Universidad Politécnica de Valencia

Escuela Técnica Superior de Informática

Proyecto Fin de Carrera

Estudio de la influencia de la red de interconexión en la ejecución de aplicaciones paralelas

Autora: Carmen Campos González
Tutor: Dr. Federico Silla Jiménez

Noviembre de 2010

Índice general

1. Introducción **5**

2. Equipo de pruebas **7**
2.1. Computadores 7
2.2. Interconexión 7

3. Pruebas realizadas **9**
3.1. Caracterización de las redes 9
3.2. Benchmarks MPI de Intel 9
3.2.1. Benchmars de comunicación simple 10
3.2.2. Benchmarks de comunicación paralela 11
3.2.3. Comunicación colectiva 12
3.3. Aplicaciones 13
3.3.1. Gromacs 13
3.3.2. Lammps 13
3.3.3. NPB 13
3.3.4. Multiplicación matricial 14

4. Resultados **15**
4.1. Ancho de banda de las redes 15
4.2. Benchmarks MPI de Intel 16
4.2.1. PingPong 16
4.2.2. PingPing 18
4.2.3. Multi-PingPong 20
4.2.4. Multi-PingPing 26
4.2.5. Sendrecv 34
4.2.6. Conclusiones sobre los resultados de comunicación paralela 39
4.2.7. Gather 39
4.2.8. Scatter 42
4.2.9. Bcast 45
4.2.10. Reduce 51
4.2.11. ReduceScatter 53

4.2.12. Alltoall 56
4.2.13. Allgather 60
4.2.14. Conclusiones a los resultados de las pruebas con benchmarks MPI de Intel 62
4.3. Aplicaciones 63
4.3.1. Gromacs 65
4.3.2. NPB 73
4.3.3. Multiplicación de matrices 77

5. Conclusiones **89**

Capítulo 1

Introducción

El primer objetivo que se ha perseguido al diseñar este proyecto fin de carrera está relacionado con la formación del alumno. Se pretende proporcionar a éste un primer contacto con las tareas de investigación, iniciando su formación en este campo. Por otro lado, se busca reforzar el dominio del alumno en el manejo de equipos informáticos y dispositivos de red, complementando así las destrezas adquiridas en la carrera.

Para alcanzar este doble objetivo se ha decidido escoger como tema de estudio las redes de interconexión en la computación paralela. Más concretamente, se va a analizar la influencia de las mismas en el tiempo de ejecución de varias aplicaciones paralelas.

La computación paralela surge debido a las limitaciones de capacidad de cálculo que presentan los sistemas monoprocesador. En este sentido, la resolución de problemas que necesitan gran potencia de cómputo se viene haciendo, desde hace tiempo, con la ayuda de sistemas multiprocesadores, en los que varios elementos trabajan en paralelo. Mediante la computación paralela es posible reducir el tiempo total de ejecución, repartiendo la carga de trabajo entre distintos procesadores, que realizarán de forma simultánea su parte del trabajo. Basicamente hay dos formas de programar las aplicaciones que serán ejecutadas en estos sistemas multiprocesador. La primera es haciendo uso del paradigma de memoria compartida, y la segunda es mediante el paso de mensajes. En la primera, los diferentes procesos o threads que componen la aplicación paralela son un mapa de memoria común y la comunicación se realiza de forma implícita accediendo a variables compartidas. En cambio, cuando se usa el paradigma de paso de mensajes, los procesos que se comunican lo hacen de forma explícita intercambiando mensajes que contienen los datos que quieren comunicar. El ejemplo más conocido de este paradigma es el MPI (Message Passing Interface). Muchos de los sistemas multiprocesadores tienen, a menudo, un coste muy elevado, por lo que cada vez es más habitual utilizar sistemas multiprocesadores formados por clusters de PCs interconectados mediante redes, ya sean redes de altas prestaciones

o redes de área local estándar. En este caso, lo habitual es usar el paradigma de paso de mensajes para comunicar los diferentes procesos.

En este contexto de interés general por la computación paralela, en el presente proyecto fin de carrera se ha decidido realizar un estudio sobre las prestaciones que podría proporcionar un sistema multiprocesador formado con elementos de uso común y al alcance de cualquiera. En concreto, nuestro sistema estará formado por 16 PCs de sobremesa convencionales, conectados en red. Para la red de interconexión, se han considerado diferentes opciones, todas ellas de uso habitual (o incluso ya en desuso), bajo coste económico y accesibles para cualquiera. Así, las redes escogidas han sido:

1. Red ethernet interconectada mediante un switch de $100Mbps$.
2. Red ethernet interconectada mediante un switch de $10Mbps$.
3. Red ethernet interconectada mediante un hub de $10Mbps$.
4. Red inalámbrica con router wifi de $54Mbps$.

