Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento Ciencia de la Computación
IIC3373 - Programación Concurrente

Tarea 5: Sistemas Distribuidos en Redes ad-hoc

Entrega: Sábado 7 de Diciembre

Introducción:

El objetivo de esta entrega es el de introducir al alumno al diseño de algoritmos en sistemas distribuidos que logren coordinar exitosamente una serie de N sistemas dispuestos en una red ad-hoc.

Una red ad-hoc es una red de dispositivos conectados mediante tecnología inalámbrica que se caracteriza por carecer de una infraestructura determinada. En este tipo de redes cada sistema es capaz de salir y entrar libremente de manera intermitente y por lo tanto el principal desafío es mantener actualizada la información de ruteo para enviar de la manera más eficiente los mensajes a sus eventuales destinatarios finales. Este tipo de redes son de alto interés en el área de las redes móviles debido a que modelan correctamente el comportamiento de los distintos usuarios que pueden moverse libremente, saliendo y entrando de forma intermitente dentro de los rangos de usuarios vecinos.

En una red de estas características, los dispositivos no son capaces de mandar directamente mensajes a todos los participantes. Cada sistema solo conoce a sus vecinos y solo puede mandarle mensajes a ellos. Para enviarle un mensaje a otro sistema, este debe ser alcanzable enviando el mensaje de vecino en vecino. Por ejemplo, si los pares {A, B} y {B, C} son vecinos y A quiere mandarse un mensaje a C, entonces A debe primero enviárselo a B y este debe después re-enviárselo a C. El número de veces que el mensaje debe viajar entre pares de vecinos para llegar a su destino final se denomina saltos o “hops”.

Lamentablemente esta situación es extremadamente difícil de replicar en una tarea pues un gran número de variables interviene. Por lo tanto nos limitaremos a una simulación con dos supuestos importantes:

1. Cada sistema dentro de esta red será representado por un proceso dentro de un mismo computador (IMPORTANTE: nótese que usaremos procesos, no threads; entiéndase proceso como una instancia de un programa).
2. La arquitectura de la red se mantendrá estática. Es decir, asumiremos que los vecinos de cada sistema permanecen constantes y que todo mensaje enviado tiene 100% de probabilidades de llegar a un vecino (este no es el caso en las redes ad-hoc estándares pues un destinatario puede salir de la red después de haberse enviado el mensaje). CONSEJO: Aproveche este hecho para resolver la tarea.

Descripción:

En esta tarea usted deberá simular la situación antes descrita y coordinar una serie de procesos corriendo en paralelo para realizar un trabajo común.

En primer lugar usted debe elegir algún problema paralelizable para probar su programa. Este debe poder ser descompuesto en tareas más pequeñas que cada proceso pueda resolver de manera independiente, y luego enviarle el resultado a un proceso principal que los pueda juntar. Usted puede elegir cualquier problema que desee que cumpla estas características. Ejemplo: multiplicación de matrices.

En segundo lugar, usted debe hacer un programa que inicialmente recibirá como parámetros mediante línea de comandos:

1. Si acaso es el proceso principal o no.
2. Un id único.
3. Los ids de los vecinos que él conoce.
4. Otros datos que usted estime convenientes.

Múltiples instancias de este programa serán ejecutadas en paralelo con un conjunto distinto de los parámetros indicados anteriormente.

El proceso principal será quien inicie el trabajo del problema paralelizable definido en primera instancia. Las otras instancias inicialmente no lo conocen. Este programa deberá descomponer este problema en subproblemas más pequeños, enviárselo al mayor número posible de procesos alcanzables dentro de la red siguiendo la restricción de vecindad y luego resolver el subproblema que le corresponde a él. Eventualmente deberá recibir la respuesta de todos los demás integrantes de la red y unir el resultado de cada subproblema para obtener el resultado final. El resultado final puede imprimirse por consola o en un archivo.

La definición de este problema paralelizable puede estar incluida en el código, o bien usted puede ser más creativo e incluirla dentro de un archivo que se lea por reflection para poder definir un problema genérico cualquiera (esto último tiene bonus).

Código Base:

Esencial para el desarrollo de esta tarea es un programa adicional que se encargue de ejecutar las N instancias del programa que usted debe desarrollar y que implemente una infraestructura de paso de mensaje que tenga conocimiento de las relaciones de vecindad. Para facilitarle el trabajo, se ha puesto a disposición de los alumnos un proyecto eclipse hecho en Java que ya tiene esto implementado más un mini proyecto adicional que demuestra cómo utilizar el paso de mensajes.

Este programa permite definir relaciones de vecindad de forma aleatoria o bien leerlas desde un archivo llamado test.txt. En el primer caso se asumen 5 procesos donde cada relación de vecindad tiene un 50% de probabilidades de existir o no. En el segundo, se debe definir una matriz donde la primera línea es el número de procesos N a ejecutar en paralelo y las siguientes líneas definen una matriz simétrica M de N x N de 0 y 1 donde M(a,b) == 0 implica que a y b no son vecinos y M(a,b) == 1 implica que a y b sí son vecinos.

