# Una mejora del framework SNMP de equilibrio de carga para controlar los computadores de la WLAN en zonas de cobertura reducida

David Sánchez, Elsa M. Macías, Álvaro Suárez
Grupo de Arquitectura y Concurrencia (GAC)
Departamento de Ingeniería Telemática
Universidad de Las Palmas de Gran Canaria
Campus Universitario de Tafira, 35017. Las Palmas de Gran Canaria, España
{dsanchez, emacias, asuarez}@dit.ulpgc.es

***Abstract****. A network formed by desktop and portable computers is a useful environment for doing parallel computing. In this infrastructure we implement Master/Slave parallel distributed programs which exhibit strict data dependences among iterations and parallel calculations inside an iteration. Due to the dynamic behaviour of portable computers is necessary to keep in mind that they can move out of coverage area at any time. In a previous work, we developed a load balancing software framework based on Simple Network Management Protocol (SNMP) that considers the beacon strength in the portable computers for executing this kind of applications efficiently. However, when a portable computer is located in a limited coverage area, our framework considers that this resource is unavailable, and therefore it can't be used. For that reason, in this paper we present a mechanism that improves the performance of our framework SNMP, allowing us to use the portable computers while there is a wireless link.*

## 1 Introducción

Actualmente, el auge espectacular en la demanda de computadores portátiles con alto rendimiento, y la aparición de nuevos estándares de comunicación en la familia de protocolos IEEE 802.11, permiten una combinación efectiva de las redes de área local inalámbricas (WLAN) con las tradicionales redes de área local (LAN) para realizar computación paralela y distribuida [1], siendo ésta uno de los nuevos desafíos en los próximos años [2].

Nosotros hemos demostrado que la computación paralela en un entorno LAN-WLAN resulta eficiente para ejecutar aplicaciones Maestro/Esclavo con dependencias estrictas entre iteraciones, donde los cálculos paralelos son implementados en cada iteración [3]. Resulta evidente, que la heterogeneidad presente en este entorno de computación (diferentes arquitecturas, potencias de procesamiento y estándares de comunicación) lleva consigo una tarea ardua y difícil para ejecutar de forma eficiente de dichas aplicaciones. Si no se consideran técnicas de equilibrio de carga, algunos procesos del programa paralelo pueden caer en estados ociosos mientras otros están calculando.

En nuestro grupo de investigación hemos desarrollado una estrategia de equilibrio de carga [4][5] para aplicaciones Maestro/Esclavo que se desarrollan en un entorno de computación LAN-WLAN. Esta estrategia utiliza información acerca del rendimiento de los computadores y de la ejecución actual de cada proceso paralelo para estimar la distribución de datos adecuada. La recopilación de esta información tiene que ser obtenida en un segundo plano para no degradar el rendimiento de la aplicación paralela. En el trabajo realizado en [6], se utiliza el protocolo de gestión de red SNMP [7] para recoger de forma eficaz la información de rendimiento de los recursos fijos que forman parte del entorno de computación. En [5], desarrollamos una arquitectura software basada en el protocolo SNMP que obtiene de forma eficiente los parámetros de rendimiento de los computadores del entorno de computación LAN-WLAN. Estos parámetros son utilizados para llevar a cabo el equilibrio de carga en presencia de computación y comunicación heterogénea. En [8] desarrollamos una biblioteca que implementa la combinación de la estrategia de equilibrio de carga con la arquitectura SNMP, y por lo tanto, abstrae al usuario del conocimiento y de la implementación de nuestra arquitectura.

En este tipo de entornos, el comportamiento dinámico de los computadores portátiles implica que se tengan que aplicar técnicas de control que consideren los parámetros sobre la calidad del enlace inalámbrico y la energía de la batería. Dicha consideración viene determinada por el hecho de que no se debe esperar por resultados que no van a llegar, ya sea porque el recurso está fuera de cobertura, o porque la energía remanente en la batería no sea suficiente para finalizar la iteración en curso. En [9] nosotros modificamos nuestro *framework* SNMP para ejecutar eficientemente aplicaciones paralelas donde la estrategia de equilibrio de carga tiene en cuenta el nivel de potencia del enlace inalámbrico y la energía de la batería de los computadores portátiles. En dicho trabajo, cuando un recurso de computación está