Se pretende estudiar el modo en que las características de las redes utilizadas para la interconexión de PCs pueden influir en el rendimiento que se obtiene al ejecutar sobre ellos aplicaciones paralelas que utilizan paso de mensajes. Se evaluará el comportamiento que presentan las diferentes redes y si éstas son o no adecuadas para la computación paralela. Para ello, utilizando las redes descritas, se ejecutará una serie de aplicaciones paralelas, implementadas sobre MPI. La medida de los tiempos de ejecución de dichas aplicaciones, sobre cada una de las redes, servirá para comparar el comportamiento que presentan éstas cuando se utilizan para computación paralela.

Como paso previo a la ejecución de aplicaciones reales, se utilizarán los benchmarks IMB de Intel para realizar un estudio del rendimiento que presenta cada una de las redes en la ejecución de las distintas funciones MPI. Este estudio dará una primera aproximación del comportamiento de las redes disponibles al ejecutar aplicaciones basadas en MPI. Cabe destacar que en este proyecto no se está proponiendo el uso de dichas redes para interconectar los nodos de un cluster. Para este fin, las tecnolgías actuales, como 1Gb Ethernet, 10 Gb Ethernet o Infiniband, son mucho más apropiados. En este proyecto se hace uso de las redes mencionadas anteriormente porque son las disponibles en los laboratorios docentes del departamento, y porque a pesar de sus bajas prestaciones, permiten introducir al proyectando en las labores de investigación, así como afianzar y ampliar sus conocimientos sobre redes. Estos dos objetivos son la meta de este proyecto.

Capítulo 2

Equipo de pruebas

La parte experimental de este estudio se ha realizado utilizando las instalaciones del laboratorio docente de redes de la escuela de informática. Se han empleado en este estudio 16 PCs. Si bien el laboratorio cuenta con más equipos, dado que el número de los mismos no alcanza los 32, se ha optado por emplear tan sólo los 16 citados, ya que de este modo nos permite mejorar la distribución del trabajo entre los equipos.

2.1. Computadores

Los PCs utilizados tienen las siguientes características:

- Placa base *dmidecode*.
- Procesador, *AMD Athlon* 64x2 4800
- Memoria, 4GB
- Sistema operativo, *Kubuntu 8.04*
- Tarjetas de red:
  - ethernet 100/1000Mbps (eth0), modelo *NVIDIAnforce*.
  - ethernet 10/100Mbps (eth1), modelo *3com 3c905c*.
  - wifi (wlan1), *Conceptronics C54RU*.

2.2. Interconexión

Los dispositivos de interconexión empleados son:

- Para la interfaz eth0 se utiliza un switch de $100Mbps$, marca *3COM*, modelo *3300 XM*.

  Este switch se utiliza para conectar la red interna del laboratorio, a *internet*.

- Para la interfaz eth1 se utiliza: un switch de 10*Mbps*, marca *3COM*, modelo 610, o bien, un hub de 10*Mbps* marca *3COM*, modelo *PS Hub 40*.
- Para la interfaz wlan1 se utiliza un router wifi de 54*Mbps*, marca *ASUS*, modelo *WL500G premium*.

Los 16 PCs comparten un directorio en el que está instalado *MPI* y las aplicaciones utilizadas. Este directorio es exportado por *NFS* desde un servidor al que se accede a través del switch de 100*Mbps*.

Capítulo 3

Pruebas realizadas

En este capítulo se describen las pruebas que se han ejecutado en el presente proyecto, con especial atención a las características de las distintas aplicaciones y benchmarks empleados.

3.1. Caracterización de las redes

Con el fin de conocer las características de ancho de banda real en las redes que se utilizan, se plantean pruebas sencillas en *java* con las que se miden tiempos de envío de ficheros y ancho de banda. En concreto, para cada una de las redes, se realizan 10 envíos de ficheros de diferentes tamaños (0, 1, 1K, 1M, 10M y 100M bytes) y se mide, para cada uno de ellos, el tiempo empleado en la transmisión. Dividiendo el tamaño del mensaje entre el tiempo obtenido, se calcula el ancho de banda correspondiente.

3.2. Benchmarks MPI de Intel (IMB)

Estos benchmarks son un conjunto de kernels elementales que permiten evaluar cualquier función de *MPI*. Serán utilizados para obtener una comparación de bajo nivel entre las redes.

Se ejecutaron los siguientes benchmarks:

- PingPong
- PingPing
- Sendrecv
- Exchange
- Allreduce
- Reduce

- Reduce_scatter
- Allgather
- Allgatherv
- Gather
- Gatherv+
- Scatter
- Scatterv
- Alltoall
- Alltoallv
- Bcast
- Barrier
- Versiones *multi* de PingPong y PingPing

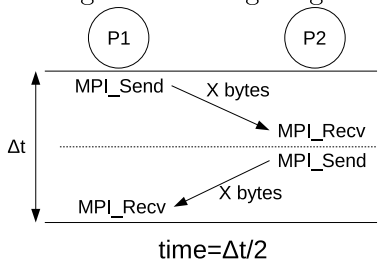
Para el presente trabajo se han seleccionado los que se describen a continuación por considerarlos más representativos.

