Algoritmos Distribuidos a Regiones

Gómez C. Roberto, Magallanes G. Gabriela
IEEE member,
ITESM-CEM Departamento de Ciencias Computacionales, Apdo. Postal 50, Módulo Servicio Postal,
Atizapán Zaragoza, 52926, México
rogomez@itesm.mx, A00471174@itesm.mx

Resumen: Existe una gran variedad de algoritmos distribuidos, la mayoría de estos cuentan con un proceso iniciador. Sin embargo es posible que no se cuente con un solo algoritmo iniciador sino que el algoritmo requiera que varios procesos inicien un mismo cálculo al mismo tiempo. En este trabajo se presentan las características principales de este tipo de algoritmos, que llamamos algoritmos distribuidos a regiones así como una implementación de estos.

1. Introducción

Un sistema distribuido es una colección de interconexiones de computadoras, procesos o procesadores interconectados entre sí. Para ser considerados como autónomos, los nodos deben contar con control privado. Para ser considerados como interconectados, los nodos deben ser capaces de intercambiar información.

Un algoritmo distribuido se basa en el envío de mensajes hacia un proceso remoto. Los procesos van a actuar dependiendo del mensaje que llegue o que envíen.

En este trabajo se dan a conocer las características delos algoritmos multiolas, sus ventajas conveniencia de su uso y un ejemplo de este. Por último, en base al esquema de programación distribuida tipo ola recursiva se implantó un algoritmo a regiones para analizar su comportamiento.

El presente trabajo se encuentra dividido de la siguiente forma, en la siguiente sección se dan a conocer las características de los algoritmos distribuidos a regiones y multiolas En la sección tres da a conocer una solución del problema de elección y de árbol de cobertura usando este tipo de un algoritmos. La cuarta sección presenta una implementación de un sistema de intercambio de información basado en este tipo de algoritmos. Por último, se presentan las conclusiones.

2. Los algoritmos distribuidos a regiones y los algoritmos multiolas

Se define un algoritmo a regiones como aquel que cuenta con más de un iniciador. El primer algoritmo de este tipo es la solución presentada por Gallager, Humblet y Spira para la solución del problema de expansión de árbol de peso mínimo [7]. A diferencia de este algoritmo nuestro enfoque se basa en el uso de la ola recursiva para poder definir las regiones.

Los algoritmos de tipo ola recursiva son una combinación de los algoritmos tipo ola, presentados por primera vez en 1982 por Chang Ernert [2], y del concepto de recursión en ambientes distribuidos. Estos tipos de algoritmos están asociados al concepto de difusión de calculo, y esta compuesto de dos fases. La primera fase permite una propagación, (broadcasting), de información a todos los sitios que componen una red; la segunda realiza un recolecta de resultados. La propagación utiliza el mecanismo RPC para su implementación. Encontramos diferentes trabajos sobre estos algoritmos [3], [4], sin embargo en [5] se presenta un nuevo esquema denominado ola recursiva.

Cada uno de los diferentes nodos iniciadores, i.e. la raíz del árbol generado por la ola, va a construir una región de cálculo.

2.1. Los algoritmos multiolas

Un algoritmo a regiones basado en olas recursivas, implica que una o más olas se encuentren en ejecución al mismo tiempo, razón por la cual en estos casos hablamos de algoritmos multiolas. Esto provoca que exista más de un iniciador, que la etapa de subida y bajada de diferentes olas se encuentran mezcladas y que se cuente con un árbol de ejecución asociado a cada ola recursiva.

No hay que confundir el concepto de multiolas con el de tren de olas [4], la diferencia entre un tren de olas y un algoritmo multi-olas, reside en el momento de la ejecución de la segunda ola. En un tren de olas una nueva ola comienza su ejecución sobre el mismo iniciador una vez que todos los otros procesos terminaron con la primera. Si una tercera es necesaria, tendrá que esperar que la segunda termine. Por otro lado, en un algoritmo multi-olas, en cualquier momento del cálculo una nueva ola puede comenzar su ejecución, a partir de cualquier sitio de la red.

