PONTIFICIA UNIVERSIDAD CATÓLICA MADRE Y MAESTRA
FACULTAD DE CIENCIAS E INGENIERÍA
ESCUELA DE INGENIERÍA EN COMPUTACIÓN Y TELECOMUNICACIONES

PROGRAMACIÓN PARALELA Y CONCURRENTE

Practica #3

Implementación de un Framework para Topologías de Redes para Procesamiento Paralelo

Objetivo:

El objetivo de esta práctica es que los estudiantes diseñen, implementen y evalúen un Framework para simular y trabajar con diferentes topologías de redes para procesamiento paralelo. Los estudiantes aprenderán sobre los conceptos de topologías de redes y aplicarán sus conocimientos en la práctica para crear una herramienta funcional y útil.

Detalles de la Práctica:

Los estudiantes trabajarán en grupos de máximo dos personas para completar esta práctica. Se espera que cada grupo implemente un Framework que pueda trabajar con las siguientes topologías de redes para procesamiento paralelo:

- Bus Network
- Ring Network
- Mesh Network
- Star Network
- Hypercube Network
- Tree Network
- Fully Connected Network
- Switched Network

Cada topología de red debe ser implementada como una subclase de una interfaz común o una clase base, y debe proporcionar métodos para configurar y ejecutar la topología específica.

Requisitos de Programación Concurrente:

Además de implementar las topologías de redes, los estudiantes deben incorporar programación concurrente en sus implementaciones. Para ello, deberán utilizar herramientas de concurrencia en Java, como Thread, ExecutorService, Semaphore o BlockingQueue.

- ExecutorService: Utilizar ExecutorService para gestionar la ejecución concurrente de los nodos.
- Thread Safety: Asegurar que las operaciones críticas sean seguras para hilos, utilizando mecanismos como synchronized, Locks o Concurrent Collections.
- Simulación Concurrente: Los nodos deben poder enviar y recibir mensajes concurrentemente.

Estructura del Framework:

El Framework debe seguir una estructura modular y extensible. Se sugiere la siguiente estructura básica:

- **Clase NetworkTopology**: Interfaz base que define los métodos comunes para todas las topologías de redes.
- **Subclases para cada Topología de Red**: Implementaciones específicas de cada topología de red, como Bus Network, Ring Network, etc.
- **Clase Node**: Representa un nodo en la red y contiene atributos como ID, vecinos, etc.
- **Clase Message**: Representa un mensaje que se envía entre nodos en la red.
- **Clase NetworkManager**:Gestiona la creación y ejecución de la red, así como la comunicación entre nodos.
- **Clase Main**: Punto de entrada principal del programa, donde se configuran y ejecutan las topologías de redes.

Ejemplo Básico de NetworkTopology

Primero, definimos una interfaz NetworkTopology que todas las topologías de red deben implementar. Esta interfaz tendrá métodos para configurar la red y para enviar y recibir mensajes.

```java
public interface NetworkTopology {
    void configureNetwork(int numberOfNodes);
    void sendMessage(int from, int to, String message);
    void runNetwork();
    void shutdown();
}
```

Implementación de BusNetwork

La clase BusNetwork que sigue la interfaz NetworkTopology. Esta clase simula una red de bus donde todos los nodos están conectados a una línea de comunicación común.

```java
public class BusNetwork implements NetworkTopology {
    private List<Node> nodes;
    private ExecutorService executor;

    @Override
    public void configureNetwork(int numberOfNodes) {
        nodes = new ArrayList<>();
        //...
        executor = Executors.newFixedThreadPool(numberOfNodes);
    }

    @Override
    public void sendMessage(int from, int to, String message) {
        //...
        executor.submit(() -> nodes.get(to).receiveMessage(message));
    }

    @Override
    public void runNetwork() {
        //...
        for (Node node : nodes) {
            executor.submit(node::run);
        }
    }

    @Override
    public void shutdown() {
        executor.shutdown();
    }
}
```

```java
private static class Node {

      public void run() {
          // Simulate node activity
          try {
              Thread.sleep((long) (Math.random() * 1000));
          } catch (InterruptedException e) {
              Thread.currentThread().interrupt();
          }
      }
  }
```

Ejemplo de Uso:

A continuación, se muestra un ejemplo básico de cómo utilizar el Framework para crear y ejecutar una red con la topología de bus (Bus Network):

```java
public class Main {
    public static void main(String[] args) {
        // Crear una instancia de NetworkManager
        NetworkManager manager = new NetworkManager();

        // Configurar la red con la topología de bus y otros parámetros necesarios
        manager.configureNetwork(new BusNetwork());

        // Ejecutar la red
        manager.runNetwork();
    }
}
```

Entregables:

- Código fuente del programa, incluyendo implementaciones para todas las topologías de redes mencionadas. El código debe subirse a un repositorio público en GitHub.
- Documentación detallada que explique la estructura del marco de trabajo, su uso y ejemplos de código. La documentación también debe subirse al repositorio en GitHub.
- El enlace al repositorio público en GitHub debe ser proporcionado en la plataforma de entrega de la tarea. Asegúrate de que el repositorio sea público para poder acceder al proyecto.

Evaluación:

La práctica será evaluada en función de la calidad del código, la completitud de la implementación, la claridad de la documentación y la presentación de los resultados. Además, se valorará la creatividad y la originalidad en la implementación de las topologías de redes.

La valoración de cada miembro del equipo se tomará en cuenta considerando los commits realizados en GitHub. Asegúrense de que ambos miembros del equipo contribuyan de manera equitativa al proyecto.