3.2.1. Benchmars de comunicación simple

Estos benchmarks evalúan el rendimiento en comunicaciones punto a punto entre dos únicos procesos activos.

PingPong Se utiliza para medir el rendimiento en comunicaciones unidireccionales entre dos nodos. En este test (ver figura 3.1) un proceso envía un único mensaje de longitud fija a otro, el cual a su vez, lo recibe y lo devuelve inmediatamente al primer proceso. El tamaño del mensaje varía dentro de un amplio intervalo. Los datos que proporciona, los obtienen dividiendo por dos la media del tiempo de un elevado número de viajes de ida y vuelta.

Figura 3.1: PingPong



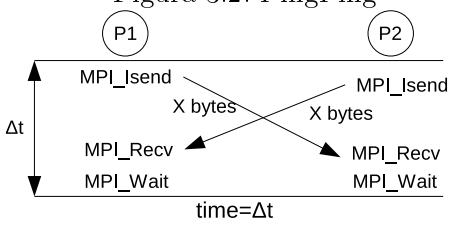
Tal como se expresa en la figura 3.1, en un tiempo igual a $time = \frac{\Delta t}{2}$, se realiza un envío de x bytes.

Con *PingPong* el tiempo de ida y vuelta medido se divide por dos para calcular la latencia en un sentido. Dividiendo el tamaño del mensaje entre el tiempo proporcionado por este test, se obtiene el ancho de banda unidireccional.

Los valores de ancho de banda proporcionados por este benchmark están expresados en $Mbytes/seg$ ($1Mbyte = 2^{20}bytes$). Para pasar de $Mbytes/seg$ a $Mbps$ basta con multiplicar el valor obtenido por 8 (para pasar a bits), y por 1,048576 (para considerar $1M = 10^6$).

PingPing Se utiliza para medir el rendimiento en comunicaciones bidireccionales entre dos nodos. En este test (ver figura 3.2), dos procesos envían y reciben mensajes simultaneamente a y desde los procesos complementarios. El envío y la recepción se hacen secuencialmente, cada envío es inmediatamente seguido de una recepción, y viceversa. El tamaño del mensaje varía dentro de un amplio intervalo.

Figura 3.2: PingPing



Tal como se expresa en la figura 3.2, en un tiempo igual a $time = \Delta t$, se realiza un envío de x bytes.

El dato de ancho de banda que proporciona este benchmark sólo tiene en cuenta el envío de x bytes (no la recepción de otros x bytes)

3.2.2. Benchmarks de comunicación paralela

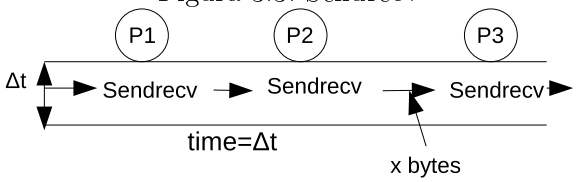
En el caso de este grupo de benchmarks, se llevan a cabo, de forma paralela, distintas comunicaciones punto a punto entre parejas de nodos.

MultiPingPong En este benchmark se ejecuta *IMB-pingpong* en varias parejas de nodos simultaneamente. Los tiempos y ancho de banda que proporciona se corresponden con los peores de entre todas las parejas.

MultiPingPing Similar al anterior. En este caso cada pareja de nodos implicada en la comunicación ejecuta el benchmark *pingping*.

Sendrecv En este test (ver figura 3.3) los procesos forman una cadena periódica de modo que cada uno envía simultaneamente mensajes al próximo y recibe del anterior. Es útil para evaluar el ancho de banda bidireccional entre hosts bajo carga global. Los valores de latencia se obtienen del tiempo que necesitan todos los nodos para enviar un mensaje al siguiente. El ancho de banda se obtiene dividiendo el tamaño de mensaje entre la latencia que mide.

Figura 3.3: Sendrecv



3.2.3. Comunicación colectiva

En este grupo se incluyen todos los benchmarks que se corresponden con funciones del estándar *MPI* de comunicación colectiva. Miden el rendimiento del sistema en su conjunto. Evalúan no sólo la capacidad de paso de mensajes del sistema, sino también la calidad de la implementación de MPI utilizada.

Gather Benchmark para la función *MPI_Gather*. Cada proceso envía *xbytes* a un nodo maestro el cual reune todos los datos recibidos.

Scatter Benchmark para la función *MPI_Scatter*. Un nodo maestro reparte un vector de bytes entre el resto de nodos, enviando *xbytes* a cada uno de ellos.

Bcast Benchmark para la función *MPI_Bcast*. Un nodo maestro realiza una operación de *broadcast* para un conjunto de nodos, enviando un mismo mensaje de *xbytes* a cada uno de los nodos del conjunto.