Se considera que la red es conexa y que los identificadores de los procesos son diferentes. No existen dos olas con el mismo identificador (i.e. una ola no tiene más que un solo iniciador). Las líneas de comunicación son fiables y no hay alteración de mensajes. Cada sitio que lanzó una ola, es considerado

como un candidato a ser el sitio elegido y la ola toma como identificador el del identificador del sitio que lo lanzó.

En estos sistemas varias olas pueden estar presentes durante la ejecución. Cada ola va a propagarse sobre toda la red. Si una ola llega a un sitio ya visitado por otra ola, debe decidir si continua o si detiene su propagación. Esta decisión es tomada en función de la relación de orden de los identificadores de la ola. Si la ola es mejor que la que visitó anteriormente el sitio, esta continua, sino se detiene. Esto nos lleva a definir tres tipos de olas: la ola dominante, una ola en plena propagación y la ola dominada.

La ola dominante o ganadora es una ola que se propaga, hasta llegar a todos los sitios generados. Por construcción de su grafo asociado, se genera un árbol de cobertura, cuyo iniciador es el elegido. Una ola puede ser dominante sobre un sitio y dominada en otro sitio. Dominante y dominada son propiedades locales, ganador es una propiedad global. Independientemente de los iniciadores siempre hay una ola ganadora.

En un principio, todas las olas son olas dominantes, a medida que se encuentran entre ellas las olas dominadas comienzan a aparecer. Al final del cálculo solo existirá la ola ganadora en la red, y ninguna información con respecto a otra ola.

3. El problema de elección y de árbol de cobertura

Desde el punto de vista de control dentro de los sistemas distribuidos, la elección y la construcción de un árbol de cobertura son problemas fundamentales. En efecto, el árbol de cobertura permite de estructurar el conjunto de procesos que forman el sistema distribuido en una topología de árbol. De esta forma el sistema cuenta con un sistema de control y de ruteo de mensajes dentro del sistema distribuido. Por otro lado, la elección es una forma de proporcionar un privilegio a un sitio. Desde un punto de vista teórico, el algoritmo de elección consiste en pasar de un sistema compuesto de procesos, todos en un mismo estado, a un sistema en el cual un sólo y único proceso se encuentra en el estado elegido. El primer algoritmo conocido de elección es el de LeLann [8] se han propuesto numerosas soluciones para, aparte de obtener un árbol de cobertura, realizar una elección dentro de un sistema distribuido. En estas soluciones el sitio elegido es la raíz de la topología de árbol.

Para un mejor análisis del funcionamiento del algoritmo se analizarán cada una de las etapas de un algoritmo de tipo ola, tal y como se define en [6]. En la bajada se propagan las diferentes olas y en la subida se informa a los iniciadores si la ola que inicializaron es dominante o dominada. Desde el punto de vista de la construcción de un árbol de cobertura, la bajada define el padre de cada sitio y elige los hijos posibles; mientras que la subida confirma si los sitios vecinos elegidos pueden convertirse en hijos o no.

3.1 La bajada

La bajada esta ligada directamente con la propagación de la ola, es decir una vez que la propagación de la ola termina, la bajada también. En un principio, todos los sitios están en un estado dormido, después un cierto número de procesos, los iniciadores, se despiertan y activan a sus vecinos a través de RPCs .

Como se dijo anteriormente los algoritmos a regiones cuentan con más de un iniciador. Entonces puede existir más de una ola ejecutando su fase de bajada. Para poder diferenciarlas, cada ola debe portar un identificador. Lo más simple es que el identificador sea el del sitio que lanzó la ola y un número de identificación de secuencias de la ola.