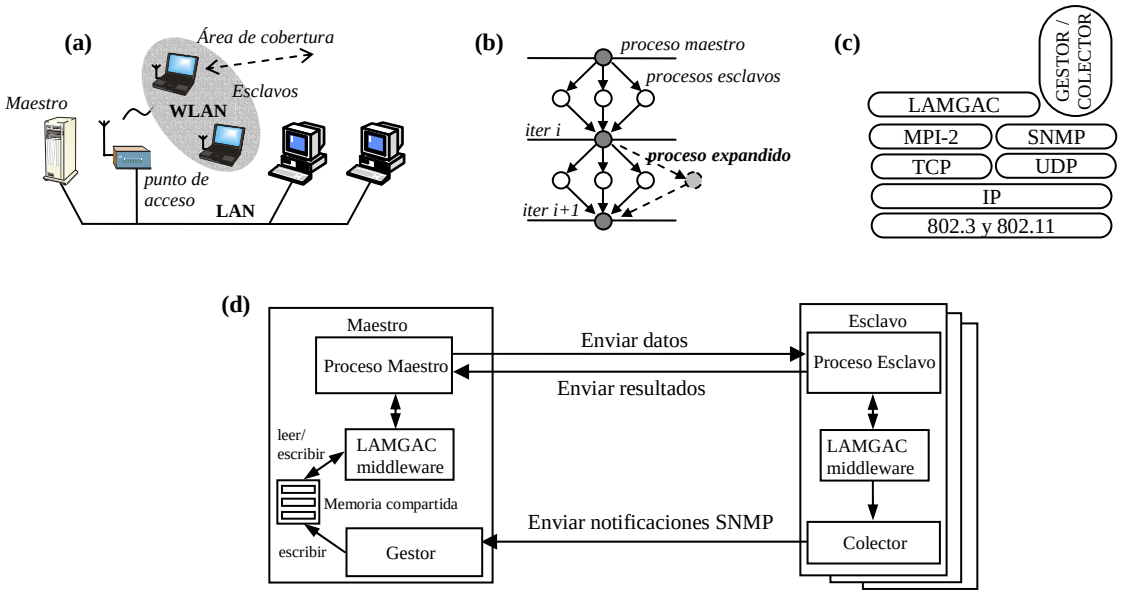


**Fig. 1.** a) Arquitectura hardware, b) Dependencias en aplicaciones Maestro/Esclavo, c) Arquitectura software, d) Interacción entre el framework SNMP y el programa paralelo

situado en una zona de cobertura reducida, la estrategia de equilibrio de carga informa, a la aplicación paralela, de que no se debe enviar ni recibir datos de dicho recurso porque puede desaparecer de forma inesperada. El recurso no participa en los cálculos mientras la calidad del enlace inalámbrico que recibe sea inferior a un cierto umbral. Como resultado, la potencia de cálculo del entorno se reduce, a pesar de que el proceso esclavo puede seguir realizando los cálculos y enviar los resultados obtenidos mientras hay conexión. El recurso permanece en una zona de cobertura reducida sin que por ello su intención sea abandonar el entorno de computación.

La nueva contribución que nosotros presentamos en este artículo es la extensión del *framework* SNMP para determinar de forma precisa los computadores de la WLAN que están accesibles, a nivel de comunicación y respecto al computador maestro, aún estando situados en una zona de cobertura reducida. De esta forma la potencia de cálculo del entorno se puede utilizar por completo mientras exista conexión con los computadores portátiles. Además, se propone un mecanismo de recepción controlada de datos que utilice la información suministrada por la arquitectura software para recibir datos sólo de aquellos computadores portátiles que estén accesibles.

El resto del artículo está organizado como sigue. En la sección 2, se describe de forma breve la arquitectura utilizada para ejecutar aplicaciones Maestro/Esclavo en un entorno LAN-WLAN, así como las principales funciones de la biblioteca diseñada. En la sección 3, se explica la implementación de la mejora introducida en la arquitectura, así como el modelo de recepción controlada que se debe utilizar. A continuación, en la sección 4, se muestran algunos resultados experimentales obtenidos. Por último, se presentan las conclusiones y el trabajo futuro.

## 2 Trabajo Previo

En la figura 1.a se muestra una combinación de la arquitectura LAN-WLAN. Por un lado, hay un computador maestro, que se encarga de distribuir la carga a los esclavos y es capaz de comunicarse con computadores situados en una LAN y en una WLAN (a través de un punto de acceso). Por otro lado, los computadores esclavos pueden pertenecer a una LAN o una WLAN. Los computadores de la LAN se consideran fijos y siempre están accesibles. Sin embargo, los computadores de la WLAN pueden cambiar su localización geográfica pudiendo entrar o salir del área de cobertura en tiempo de ejecución.

Nosotros consideramos aplicaciones paralelas en las cuales el proceso maestro distribuye, en cada iteración, una determinada cantidad de carga a cada proceso, figura 1.b. La distribución de datos la realiza el proceso maestro debido a que los procesos expandidos en tiempo de ejecución en los computadores portátiles sólo pueden comunicarse con éste. Cuando cada proceso finaliza sus cálculos, estos envían los resultados al proceso maestro. Este proceso tiene que recibir los resultados de todos los procesos esclavos antes de enviarles nuevos datos en la próxima iteración, debido a las dependencias de datos entre iteraciones en el tipo de aplicaciones consideradas.