Reduce Benchmark para la función *MPI_Reduce*. Un nodo maestro combina los datos proporcionados por un conjunto de nodos, utilizando una determinada operación (en este caso *MPI_Sum*), y obtiene un resultado final para dicha operación.

Reduce-scatter Benchmark para la función *MPI_Reduce_scatter*. En la que se lleva a cabo una operación *reduce* seguida de otra *scatter*.

Alltoall Benchmark para la función *MPI_Alltoall*. Es una operación colectiva en la que todos los procesos envían la misma cantidad de datos a todos los demás, y recibe la misma cantidad de datos de todos los demás.

Allgather Benchmark para la función *MPI_ Allgather*. El nodo maestro reúne datos de todos los nodos y envía el resultado al resto de nodos.

3.3. Aplicaciones

En esta sección se describen las cuatro aplicaciones que se han empleado en presente estudio para complementar los resultados obtenidos con la ejecución de los benchmarks descritos en la sección anterior.

3.3.1. Gromacs

GROMACS (GROningen MAchine for Chemical Simulations) es un paquete de simulación de dinámica molecular desarrollado originalmente en la universidad de Groningen y que se distribuye bajo licencia GPL. Esta aplicación se utiliza para simular las ecuaciones Newtonianas de movimiento para sistemas con cientos o millones de partículas. Se ejecuta distribuyendo los átomos del sistema simulado entre todos los nodos activos. El tiempo se divide en muchos pasos, cada uno de ellos de una pequeña fracción de picosegundo de tiempo de simulación.

Gromacs es muy sensible a la latencia y no sólo al ancho de banda, ya que debe existir sincronización entre todos los nodos antes de continuar al próximo paso.

Esta aplicación proporciona unos sistemas moleculares de ejemplo que sirven para medir el rendimiento de Gromacs en un sistema. En concreto, para este estudio utilizaremos los sistemas DPPC y POLY.

3.3.2. Lammps

LAMMPS ("Large-scale Atomic/Molecular Massively Parallel Simulator"). Es un programa de dinámica molecular mantenido y distribuido, bajo licencia GPL, por los laboratorios Sandia. Utiliza técnicas de descomposición espacial para dividir el dominio de simulación en pequeños subdominios de tres dimensiones, asignando cada uno de ellos a un procesador. Los procesadores comunican y almacenan información sobre los átomos vecinos a su subdominio.

Lammps proporciona unos ejemplos de muestra, de tamaños variables, para ser ejecutados con el programa. En nuestro caso hemos optado por el sistema ejemplo denominado *crack*, que es uno de los de pequeño tamaño.

3.3.3. NPB

NPB (NAS Parallel Benchmarks) forman un conjunto de benchmarks desarrollados por la NASA, que han sido diseñados, a partir de aplicaciones de cómputo de dinámica de fluidos, para la evaluación del rendimiento de

sistemas y redes de computación paralela. Estos benchmarks, son proporcionados en forma de kernels, que aunque no son aplicaciones paralelas reales, forman parte de un gran número de ellas. De los 5 kernels disponibles se han seleccionado dos (EP y FT):

EP (Embarrassingly parallel): Se realiza una simulación de problemas que requieren poca o ninguna comunicación entre procesos, y proporciona una estimación del rendimiento máximo alcanzable, ya que no existiría interferencia por la comunicación entre procesos. En concreto, genera pares de números pseudoaleatorios siguiendo una distribución normal, de modo que el trabajo se reparte entre los nodos sin que éstos precisen comunicación alguna hasta la puesta en común final.

FT (Fourier Transformation): kernel que resuelve ecuaciones diferenciales utilizando *FFTs* (Fast Fourier Transformation). Ha sido seleccionado porque tanto *Lammps* como *Gromacs* hacen uso de la librería *fftw* para el cálculo de la *Transformada de Fourier*.

3.3.4. Multiplicación matricial

Se utiliza una implementación sobre *MPI* de un algoritmo paralelo para la multiplicación de matrices. Las matrices a multiplicar se consideran distribuidas por bloques entre los nodos procesadores. Cada nodo consigue completar el cálculo del bloque correspondiente de la matriz producto, mediante el intercambio secuencial de bloques entre los nodos de su misma fila y columna.

Capítulo 4

Resultados

En este capítulo se han reunido los resultados obtenidos de las experiencias ejecutadas, expresados en forma de gráficas y tablas. También se incluye un pequeño análisis de los resultados que se han obtenido en dada uno de los pasos que se han dado, con el fin de decidir qué nuevas pruebas nos van a ser más útiles para completar el estudio que se pretende culminar.