El sistema de paso de mensajes está basado en sockets utilizando a este proyecto base como hub común por donde todos los mensajes deben pasar (es decir todas las instancias establecen un socket con este proyecto base en vez de entre sí). Al iniciar cada instancia, se añade dentro de los parámetros de la línea de comandos el número del puerto con el cual comunicarse.

Los programas que se ejecutan son JARs ejecutables.

Usted es libre de modificar el código de este proyecto como guste, o bien ignorarlo por completo y hacer su propia base con una arquitectura de paso de mensajes completamente distinta, e incluso en un lenguajes distinto. Si decide esto último debe incluir una manera de definir manualmente las relaciones de vecindad. No obstante, esta parte no será evaluada.

Nótese que este proyecto fue probado en el sistema operativo MAC y requiere de varios pequeños cambios para funcionarle a usted. En particular es necesario modificar el nombre y la ruta del programa del cual ejecutar las múltiples instancias.

Restricciones:

Usted puede desarrollar su tarea en Java, C# o C++. Sin embargo solo habrá un código base en Java. Si usted desea usar los otros lenguajes deberá hacer su propio código base.

Evaluación:

El puntaje se divide en tres ítems. En primer lugar, igual que en la tarea 3, usted debe seleccionar una de las siguientes alternativas. Si su programa cumple con lo especificado a plenitud, obtiene todo el puntaje; si no, recibe 0 puntos. Sea prudente en seleccionar una alternativa apropiada. No elija una que da más puntos si no la logra cumplir.

1. Existe un proceso con la característica de “principal” que recibe el problema paralelizable, lo resuelve e imprime. (1 punto)
2. El proceso “principal” descompone el problema paralelizable en subproblemas y lo comparte solo con sus vecinos. Cada instancia luego resuelve su parte del problema, pero las soluciones no se juntan. (2 puntos)
3. El proceso “principal” descompone el problema paralelizable en subproblemas y lo comparte solo con sus vecinos. Estos le devuelven el resultado al principal quien los une y lo imprime. (3 puntos)
4. El proceso “principal” descompone el problema paralelizable en subproblemas y lo comparte con todos sus vecinos alcanzables directa e indirectamente. Estos le devuelven el resultado al principal quien los une y lo imprime. (4 puntos)
5. El proceso “principal” logra descubir el tamaño N’ de la red de vecinos alcanzables (que podría ser inferior al N original si hay vecinos no alcanzables), descompone el problema paralelizable en N’ subproblemas de igual tamaño y lo comparte con todos sus vecinos alcanzables directa e indirectamente. Estos le devuelven el resultado al principal quien los une y lo imprime. (5 puntos)

El segundo ítem corresponde a ingeniería del software y se evalúa en 1 punto. El tercer ítem corresponde a un informe breve (no debiese ser más de 2 páginas, pero si gusta puede hacerlo más largo) donde explique de manera resumida cómo resolvió el problema.

Bonus:

Usted puede obtener un bonus de 1 punto si el problema paralelizable a resolver puede definirse en un archivo aparte que pueda ser leído por reflection.

Como verá, la nota máxima es un 9.

Plazos y entrega:

El plazo para entregar esta tarea es el día sábado 7 de diciembre a las 23:59. Se descontará un punto por cada hora o fracción de atraso.

IMPORTANTE: Ha habido un mal hábito de no cumplir los plazos durante el semestre, por eso esta vez se les está dando mucho tiempo de partida. Pero debido a que las notas deben estar listas el 11 de diciembre y la tarea debe ser corregida y luego debe darse la posibilidad de recorregir, no será posible bajo ninguna circunstancia extender este plazo.

Los archivos que usted debe entregar son:

- El código escrito.
- El código base modificado, o bien uno nuevo hecho por usted.
- El informe donde explique como resolvió el problema.
- Un archivo de texto .readme en donde explique cómo utilizar su programa, se diga cuál es el problema paralelizable seleccionado por usted para probarlo y donde precise la alternativa de evaluación seleccionada. IMPORTANTE: En caso de que usted omita este archivo o bien sus instrucciones sean poco claras, el ayudante hará lo posible por evaluarlo pero si al final no se puede, se le colocará automáticamente un 1 y usted deberá venir a recorrección para explicar su solución.

Pueden enviar estos entregables comprimidos en un .zip o un .rar via mail a jibenedettoc@gmail.com.

En caso de problemas de envío por cualquier motivo, se aceptarán vías secundarias de entrega como por ejemplo compartir un Dropbox o subir su archivo a un servicio de File Sharing en línea (rapidshare, etc.) siempre y cuando se realice antes del plazo límite. Recuerden en este caso enviar un mail al ayudante con copia al profesor detallando los problemas que tuvieron al realizar el envío.