En el caso de los algoritmos multi-olas varias olas atraviesan un mismo sitio durante la construcción del árbol de cobertura. Una vez que una ola llega a un sitio hay dos operaciones a realizar, la actualización del padre y el cálculo de los sitios a los cuales la ola será propagada

En la actualización del padre los procesos no iniciadores pasan del estado dormido al estado despierto cuando son activados o alcanzados por el primer RPC. Un proceso que esta en el estado dormido y que recibe un RPC por la primera vez, pasa al estado despierto y forma parte de la ola (y por consecuencia del árbol) asociada al RPC. Una vez que el proceso forma parte de una ola, considera como su padre aquel que le envío el RPC, y como sus posibles hijos a sus vecinos. Cuando un proceso se encuentra en un estado despierto y recibe un RPC relacionado con otra ola, decide a que ola pertenecer en función del identificador. Si el RPC corresponde a una ola dominante, el proceso será parte de esta ola y cambia el identificador de su padre por el del emisor del RPC, el antiguo padre es considerado como un hijo posible. Por consecuencia, por cada ola dominante que llegue a un proceso, puede

haber un nuevo padre. Un algoritmo de construcción de un árbol de cobertura se debe asegurar que una vez terminado, cada sitio cuenta con un padre. Un ejemplo lo encontramos en la figura 1. El sitio x tiene recibe a la ola 1 por primera vez y toma como padre al sitio b. Sin embargo, con la llegada de ola dominante 2, el sitio tiene que cambiar a su padre por el sitio a.

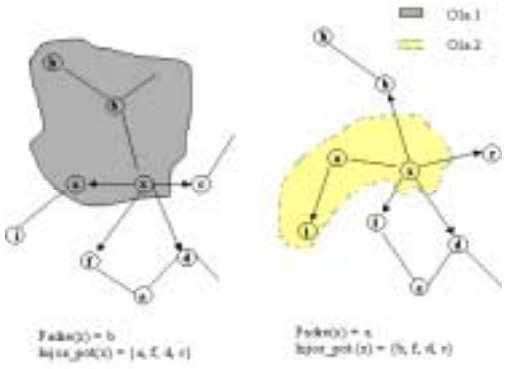

Figura 1. Ejemplo de dos olas sobre un mismo sitio.

Una vez que el sitio ha actualizado a su padre, debe calcular los sitios a donde la ola será propagada. Lo más simple es enviar la ola a los sitios vecinos a excepción del padre. Sin embargo si hay más de una ola, habrán varios árboles en la topología, cada uno asociada a una ola. En ese caso, los árboles construidos por las olas dominantes son considerados como subárboles para su propagación, ver figura 2. Inicialmente tenemos todos los nodos interconectados y en estado “dormido”, una ola llega al sito x y se propaga, después de la propagación algunas ligas entre nodos desaparecen, posteriormente otra ola llega al sitio x y se propaga por el camino definido por la ola anterior.

Cada sitio administra una variable de tipo conjunto que contiene los sitios a los cuales la ola se debe enviar. Sea E esta variable, en un principio contendrá todos los vecinos del sitio i, entonces: $E_i = (V_1, V_2, \ldots, V_n)$, donde V_1, V_2, ..., V_n son los identificadores de los vecinos del sitio.

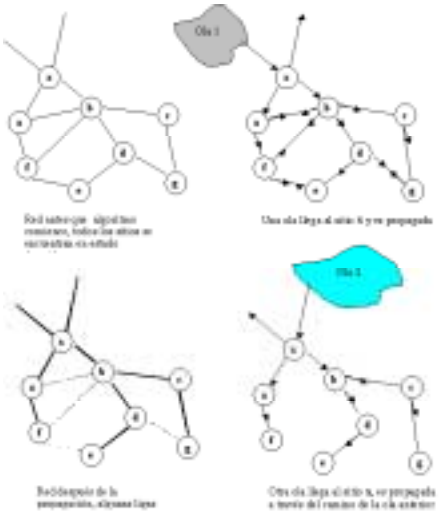

Figura 2 Ejemplo de calculo de hijos

Donde V1, V2, ..., Vn son los identificadores de los vecinos del sitio

A la llegada de la primero ola, el sitio emisor de esta ola (i.e. el padre), es suprimido de E y la ola es propagada al resto de los sitios, que son considerados como posibles hijos del sitio i. Cada vez que un sitio V_i responde que no puede ser hijo del sitio i, es suprimido del conjunto E.