4.1. Ancho de banda de las redes

En la figuras 4.1 y 4.2 se comparan los anchos de banda, medidos mediante el envío de ficheros en java, tal como es describe en el apartado 3.1 y obtenidos para cada una de las redes que se emplean en el estudio. En ambas gráficas se muestran idénticos parámetros, la única diferencia entre ambas es que en la segunda se amplía la parte baja de la escala de anchos de banda (ordenadas), con el fin de apreciar las diferencias entre las redes cuyo comportamiento es similar. Es decir, la gráfica 4.2 es un zoom de 4.1.

Figura 4.1: Ancho banda (*Mbps*)

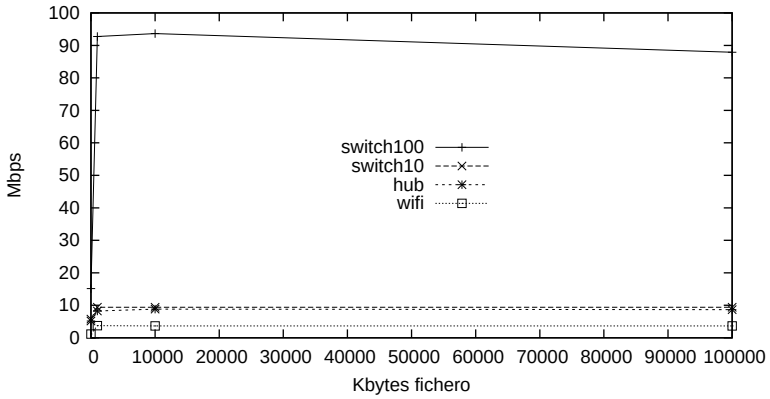
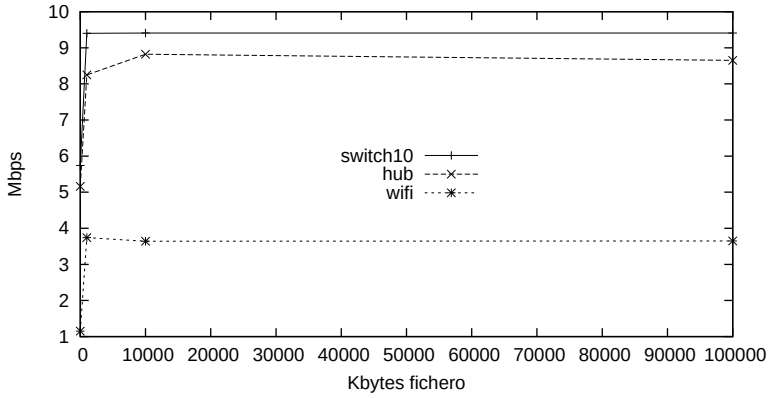


Figura 4.2: Ancho banda (*Mbps*)



En la tabla 4.1 se muestran, a modo de ejemplo, las medidas de ancho de banda que se obtienen en el envío de ficheros de $10 Mbytes$.

Tabla 4.1: Medida de ancho de banda real para ficheros de $10 Mbytes$

red	**switch** $100 Mbps$	**switch** $10 Mbps$	**hub** $10 Mbps$	**wifi** $54 Mbps$
Mbps reales	93,90	9,41	8,82	3,64

En los valores mostrados en la tabla 4.1, se observa que el ancho de banda real que presentan las redes utilizadas es algo menor que el nominal en las redes ethernet, mientras que es considerablemente menor al nominal en la red inalámbrica (6.7 % del ancho de banda nominal).

4.2. Benchmarks MPI de Intel

En esta sección se exponen los resultados obtenidos con los benchmarks utilizados, y que se han descrito en el apartado 3.2.

4.2.1. PingPong

En las figuras 4.3 y 4.4, donde la segunda es un zoom de la primera, se representan las medidas de ancho de banda proporcionadas por el benchmark *IMB-PingPong*.

Figura 4.3: *PingPong* (*Mbytes/seg*)

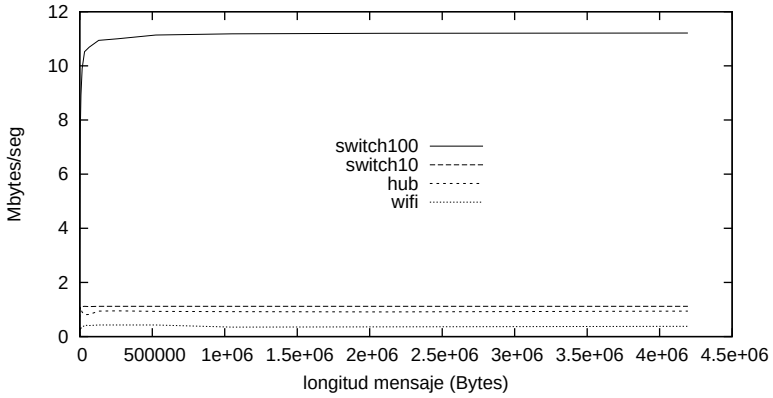
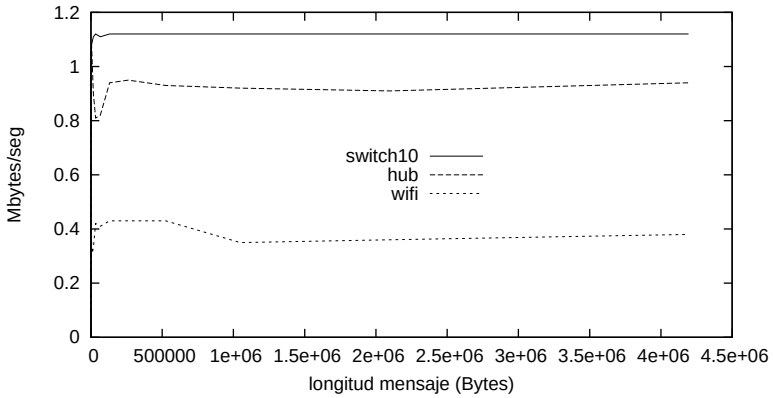