En la figura 1.c se muestra la arquitectura software utilizada para desarrollar nuestro mecanismo de equilibrio de carga. Por un lado, utilizamos nuestro *middleware* LAMGAC [3], en su versión original,

basado en MPI-2 [10]. Este software nos permite gestionar la expansión dinámica de procesos esclavos en computadores portátiles que entran y salen de cobertura durante la ejecución de aplicaciones paralelas. La variación del número de procesos en los computadores portátiles puede ser controlada en cada iteración. Además, debido a las características heterogéneas del entorno y al tipo de aplicaciones que se ejecutan, es necesario aplicar algún mecanismo de equilibrio de carga para evitar estados ociosos en los procesadores más rápidos. Por este motivo, LAMGAC ha sido extendido en [5] y en [9] para implementar un mecanismo de equilibrio de carga efectivo en entornos de computación y comunicación heterogéneos. A continuación, en la tabla 1, se muestran las principales funciones de LAMGAC.

**Tabla 1.** Principales funciones de LAMGAC

| ***Funciones*** | ***Descripción*** |
|---|---|
| LAMGAC_Update | Actualiza el número de procesos paralelos que se ejecutan en los computadores fijos y portátiles |
| LAMGAC_Balance | Estima la cantidad de datos que tiene que ser enviada a los procesos esclavos para alcanzar el equilibrio en los tiempos de ejecución |
| LAMGAC_ItestBattery_beacon | Averigua cuáles son los procesos ubicados en los computadores portátiles que están situados en una zona de cobertura reducida o cuya batería se agotará en breve |
| LAMGAC_Store_info | Almacena la cantidad de datos procesados por cada proceso y el tiempo empleado por éstos |

Por otro lado, para llevar a cabo un equilibrio de carga eficiente se necesita información de rendimiento de los computadores. En este sentido, nosotros hemos diseñado un *framework* basado en SNMP para obtener dicha información de forma efectiva [5][9]. Debido a que la arquitectura diseñada debe consumir la cantidad mínima de recursos de cada computador, y no se necesita seguridad en las comunicaciones, se utiliza la versión SNMPv2C. A continuación, resumimos la interacción entre este *framework* y el programa paralelo, figura 1.d. En cada recurso se ejecuta un agente SNMP extendido, denominado *Colector*, que monitoriza algunos parámetros de rendimiento del computador, como son: la carga del procesador, latencia de comunicación, nivel de batería, calidad del enlace inalámbrico, etc. Cuando un evento significativo relacionado con estos parámetros ocurre (la calidad del enlace inalámbrico o nivel de batería está por debajo de un umbral, etc) el agente envía una notificación *(SNMP PDU-InformRequest)* a otro proceso SNMP, denominado *Gestor*, ubicado en el computador maestro. Este último proceso se encarga de obtener la información de rendimiento adjunta a la notificación recibida desde cada agente y, almacenarla en una zona de memoria compartida que es accedida por las funciones de la biblioteca LAMGAC. Esta información se utiliza para estimar la cantidad adecuada de carga a enviar a cada proceso cuando se invoca la función *LAMGAC_Balance()*, y para conocer los computadores con problemas de enlace inalámbrico y/o batería cuando se invoca la función *LAMGAC_ItestBattery_beacon()*.

# 3 Mecanismo de Mejora

En primer lugar, en esta sección se detalla la implementación previa a la mejora del *framework* SNMP en lo que respecta al control de los computadores portátiles. A continuación se presenta la mejora introducida para controlar de forma precisa los recursos que están en zonas de cobertura reducida, así como el esquema de recepción controlada.

## 3.1 Estado Actual

Como se explicó en la sección anterior, el proceso *Colector*, situado en los computadores esclavos, monitoriza diversos parámetros referentes al rendimiento de los computadores. En concreto, nuestro mecanismo de mejora se centra en el parámetro que refleja la calidad del enlace inalámbrico o nivel de potencia: el parámetro *lbLinkLevel* definido en la base de datos de información de gestión, LBGAC-MIB [9]. A continuación se detalla el funcionamiento del *framework* respecto a este parámetro.

El proceso *Colector* monitoriza periódicamente el parámetro *lbLinkLevel*, y envía dicho valor al proceso *Gestor* cuando se produce alguna de las siguientes situaciones:

- Comienza la ejecución de un nuevo proceso paralelo en el computador.
- Descenso continuo de la potencia de señal inalámbrica durante un intervalo de tiempo.
- La potencia de señal inalámbrica es inferior a un umbral establecido.
- Hay nivel de señal después de haber estado un periodo de tiempo fuera de cobertura.

