus programas.
6. Se debe agregar un README.txt que contenga :
   - Número de la práctica
   - Nombre y número de cuenta de los integrantes

   Debe ir al mismo nivel(en la jerarquía de carpetas que el reporte).
7. Si ni el README ni el reporte llevan los nombres de los integrantes , habrá una penalización de la calificación con un punto menos.
8. Queda estrictamente prohibido el uso de cualquier código generado por Inteligencia Artificial, así como código copiado de Internet y copias entre equipos.De haber sospecha por alguna de estas situaciones, los integrantes del equipo deberán tener una entrevista con el ayudante de laboratorio.En esta entrevistas se les cuestionara aspectos de su implementaciones, algoritmos y código en general. En caso de incumplirse esta norma, la calificación de dicha práctica será automáticamente 0.
9. Si se tiene alguna complicación con la fecha y horario de entrega de la práctica , se debe avisar al ayudante para buscar una solución.
10. Si se entrega código que no compile y/o muera nada más ejecutarlo, la calificación será 0.

Computación distribuida - Algoritmos básicos

Nota : En su implementación, no deberán importar bibliotecas o usar otros archivos a menos que la práctica o el profesor lo indique.Para esta práctica bastará con usar Simpy , time y pytest.

En esta práctica implementaremos algoritmos y estructuras básicas en la computación distribuida.

1. Implementar las interfaces de Nodo y Canal en las siguientes clases

   - CanalBroadcast
   - NodoVecinos (ALgoritmo 1)
   - NodoTopologia (Algoritmo 2)
   - NodoBroadCast (Algoritmo 3)

   Además completa la función de cada clase según el tipo de Nodo siguiendo sus respectivos algoritmos.

2. Clases y variables

   - Prueba para conocer a los vecinos de los vecinos de un nodo identifiers (n.identifiers): Dicha variable debe corresponder con una lista donde cada nodo almacena a los otros nodos.La prueba arroja un error mostrando el nodo que tiene mal dicho valor de la variable en cuestión.
   - Prueba para el algoritmo Broadcast mensaje (nodo.mensaje): variable que hace referencia al mensaje que fue enviado por el nodo p_s

3. Uso Para el uso de las pruebas sigue los siguientes pasos:

   - Localiza en la misma carpeta que tus códigos fuentes el archivo test.py
   - Ejecuta el siguiente comando:

     myUser:$ pytest -q test.py

Algoritmos

Algorithm 1 Código vecinos para proceso p_i

1: $id_i = i$
2: $neighbors_i = \{\text{Conjunto de vecinos de } p_i\}$
3: $identifiers_i = \{\emptyset\}$

4: **main**()
5: **for all** $j \in neighbors_i$ **do**
6: **send** MYNAME($neighbors_i$) to j
7: **end for**

8: **when** MYNAME($identifiers_j$) is received from neighbor p_j:
9: $identifiers_i = identifiers_i \cup identifiers_j$

Algorithm 2 Código vecinos topología para p_i

1: $vecinos_i, proc\_conocidos_i = \{i\}$
2: $canales\_conocidos_i = \{< i, j > |\ j \in vecinos_i\}$

3: **start**()

4: **for all** $j \in vecinos_i$ **do**
5: **send** POSITION($i, vecinos_i$) to j
6: **end for**

7: **when** POSITION($k, vecinos$) is received from neighbor p_j
8: **if** $k \notin proc\_conocidos_i$ **then**
9: $proc\_conocidos_i = proc\_conocidos_i \cup \{k\}$
10: $canales\_conocidos_i = canales\_conocidos_i \cup \{< k, l > |\ l \in vecinos\}$
11: **for all** $l \in vecinos_i \setminus \{j\}$ **do**
12: **send** POSITION($k, vecinos$) to l
13: **end for**
14: **if** $\forall < l, m > \in canales\_conocidos_i : \{l, m\} \subset proc\_conocidos_i$ **then**
15: p_i conoce la gráfica de comunicación; **return**
16: **end if**
17: **end if**

Algorithm 3 Broadcast ingenuo para el proceso p_i

1: $neighbors_i = \{$conjunto de vecinos$\}$
2: $seen\_message =$ **false**
3: **if** $p_i = p_s$ **then**
4: $\quad seen\_message =$ **true**
5: $\quad$ **send** M to all neighbors
6: **end if**

7: **when** M is received from neighbor p_j:
8: **if** $seen\_message =$ **false then**
9: $\quad seen\_message =$ **true**
10: $\quad$ **send** M to all neighbors
11: **end if**