Figura 4.4: *PingPong* (*Mbytes/seg*)



La tabla 4.2 muestra las medidas de ancho de banda para envíos de $4Mbytes$, que se obtienen utilizando *IMB-PingPong*:

Tabla 4.2: Medida de ancho de banda dada por *IMB-PingPong* (para $4Mbytes$)

red	**switch** $100Mbps$	**switch** $10Mbps$	**hub** $10Mbps$	**wifi** $54Mbps$
Mbytes/sec	11,20	1,12	0,94	0,38
Mbps	93,95	9,39	7,88	3,19

Comprobamos que los resultados son similares a los obtenidos en la sección 4.1, aunque para alguna de las redes se observa que existe pérdida en el ancho de banda debido a la sobrecarga de *MPI*. En la tabla 4.3 se muestra la evaluación de esta pérdida de ancho de banda.

Tabla 4.3: Pérdida de ancho de banda por sobrecarga *MPI*

red	**switch** $100Mbps$	**switch** $10Mbps$	**hub** $10Mbps$	**wifi** $54Mbps$
diferencia*Mbps*	$-0{,}05$	0,02	0,93	0,45
% pérdida	0 %	0,2 %	10,5 %	12,3 %

Por un lado se observa que la sobrecarga que representa *MPI* es despreciable en las redes conectadas mediante un switch. Por otro lado, la pérdida de ancho de banda no depende únicamente del propio ancho de banda de la red. Por ejemplo, las redes conectadas con un switch de $10Mbps$ y con un hub de $10Mbps$ tienen un ancho de banda en comunicaciones punto a punto muy similar, y sin embargo la pérdida que presenta la segunda es significativamente mayor que la primera. Como veremos más adelante, esto es debido a que las redes conectadas mediante un switch tienen una importante ganancia de ancho de banda cuando se realizan múltiples comunicaciones simultáneas (necesarias para la sincronización con *MPI*). De forma equivalente puede decirse que la red conectada mediante un hub pierde ancho de banda ante comunicaciones paralelas (la red wifi tendrá un comportamiento similar a la red conectada mediante un hub).

4.2.2. PingPing

En las figuras 4.5 y 4.6 (la segunda es un zoom de la primera) se muestran los valores de ancho de banda bidireccional proporcionados por *IMB-PingPing*.

Figura 4.5: *PingPing* (Mbytes/seg)

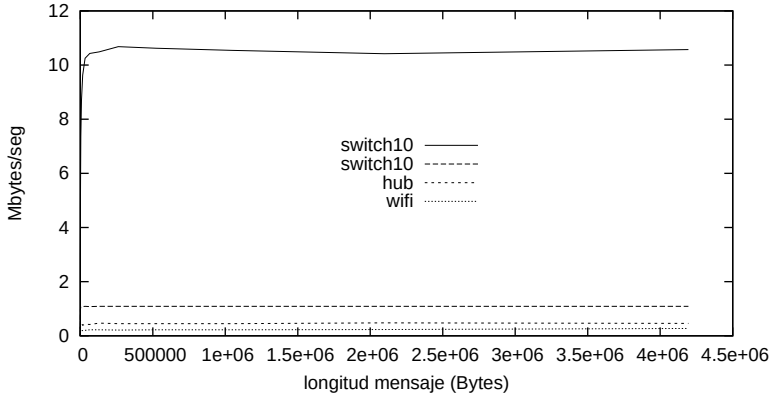
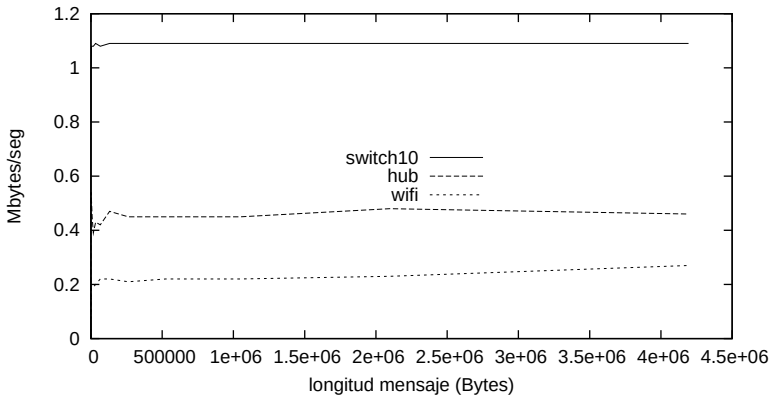