Cada una de estas situaciones se corresponde con el envío de una notificación asíncrona, las cuales están definidas en LBGAC-MIB, y se indican en la tabla 2.

Por parte del *Gestor*, una vez que se extrae de la notificación el parámetro *lbLinkLevel*, éste se compara con un umbral fijado por el usuario. Si dicho umbral es superior al nivel de potencia indicado, el

computador se considera situado en un área de cobertura limitado, y por tanto, se declara el recurso como no accesible. Una llamada a la función *LAMGAC_ItestBattery_Beacon()* indicaría el rango del proceso que se ejecuta en dicho computador. El proceso maestro no debe considerar ese proceso en las sucesivas distribuciones de datos mientras la calidad del enlace inalámbrico que recibe el computador portátil no esté por encima del umbral.

**Tabla 2.** Notificaciones que incluyen el objeto lbLinkLevel

| *Notificaciones* | *Descripción* |
|---|---|
| lbnAppStart | Se envía cuando un nuevo proceso paralelo comienza su ejecución en el recurso |
| lbnDescLink | Se envía cuando se detecta un descenso consecutivo en la potencia de la señal |
| lbnCriticalArea | Se envía cuando el nivel de señal está por debajo de un cierto umbral |
| lbnUpLink | Se envía cuando hay nivel de señal después de haber estado fuera de cobertura |

El valor del umbral presenta un claro compromiso. Si se toma un valor alto, se reduce el área de movilidad de los recursos ya que éstos se consideran inalcanzables cuando la potencia de la señal disminuye levemente y es inferior al umbral (aunque pueda existir conexión), con el resultado de no aprovechar la potencia de cálculo del computador. Sin embargo, si la elección del umbral es un valor muy bajo puede darse el caso que el recurso quede fuera de cobertura de forma inesperada sin dar a lugar a enviar una notificación del tipo *lbnCriticalArea*, con la consecuencia pérdida de control sobre el recurso. Esta última situación puede suponer esperas indefinidas en la recepción de datos, con el consecuente bloqueo del proceso maestro esperando por datos que no van a llegar desde el proceso esclavo.

En este sentido, la mejora que se explica a continuación va en la línea de acotar el rango de incertidumbre que existe al fijar el umbral de cobertura, ya que el recurso sólo se declara inaccesible cuando se determina de forma precisa que no existe comunicación entre éste y el computador maestro.

## 3.2 Control de los Computadores Portátiles

El nuevo mecanismo que se presenta en esta sección consiste en la ampliación del *framework* SNMP. En la figura 2 se muestra un esquema de funcionamiento. Esta ampliación se basa en la creación, por parte del proceso *Gestor*, de una hebra de ejecución *(thread)* cuando el umbral de cobertura es superior al nivel de potencia de la señal inalámbrica que recibe un computador portátil (hebra *SnmpPing* en la figura 2). El proceso *Gestor* crea tantas hebras como computadores estén situados en una zona de cobertura limitada. El proceso *Gestor* conoce la calidad del enlace inalámbrico que recibe cada computador portátil a través de las notificaciones (tabla 2) enviadas por cada proceso *Colector*.

La función de las hebras de ejecución es monitorizar el estado del enlace de comunicación entre el computador maestro y los computadores portátiles mientras estos estén ubicados en un área de cobertura reducida. Para ello, cada hebra, mediante una operación *Get-Request* del protocolo SNMP, consulta la potencia de señal recibida en el computador que envió alguna de las notificaciones descritas en la tabla 2, es decir, se consulta al agente *Colector* sobre el objeto *lbLinkLevel*. Cada hebra permanece en ejecución mientras el computador portátil esté ubicado en el área de cobertura limitada. Por lo tanto, la hebra finaliza cuando el nivel de potencia de la señal inalámbrica está por encima del umbral (cobertura aceptable) o cuando no hay enlace de comunicación entre el computador maestro y el esclavo (expira el *timeout* de varias operaciones *Get-Request* consecutivas). Una vez que se den las condiciones que provocan la finalización de la hebra de ejecución, se escribe en la memoria compartida por la biblioteca LAMGAC y el proceso *Gestor* el estado del computador portátil: accesible o no. Esta información es consultada por la función de control de batería y enlace (*LAMGAC_ItestBattery_beacon()*) para informar al proceso maestro qué procesos se están ejecutando en los computadores no disponibles.

La implementación de este mecanismo de mejora modifica la definición de la información retornada por la función *LAMGAC_ItestBattery_beacon()* de la siguiente forma (en cursiva resaltamos las modificación de la definición con respecto a la mostrada en tabla 1): "Averigua cuáles son los procesos ubicados en los computadores portátiles que *no tienen conexión de red inalámbrica con el computador maestro* o cuya batería se agotará en breve".