Cuando una ola dominante llega al sitio i, la variable E es modificada. Las operaciones a realizar son:

1) Añadir al antiguo padre.
2) Suprimir al nuevo padre

Lo anterior equivale a considerar al antiguo padre como un posible hijo. Una vez que la variable E es actualizada, la ola es propagada sobre los sitios que son incluidos en la variable E. De esta forma no se propaga sobre los sitios ya eliminados.

El cálculo de los hijos se hace a partir de la variable E, esta variable contiene los vecinos a los cuales la ola será propagada. Si uno de estos sitios no puede convertirse en hijo, a causa de las condiciones presentadas anteriormente, será eliminado de esta variable. Una vez que el sitio pasa al estado consiente, él considera como sus hijos, los sitios contenidos en E.

Es importante definir un criterio para detener la propagación de la ola. La ola se propaga por inundación y se detiene si:

1. No hay más sitios a los que enviar la ola.
2. La ola encuentra un sitio que pertenece a la misma ola, o a una ola dominante.

Una vez que la ola se detiene estamos seguros que todos los sitios que ella toca en su propagación se encuentran en estado despierto.

El algoritmo es considerado como terminado cuando todos los sitios conocen al sitio elegido así como su padre y sus hijos en el árbol de cobertura. Esto equivale a decir que todos los sitios están en un estado consciente.

El primer sitio a estar dentro del estado consiente es el iniciador de la ola dominante. Es este el que informa a los otros sitios que el algoritmo terminó e informa el identificador del sitio elegido.

Cuando una ola dominante encuentra una ola dominante debe detener su propagación y enviar como resultado una respuesta de perdida. Esta respuesta será pasada de hijo a padre hasta llegar al iniciador de toda la ola dominante. Entonces, si el iniciador de una ola no recibe una respuesta de perdida, significa que no ha encontrado de una ola dominante durante su propagación.

Dado que la red es conexa, podemos estar seguros que una ola va encontrar al menos una ola dominante. Tomando en cuenta que todas las olas cuentan con identificadores diferentes, es seguro que una sola ola dominará a todas las otras.

Una vez que la propagación de la ola ha terminado comienza una nueva fase que denominamos la remontada, la cual termina cuando la ola llega al sitio iniciador. El objetivo principal de la remontada es informar a los padres de los sitios si la ola es dominada o dominante, así como precisar si un sitio puede convertirse en el hijo del que le envío el RPC. Más adelante mostraremos que las dos informaciones están relacionadas y que respondiendo a una podemos concluir la otra.

En un primer tiempo vamos a analizar las respuestas a la proposición de convertirse en hijo. Después de que un sitio ha propagado la ola a través de sus vecinos, este puede recibir tres posibles respuestas:

1. El sitio no puede ser un hijo debido a que pertenece a una ola dominante.
2. El sitio no puede ser un hijo porque forma parte de la misma ola.
3. Puede convertirse en hijo.

A partir de la respuesta el padre sabe si el RPC que envío corresponde a una ola dominante o a una ola dominada. El hecho de que un sitio no pueda convertirse en el hijo de otro no significa que la ola esta dominada

3.2 La subida

La subida comienza una vez que todos los resultados son recibidos. Un sitio puede propagar la ola a través de los mismos sitios varias veces, entonces nos tenemos que asegurar que los resultados recibidos corresponden a la propagación (i.e. bajada) correcta. Esto puede dejarse a la semántica del RPC o a la naturaleza misma de la recursividad, la elección dependerá del ambiente en el cual la ola sea implementada.

Es posible que una o más olas no terminen su propagación, pero habrá al menos una bajada completa. Es decir que al menos una ola visitará todos los sitios.