Figura 4.6: *PingPing* (Mbytes/seg)



Para ficheros de $4Mbytes$ se obtienen las medidas para *IMB-PingPong* e *IMB-PingPing* presentadas en la tabla 4.4:

Tabla 4.4: Comparación *IMB-PingPong* e *IMB-PingPing* ($Mbytes/sec$)

(**Mbps**)	**switch** $100Mbps$	**switch** $10Mbps$	**hub** $10Mbps$	**wifi** $54Mbps$
PingPong	11,20	1,12	0,94	0,38
PingPing	10,57	1,09	0,46	0,27
% pérdida	5,6 %	2,7 %	51,06 %	28,95 %

Se observa que en las redes que utilizan un switch para la interconexión, apenas disminuyen el ancho de banda, mientras que la red que utiliza un hub reduce el ancho de banda a la mitad. Esto es debido a que los switches presentan comunicación fullduplex y el hub no. Respecto a la red inalámbrica, reduce su ancho de banda entre un tercio y un cuarto aproximadamente.

Hay que tener en cuenta que los datos de ancho de banda que proporciona este benchmark sólo consideran uno de los envíos que hace la pareja de nodos, y representa el ancho de banda que localmente detectaría uno de los nodos en su envío. Si se considerasen los dos envíos que realiza la pareja de nodos simultaneamente, se observaría una ganancia de ancho de banda en las redes conectadas con un switch.

Pueden hacerse dos lecturas del mismo hecho: una es que, al realizar dos envíos simultáneos las redes con switch mantienen su ancho de banda (local), mientras que la red con hub reduce su ancho de banda a la mitad. La otra lectura sería que al realizar los dos envíos simultáneos, las redes con switch presentan una ganancia en su ancho de banda (global), que se duplica. Por el contrario, en la red conectada con un hub, el ancho de banda (global) se mantiene constante.

4.2.3. Multi-PingPong

Switch-100Mbps: Las gráficas 4.7 y 4.8 muestran que las curvas resultantes de variar el número de nodos han quedado superpuestas, lo cual implica que el número de parejas de nodos que efectuan simultaneamente comunicaciones punto a punto no influye en el tiempo de envío de mensajes, ni en el ancho de banda en cada nodo. Este comportamiento se debe al hecho de que una red conectada mediante un switch permite distintos canales de comunicación simultánea entre pares. De esta forma, el ancho de banda teórico puede verse multiplicado por el número de comunicaciones que se dan de forma paralela en un determinado momento.

Figura 4.7: *Multi-PingPong* Switch-100*Mbps*

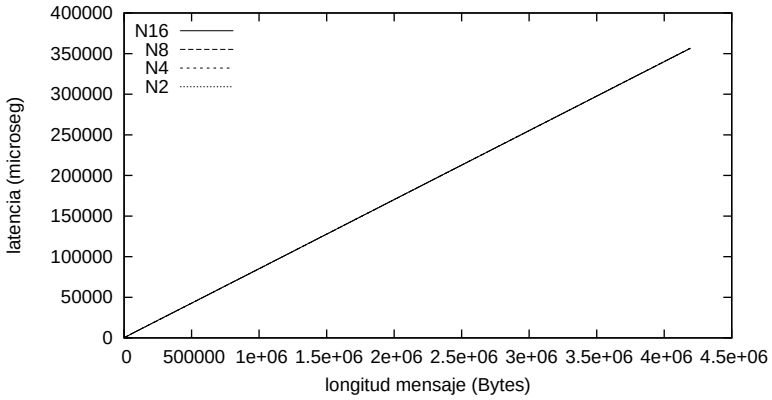
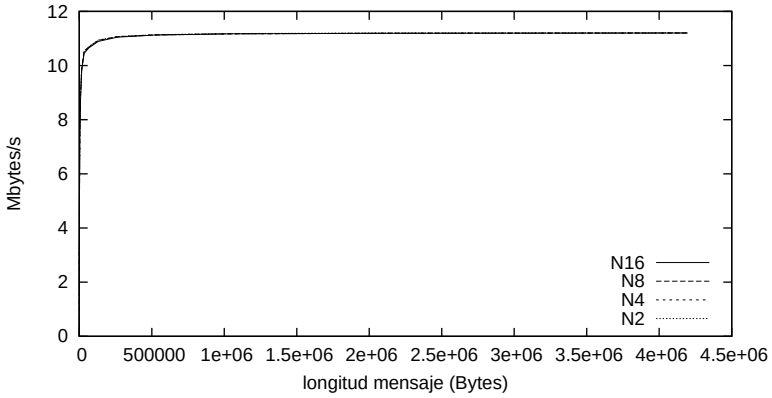