Por otro lado, debido al comportamiento dinámico del canal de comunicaciones inalámbrico, la carga del procesador y la localización del computador portátil, el tiempo transcurrido desde que la consulta SNMP se realiza hasta que la respuesta se recibe puede variar de forma considerable de un instante a otro. Este tiempo representa un serio compromiso en la elección del valor del *timeout* de la operación de consulta. Un valor elevado provoca un tiempo de espera alto cuando existe un fallo en la comunicación (fuera de cobertura, fallo en el canal, etc) y, por lo tanto, la función de control de batería y enlace pueden indicar que existe una conexión física cuando realmente no la hay (se invocan las funciones cuando aún no ha expirado el *timeout* de la operación de

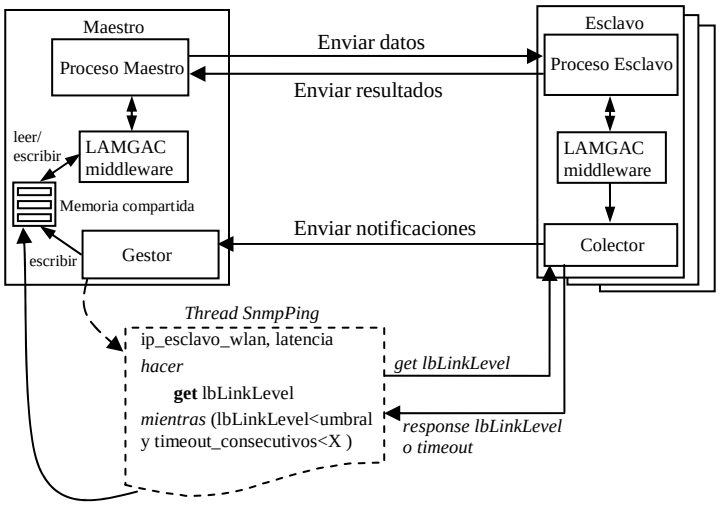


**Fig 2.** Framework SNMP modificado con hebra de ejecución SnmpPing

consulta y la información almacenada en la memoria compartida no es consistente con la situación actual). Por otro lado, si se opta por elegir un valor demasiado bajo, cualquier situación de congestión en el canal puede provocar que dicha función indique que no hay conexión cuando realmente si la hay (el *timeout* de la operación de consulta expira sin dar a lugar a la llegada de la respuesta enviada por el proceso *Colector*). Teniendo en cuenta este compromiso, el valor apropiado para el *timeout* debe ser ligeramente superior al tiempo consumido al realizar una consulta con éxito. Matemáticamente, el tiempo máximo permitido para llevar a cabo una operación de consulta al computador *i* ($c_i$) se calcula como:

$$tout(c_i) = 2 \times t_{lat}(c_i) + \frac{enviados + recibidos}{B} + t_{get}(c_i)$$

donde:

- $t_{lat}(c_i)$ es la latencia de comunicación entre el computador maestro y el computador portátil. Por simplicidad, asumimos que este valor es siempre igual en ambos sentidos de comunicación, y que su valor permanece constante. Este valor y el parámetro *B* son calculados por el agente *Colector* cuando el demonio *lamd* (distribución LAM-MPI) se inicia, y son enviados al proceso *Gestor* mediante la notificación *lbnAppStart.*
- *enviados* es el tamaño en bytes del paquete generado en la operación de consulta. En nuestro entorno de computación su valor es de 92 bytes.
- *recibidos* es el tamaño en bytes del paquete generado en la operación de respuesta a la consulta. En nuestro entorno de computación su valor es de 93 bytes.
- *B* es la velocidad de transmisión entre el computador maestro y el computador esclavo.
- $t_{get}(c_i)$ es el tiempo consumido por el agente *Colector* para decodificar la consulta, ejecutarla y devolver el resultado. Este valor depende de varios factores, como son: velocidad del procesador, tamaño de memoria, carga del sistema, etc. Por lo tanto, para calcular el valor de $tout(c_i)$, este valor se ha estimado de forma empírica para el entorno y condiciones en las que trabajamos (especificado en la tabla 3 de la sección 4). Se han realizado muchas medidas sobre el tiempo transcurrido al realizar una consulta *Get-Request* con éxito, y el valor medio obtenido es aproximadamente 0.7 ms.

Por lo tanto, los datos que tiene que comunicar el proceso *Gestor* a la hebra de ejecución para que ésta pueda realizar la consulta SNMP con el valor de *timeout* apropiado, son: la dirección IP del computador portátil y la latencia de comunicación entre este último y el computador maestro.

Por otro lado, sabiendo que SNMP utiliza el protocolo de transporte no fiable UDP y que la cantidad de bytes enviados y recibidos es pequeña, podemos asumir que el tráfico generado debido a la consulta afecta de forma mínima al rendimiento de la red [11].