Durante un cálculo una subida puede encontrarse con la bajada de otra ola. Entonces, la subida debe respetar los argumentos siguientes: la subida comienza en las hojas de los árboles de ejecución, las propiedades a probar serán las mismas para todas las olas y establecer la condición de terminación para todas las olas.

Varias olas se encuentran atravesando la red. Un sitio puede recibir respuestas que corresponden a olas diferentes. Para evitar confusiones se detecta a que ola corresponde cada respuesta, comparando el valor de la variable que es un parámetro por referencia del RPC.

Para evitar problemas de incoherencia se diferencian los sitios de propagación con los hijos de un sitio.

4. Implantación de un sistema de intercambio de información basado en algoritmo a regiones

En las secciones anteriores se presentó un ejemplo de diseño de un algoritmo a regiones basado en olas

recursivas. En esta sección se presenta la implementación de un algoritmo de intercambio de información.

La red en la cual se implantó el sistema, esta compuestas de equipos SUN, localizados en dos laboratorios. Los nombres de las máquinas son países del continente americano (México, USA, Guatemala, etc.)

La topología lógica de la red se puede observar en la figura 3. Se cuenta con tres nodos iniciadores, Jamaica, México y Ecuador, cada uno de ellos provoca la propagación de una petición hacia otros nodos, en el caso de Ecuador por ejemplo hacia Antigua, Surinam y Guatemala.

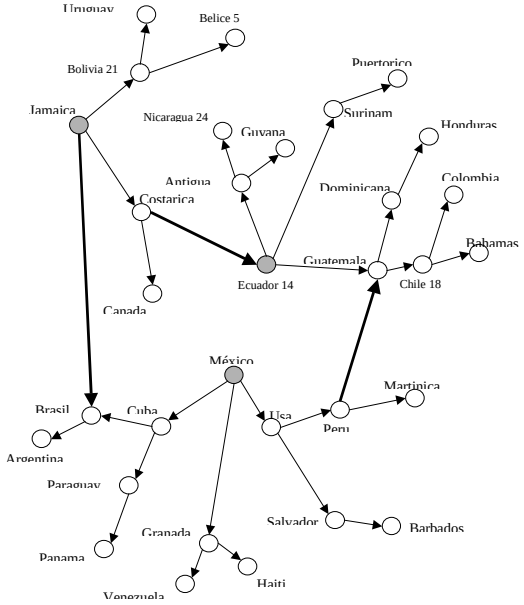

Figura 3. Topología de la implementación

Como se mencionó esta topología es lógica y fue diseñada para los fines del proyecto, los equipos están conectados en una red local ethernet.

4.1 El algoritmo de intercambio de información

El intercambio de información que se utilizó para este sistema es el reporte de procesos corriendo en el nodo, simulando la búsqueda del nodo con menos carga al que se le puedan hacer peticiones posteriores.

Un proceso iniciador solicita a otro nodo los procesos que tiene corriendo, éste a su vez hace la misma petición hacia otros nodos, propagándose bajo una estructura de árbol predeterminada para cada nodo iniciador. Así por ejemplo Jamaica, que es un nodo iniciador, solicita a sus "hijos", Bolivia, Costarica y Brasil, corran la petición denominada "obten_procesos" que obtiene el número procesos que están en ejecución.

Pero a su vez éstos buscan a sus nodos hijos para solicitar ejecuten la misma petición, y de esta manera Bolivia solicita a Uruguay y Belice ejecuten "obten_procesos", estos son nodos hojas así que su labor se limita a ejecutar y devolver el resultado a quien lo invocó en este caso Bolivia. Bolivia a su vez selecciona el número menor para reportarlo a su nodo padre, Jamaica que concentra los resultados de todos sus hijos.

Se puede observar que la metodología utilizada es la de ola recursiva, pues como se definió con anterioridad, este método hace propagaciones de un mismo procedimiento en nodos diferentes. En este caso se provoca la ejecución de "obten_procesos" en diferentes nodos.

