Universidad Nacional Autónoma de México

Facultad de Ciencias

Computación Distribuida

Práctica 07

Consenso

Profesores:

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

Fecha de entrega

Viernes , 21 de noviembre 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 - Consenso

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 de búsqueda,ordenamiento, etc..

1. Introducción
   Básicamente el consenso tiene como objetivo proveer a los procesos con un mecanismo de consistencia global, es decir, que para todo proceso que proponga una valor a sus demás procesos, entonces el consenso asegura que todos los procesos que logran terminar su ejecución se pongan de acuerdo en el valor propuesto por sus demás procesos. Para el caso de un sistema síncrono, este problema se puede resolver, sin embargo para el caso de un sistema asíncrono con tendencia a fallos existe un resultado el cual nos dice que es imposible diseñar una solución determinista para el problema del consenso, razón por la cual estaremos estudiando sistemas síncronos, en donde además, supondremos que el sistema distribuido es una gráfica completa.

   El problema del consenso se establece de manera informal como: Cada proceso p_i propone un valor v_i y todo proceso correcto tiene que decidirse sobre algún valor v, en relación al conjunto de valores propuestos. Más precisamente, el problema del consenso se define por las siguientes tres propiedades:

   - Terminación: Cada proceso correcto eventualmente se decidirá por algún valor.
   - Validez: Si un proceso decide v, entonces v fue propuesto por algún proceso.
   - Acuerdo: Ningún par de procesos correctos decide por dos valores diferentes.

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

   - NodoConsenso(Algoritmo 1)

3. Consideraciones:

   - Respeta los constructores proporcionados.
   - Tendremos como parámetro el número de fallos f en la función consenso.
   - Por convención llamaremos $V = \{\ v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7\ \}$ y se tiene $f = 4$, entonces los vértice que fallaran serán$\{v_0, v_1, v_2, v_3\ \}$
   - Simularemos los fallos haciendo que los nodos ya no envíen mensajes ni procesen nada del algoritmo. Pueden hacerlos fallar en cualquier ronda que ustedes quieran, pero recuerden que si un proceso falla en la ronda r en todas las demás rondas siguientes fallará.
   - Noten que este algoritmo es sensible a la ronda actual. Recuerden que el número de ronda en simpy podemos obtenerlo con env.now. Si no logran hacerlo con simpy, son libres de simular este comportamiento usando solo Python. (El comportamiento que tienen que simular es que los nodos solo ejecuten el algoritmo un número fijo de veces).
   - Atributos importantes de un nodo:
     - V : corresponde al arreglo de id's que tiene el nodo. El pseudocódigo original se nos dice que se trata de la referencia al nodo en cuestión, sin embargo para la practicas solo usaremos los id's por facilidad.

  - New : el conjunto de mensajes de cada nodo. De nuevo, en el pseudocódigo original el consenso se hace sobre las referencias a los nodos en sí, por lo que basta con que en este conjunto se tengan de igual manera sólo los id's de cada uno de los nodos y no las referencias de estos.
  - $lider$: valor del id
  - $fallare$: booleano que será verdadero solo si el nodo fallará en algún punto del algoritmo.

4. 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

Consenso para crash failures