## 3.3 Esquema de Recepción Controlada

Una vez iniciada la recepción de datos por parte del proceso maestro puede que algún computador portátil se sitúe fuera del área de cobertura, y por lo tanto, no

pueda enviar sus datos. Si esto ocurre, el proceso maestro espera por los datos calculados en dicho computador el tiempo que éste tarde en incorporarse al área de cobertura o indefinidamente si no regresa a la WLAN. Esta espera es perjudicial para la aplicación paralela, ya que introduce desequilibrios en los tiempos de ejecución de los procesos. Para solucionarlo, se propone un esquema de recepción controlada para la recepción de datos, basado en funciones de LAMGAC. A continuación, se explica en detalle dicho esquema, el cual se presenta en la figura 3.

```
LAMGAC_ItestBattery_beacon (…);
∀ procesos disponibles
  // Se inicia la recepción de datos no bloqueante

mientras queden operaciones de recepción por completar
  // Comprobar operaciones completadas
  ∀ nuevas operaciones de recepciones completadas
      LAMGAC_Store_info (…);

  LAMGAC_ItestBattery_beacon (…);
  ∀ nuevos procesos no disponibles
      // Cancelar la recepción de sus datos
fin mientras
```

**Fig 3.** Esquema de recepción controlada

Antes de iniciar la recepción de datos, se realiza una llamada a la función *LAMGAC_ItestBattery_beacon()* para conocer si hay algún computador que actualmente están inalcanzable. Si existen algún recurso con el que no es posible la comunicación desde el computador maestro, entonces no se implementa la operación de recepción para el proceso que se ejecuta en ese computador. Una vez iniciadas las recepciones no bloqueantes de los procesos esclavos que se desarrollan en los computadores accesibles, el proceso maestro se queda esperando por los resultados. Durante dicha espera, continuamente se comprueba si ha completado alguna de las operaciones de recepción iniciadas y si hay algún nuevo recurso con problemas de cobertura. En el caso de que alguna operación se complete, se llama a la función *LAMGAC_Store_info()* para almacenar en la memoria compartida los datos referentes al rendimiento del proceso cuya operación de recepción se acaba de completar. Estos datos son utilizados por el mecanismo de equilibrio de carga. En el caso de que existan nuevos procesos con problemas de cobertura, se cancela la recepción de éstos para no provocar esperas indefinidas por resultados que nunca van a llegar. El programador de la aplicación tiene que considerar este hecho para calcular los resultados que no pudieron ser devueltos por los procesos cancelados. Este bucle se repite continuamente hasta que no existan datos por recibir.

# 4 Resultados Experimentales

La mejora realizada en la arquitectura SNMP aporta claras ventajas en la ejecución de la aplicación paralela, ya que los procesos paralelos que se desarrollan en los computadores portátiles son considerados por la aplicación mientras exista comunicación con el computador maestro. Sin embargo, la ejecución de la hebra SnmpPing implica una sobrecarga en el entorno de computación. En esta sección se presenta cómo afecta la mejora realizada en el *framework* SNMP en el tiempo de ejecución global de la aplicación paralela.

Para ello, se han realizado varios experimentos con la versión previa de la arquitectura SNMP y la nueva mejora aportada en este artículo. El entorno de computación utilizado está formado por una red de computadores que combina segmentos de red IEEE 802.3 e IEEE 802.11. Las características de los computadores se indican en la tabla 3, los cuales utilizan la librería de paso de mensajes MPI-2 bajo el sistema operativo Linux. Cada simulación se repitió 10 veces obteniendo una desviación típica reducida. La aplicación secuencial se ejecutó en el procesador más rápido.

**Tabla 3.** Características de los recursos

| ***Procesador / Memoria*** | ***Red (Mbps)*** |
|---|---|
| PIV 2.4Ghz/512 MB (maestro) | 100 |
| PIV 2.4Ghz/512 MB | 100 |
| PIV 2.4Ghz/512 MB (portátil) | 11 |
| PIV 2.4Ghz/512 MB (portátil) | 11 |
| PIII 450/128MB | 100 |

Para llevar a cabo los experimentos, se ha utilizado como aplicación paralela una herramienta de Codiseño Hw/Sw, explicada con detalle en [5], utilizando el modelo de programación para equilibrio de carga especificado en [8]. Esta herramienta calcula la mejor combinación de recursos Hw/Sw, que cumple una serie de restricciones, para implementar un sistema de reconocimiento de voz. La especificación del sistema viene dada en el lenguaje de descripción VHDL. Esta herramienta posee dos aplicaciones: una fase de estimación y otra de particionado. Para realizar los experimentos se ha utilizado sólo la fase de estimación.