Se tienen tres nodos iniciadores, México, Jamaica y Ecuador. Estos forman un área de influencia o región, pero estas regiones "chocan" al momento de propagarse, y es aquí donde se puntualizó en es este estudio, ¿qué hacer cuando las regiones se intercepten? Se plantearon tres alternativas para cada punto de contacto entre regiones.

La primera alternativa se planteó en el "choque" de la región de Jamaica y la región de México, ambas regiones se expanden y tienen un nodo común: Brasil. Jamaica tiene a Cuba como hijo inmediato, mientras que México lo hace a través del nodo Cuba (ver figura 4). La forma de atacar este "choque" fue a través de un "apartado" del nodo. Es decir, cuando Jamaica o Cuba intenten propagarse solo uno de los dos lo logrará el que llegué primero, a través de una bandera se sabrá la disponibilidad de Brasil, el nodo que llegue y lo encuentre ocupado se regresará sin resultado, mientras que aquel que lo encontró disponible esperará a que este termine y le devuelva el resultado.

Como segunda opción, en un traslape de regiones, se planteó el ignorar si el nodo a acceder está ocupado o influenciado por otra región, solicitándose la propagación. Esta alternativa se implantó en las regiones de Ecuador y México (ver figura 4), estas regiones se interceptan en el nodo Guatemala, Ecuador

directamente hacia Guatemala y México vía el nodo Usa, en ambos casos las dos regiones harán su propagación y esperarán a que Guatemala conteste, sin importar si alguno de ellos lo accedió antes como en el caso anterior. Casi siempre los resultados son diferentes pues fueron hechas las peticiones en tiempos distintos y las condiciones son diferentes.

Figura 4. Ejemplo de traslapamiento de olas

Finalmente el último caso de estudio que se implantó se refiere al aprovechamiento del trabajo hecho por otra región que se propagó previamente. Las regiones de Ecuador y Jamaica se traslapan justamente en el nodo Ecuador que como se observa en la figura 4, este caso se planteó haciendo esperar a Jamaica, vía el nodo Costarica, a que la región de Ecuador termine su procesamiento y enviarle su resultado al nodo Jamaica y este a su vez termine con su proceso.

4.2 Pruebas de la aplicación distribuida

Las pruebas del sistema se hicieron en forma modular, conforme se iban realizando avances al mismo. En primer lugar se probaron en forma local las funciones, las que obtiene el número de procesos y la que ordena los datos. Una vez probadas se procedió a hacer las pruebas de comunicación entre dos nodos, luego un nodo con dos o tres y finalmente una región. Como se puede observar, una región tiene forma de árbol donde la raíz es el nodo iniciador. Las pruebas posteriores se enfocaron a verificar las decisiones en el traslape de las regiones. Es decir identificar si un nodo había sido accedido por la influencia de otra región diferente a la región que lo quiere acceder, esto se hizo mediante la identificación del archivo generado para guardar la información solicitada si este ya existía el nodo ya había sido accedido de otra forma no.

En la gráfica de la figura 5, se observan tres traslapes, en cada uno de ellos se implantó cada una de las posibles situaciones, ignorar y continuar, retroceder sin solicitar información y finalmente aprovechar lo que la otra región hizo, como se explicó anteriormente.

5. Conclusiones

Se ha presentado un nuevo tipo de algoritmos distribuidos conocidos como algoritmos distribuidos a regiones. El esquema fue probado e implementado presentando un buen desempeño.

El proyecto da una estrategia de distribución de tareas a través del algoritmo Ola Recursiva y la metodología de programación RPC con la variante que se hizo con un enfoque hacia la creación de regiones, que en este caso se refiere a la agrupación de nodos o equipos, que involucra la decisión de que alternativa tomar ante el cruce de otra región.

Si bien el algoritmo Ola nos permite hacer la propagación de un mismo proceso en otros nodos vecinos, éste no incluye la decisión o decisiones de traslape con otra región que en nuestro caso fue implantada a través de la agrupación de nodos o equipos a los que denominamos regiones. Nos vimos obligados a agregar al algoritmo una fase de toma de decisiones. Se plantearon tres alternativas que se describieron en este trabajo. Y en cada caso implantado se obtuvo un resultado, con la consideración de que tipo de dato se estaba consiguiendo.