Antes de cada distribución de datos, el proceso maestro calcula, con un procedimiento recursivo, todas las combinaciones posibles de recursos Hw/Sw para implementar un proceso VHDL del sistema. Como este procedimiento no está paralelizado, el *speedup* de la aplicación se reduce considerablemente (figura 4.b). Después de calcular las combinaciones, el proceso maestro distribuye a los procesos esclavos un rango de combinaciones de recursos para estimar el costo de la implementación de cada una.

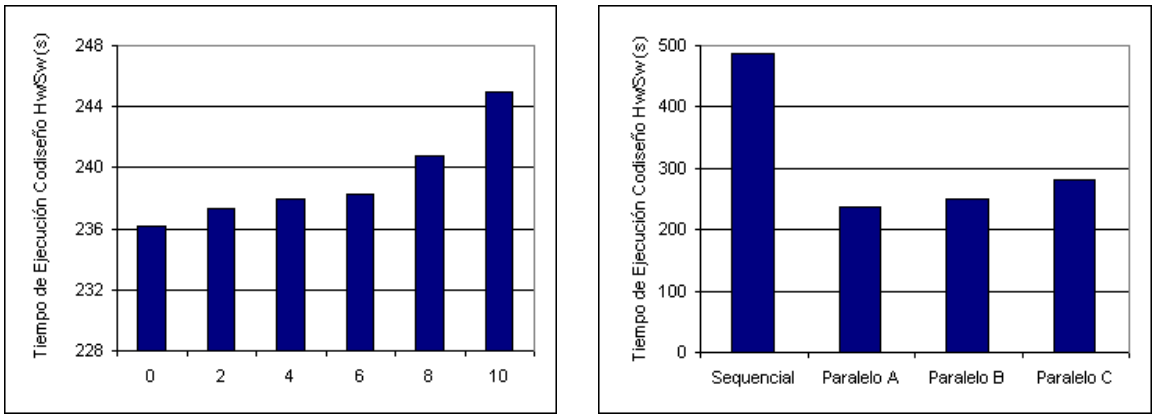


**Fig 4.** a) Sobrecarga de la hebra SnmpPing, b) Mejora del tiempo de ejecución

## 4.1 Sobrecarga de la Hebra

La ejecución de la hebra SnmpPing introduce sobrecarga en el computador maestro, en el computador portátil y en la red. Por lo tanto, es necesario estudiar la sobrecarga que introduce en el tiempo de ejecución de la aplicación paralela. La sobrecarga se debe a:

- Computador maestro: crear la hebra, construir la operación *Get-Request*, enviar la consulta al proceso *Colector* y, recibir y almacenar los datos en la memoria compartida.
- Computador portátil: decodificar la consulta SNMP, obtener la información y enviarla al proceso *Gestor*.
- Red: tráfico de paquetes UDP para las tramas de consulta y respuesta.

El primer y segundo punto depende del rendimiento y la carga actual de los computadores implicados. El tercero depende del rendimiento de la red, como puede ser: la latencia de comunicación y la tasa de transferencia. Además, también se ve afectado por las características especiales de las redes inalámbricas (alta congestión, cobertura, medio compartido, etc). Teniendo en cuenta estas consideraciones, los resultados presentados en esta sección pueden variar de forma considerable de un entorno de computación a otro.

Para evaluar la sobrecarga que introduce la hebra SnmpPing, se han realizado varios experimentos situándonos en el peor escenario posible, es decir, se ha forzado la ejecución de una hebra desde el comienzo de la aplicación paralela por cada uno de los recursos portátiles que intervienen en la aplicación. Debido al número limitado de computadores portátiles disponibles, igual a dos, la creación de un número de hebras superior a dos se realiza replicando las hebras creadas.

La figura 4.a muestra el tiempo de ejecución medio de la fase de estimación de la herramienta de Codiseño Hw/Sw en función del número de hebras ejecutadas. Como se puede apreciar, el tiempo de ejecución de la aplicación paralela sigue aproximadamente una relación lineal con el número de hebras, aumentando sobre nueve segundos cuando se ejecutan diez hebras durante toda la ejecución de la aplicación paralela. En cualquier caso, este incremento tiene una influencia mínima en el tiempo de ejecución global y, por lo tanto, la sobrecarga es despreciable. Además, en una situación real, las hebras se crean y destruyen dinámicamente si se producen las condiciones comentadas en el apartado 3.2 (sólo cuando el computador está en una zona de cobertura reducida), por lo que en la práctica es más que probable que la sobrecarga que introducen estas hebras sea inferior a la que nosotros forzamos en nuestra simulación.

## 4.2 Mejora del Tiempo de Ejecución

Para demostrar la mejora del tiempo de ejecución cuando se utiliza la aportación presentada en este artículo, se han realizado varias simulaciones donde los computadores portátiles están localizados dentro del área de cobertura durante algunas iteraciones. En concreto, la aplicación paralela tiene ocho iteraciones, y durante las cuatro primeras los computadores portátiles se ubicaron en el área de cobertura limitada.

En la figura 4.b se muestran los resultados. El experimento etiquetado como *Paralelo A* indica el tiempo medio de ejecución de la aplicación paralela cuando se utiliza la mejora aportada en este artículo. Los experimentos etiquetados como *Paralelo B* y *Paralelo C* representan el tiempo medio de ejecución cuando no se utiliza la nueva contribución y, además, hay uno y dos computadores en el área de cobertura limitada, respectivamente. Como se puede apreciar, el tiempo de ejecución global se reduce cuando se utiliza la contribución de este artículo. Con respecto al experimento *Paralelo C*, el tiempo se reduce aproximadamente en 50 segundos, lo que representa una disminución del 18%.

# 5 Conclusiones y Trabajo Futuro

En este artículo se ha presentado una mejora sobre una arquitectura SNMP desarrollada en nuestro grupo de investigación para equilibrar la carga en aplicaciones paralelas que se desarrollan en entornos heterogéneos LAN-WLAN. La mejora consiste en aprovechar la capacidad de procesamiento de los computadores portátiles mientras exista un enlace de comunicación entre el proceso maestro y éstos. Esta mejora se ve complementada con la propuesta de una recepción de datos controlada. Además, hay que añadir que esta mejora no introduce ningún cambio en la utilización de las funciones de LAMGAC definidas en trabajos previos, por lo que es transparente al usuario.

Respecto al trabajo futuro, queremos diseñar un esquema para detectar aquel computador portátil que ha desaparecido de forma inesperada del área de cobertura sin darle tiempo a notificar este hecho al proceso maestro. Además, en éste y en trabajos previos consideramos constante la latencia de red. Debido a las características de las redes WLAN, este parámetro puede variar fuertemente de un instante a otro. Por lo tanto, queremos modificar la arquitectura SNMP para estimar el valor de la latencia de forma periódica, de tal manera que afecte de forma mínima al tráfico generado por otras aplicaciones.

# Agradecimientos

Este trabajo está subvencionado por la Consejería de Educación, Cultura y Deportes del Gobierno de Canarias (PI:164/2004).

# Referencias

[1] Cheng, L. Wanchoo, A., Marsic, I., "Hybrid Cluster Computing with Mobile Objects", 4th IEEE International Conference on High Performance Computing in Asia-Pacific. Beijin, China (2000) 909-914.

[2] A. Zomaya. "Mobile Computing: Opportunities for Parallel Algorithms Research", Proceedings of the International Parallel and Distributed Processing Symposium, USA, 2002, 144-147.

[3] E. Macías, A. Suárez. "Solving Engineering Applications with LAMGAC over MPI-2", Proceedings of 9th European PVM/MPI. LNCS 2474, Springer Verlag. Linz, Austria, 2002, 130-137.

[4] D. Sánchez, E. Macías, A. Suárez. "Effective Load Balancing on a LAN-WLAN Cluster", Proceedings of Parallel and Distributed Processing, Techniques and Applications, Vol. I, CSREA, Las Vegas, USA, 2003, pp. 473-479.

[5] D. Sánchez, E. Macías, A. Suárez. "Anticipating Performance Information of Newly Portable Computers on the WLAN for Load Balancing". Proceedings of 5th Parallel Processing and Applied Mathematics, LNCS 3019. Springer-Verlag, Czestochowa, Poland, 2003, 946-953.

[6] R. Busby, M. Nielsen, D. Andresen, "Enhancing NWS for use in an SNMP Managed Internetwork", Proceedings of International Parallel and Distributed Processing Symposium, Cancún, Mexico, 2000, 506-511.

[7] M. Subramanian, *Network management: principles and practice*, Addison-Wesley, USA, 2000.

[8] D. Sánchez, E. Macías, A. Suárez. "A library for Load Balancing in master/Slave Applicattions on a LAN-WLAN Environment". Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network-based Processing. A Coruña, España, 2004, 168-175.

[9] D. Sánchez, E. Macías, A. Suárez. "Load balancing Detecting Battery Energy Level and Wirelles Beacon Strength". Proceedings of the 15th IASTED International Conference on Parallel and Distributed Computing and Systems. Marina del Rey, USA, 2003, 268-273.

[10] William Gropp, Ewing Lusk, Rajeev Thakur, *Using MPI-2: advanced features of the message-passing interface*, Cambridge, Mass. MIT Press, 1999.

[11] M.G. Arranz, R. Agüero, L. Muñoz, P. Mähönen. "Behaviour of UDP-Based Applicationes over IEEE 802.11 Wireless Networks". Proceedings of 12th IEEE International Symposium on Personal Indoor and Mobile Radio Communication. San Diego, USA, 2001, vol II 72-77.