Dado que número de equipos involucrados es finito y conocido no se observa una ganancia importante el elegir cualquier opción, pero el sistema puede extrapolar su funcionamiento a un número mayor o desconocido de equipos y las ventajas saltan a la vista. Se puede optar por retroceder y no avanzar ante la posibilidad de que el equipo en ese momento se encuentre ocupado y no pueda contestar en un periodo razonable para quien solicita un servicio.

La opción de aprovechar los resultados obtenidos por una petición anterior diferente a que lo esta solicitando puede funcionar en casos como el que no es necesario contar con información actualizada al 100%. Lo anterior podría deberse al hecho de que el resultado que se está arrojando no se haya obtenido en un lapso

"corto" o inmediato anterior a la petición y es probable que la situación haya cambiado.

La última opción es más completa, pues ordena volver a hacer las peticiones no importando que éstas hayan sido solicitadas, garantizando la "frescura" de la información.

La opción a seguir será tomada en cada caso cuando la situación se presente y debe de considerar los factores como tiempo, oportunidad de la información, así como actualización de datos para hacer la petición.

Referencias

[1] Recursive Distributed Programming Schemes, Florin Gerard, Gómez Roberto, Lavallée Ivan, ISADS'93, Marzo-Abril 93, Kawasaki Japón

[2] Chang Ernest –Echo algorithms: depth parallel operations on general graphs. IEEE Trans. On Soft. Eng. Vol. Se-8 No 4 July 1982.

[3] Tel Gerard. Introduction to Distributed Algorithms. Cambridge Universite Press, 1994.

[4] Raynal M. Helary J. M. Synchronization et contrôle des systémes et des programmes répartis. Eyrolles, Paris 1998.

[5] Gómez Cárdenas Roberto, La recursión como un estilo de programación dentro de los sistemas distribuidos, XXVI Reunión de Investigación y Desarrollo Tecnológico del Sistema ITESM, pp. 563-572 12 Enero 1996, Monterrey México

[6] Systematic Building of a Distributed recursive Algorithm, example: The shortest path problem, Florin Gerard, Gómez Roberto, Lavallée Ivan, Acta Mathematica Vietnamica, Vol. 20, No. 1, 1995,

[7] Gallager R. G., Humblet P.A., Spira P.M. A distributed algorithm for minimum spanning trees, ACM Trans. On Prog. Lang. Adn Systems Vol 5:66-77, Jan, 1983.

[8] G. LeLann. Distributed systems, towards a formal approach. In IFIP Congress Proceedings, pages 155-160, 1977

Roberto Gómez Cárdenas

Profesor-investigador del ITESM-CEM en el departamento de ciencias computacionales. En 1987 recibión el titulo de Ingeniero en Sistemas Electrónicos por parte del ITESM-CEM. La misma institución le otorgo su grado de maestría en 1990. Recibe el diploma DEA en 1991 por parte de la Universidad de Paris 7 y en 1995 recibe su doctorado por la Universidad de Paris 8. Es miembro de la IEEE desde hace más de 10 años. Sus prinicpales areas de interes son los sistemas distribuidos, la seguridad informática y la criptologia.

Dirección del autor: Apdo. Postal 50, Módulo Servicio Postal, Atizapán Zaragoza, 52926, México México.
email: rogomez@itesm.mx

Gabriela Magallanes Gonzalez

Egresada de la UNAM como Ingeniera en Computo en 1992. Maestría en Ciencias Computacionales por el ITESM-CEM en 2002. Actualmente labora como líder de proyectos en Bancomer relacionados con sistemas distribuidos.

Dirección del autor: Apdo. Postal 50, Módulo Servicio Postal, Atizapán Zaragoza, 52926, México México.
email: A00471174@itesm.mx