Figura 4.8: *Multi-PingPong* Switch-100*Mbps*



Veamos con un ejemplo la ganancia de ancho de banda que se tiene cuando efectuamos n comunicaciones de forma paralela. Calcularemos el ancho de banda del sistema cuando $2n$ nodos ejecutan simultaneamente IMB-PingPong, para un envío de $x = 4Mbytes$:

Independientemente del valor de n, el tiempo obtenido al ejecutar *IMB-MultiPingPong* es aproximadamente $t = 356600\mu s$. Así, por cada pareja, en $t = 356600\mu s$ se envían $x = 4Mbytes$. Por tanto el *ancho de banda global* (ABG) del sistema viene dado por la expresión:

$$ABG = \frac{n * 4 * 1048576Bytes}{356600\mu s} = \frac{n * 4 * 8 * 1048576bits}{356600 * 10^{-6}s} = n * 94Mbps$$

Tabla 4.5: *ABG-MultiPingPong* con Switch-100$Mbps$

nº nodos	**2**	**4**	**8**	**16**
n	1	2	4	8
ABG(Mbps)	94	188	376	752

La tabla 4.5 muestra cómo el *ancho de banda global* crece de forma proporcional al número de comunicaciones punto a punto simultáneas entre pares de nodos.

Nótese que no ha habido pérdida de generalidad al efectuar los cálculos para un determinado tamaño de envío ($x = 4Mbytes$). Tal como muestra la gráfica 4.7, el tiempo crece de forma lineal con el tamaño de envío, por lo que los resultados obtenidos habrían sido los mismos si consideramos otro tamaño de envío, con su latencia correspondiente.

Switch-10Mbps: Las gráficas 4.9 y 4.10 muestran que la red conectada mediante un switch de $10Mbps$ presenta un comportamiento similar al de la red conectada mediante un switch de $100Mbps$. Independientemente del valor de n (número de comunicaciones punto a punto simultáneas) los tiempos obtenidos al ejecutar *IMB-PingPong* en cada uno de los pares es aproximadamente el mismo para todos ellos.

Figura 4.9: *Multi-PingPong* Switch-10$Mbps$

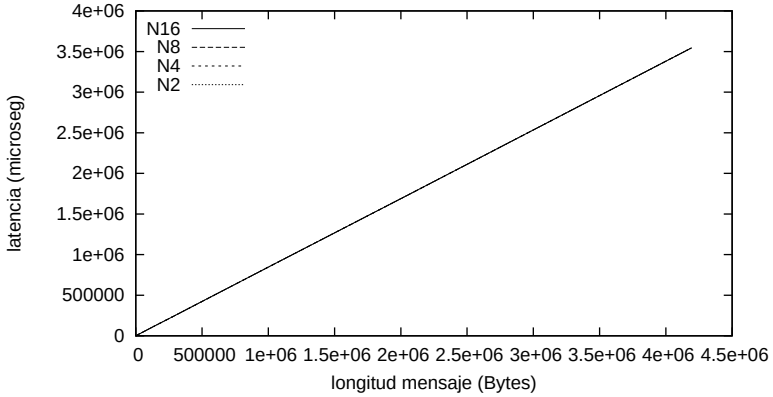
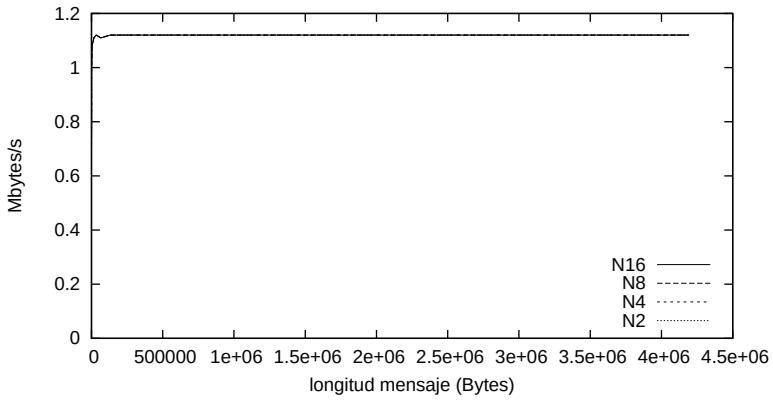


Figura 4.10: *Multi-PingPong* Switch-10$Mbps$



De forma similar al ejemplo anterior, para estimar la ganancia de ancho de banda que se produce al tener comunicaciones simultáneas, calculamos el *ancho de banda global* del sistema (ABG). Para ello consideramos, sin pérdida de generalidad, un tamaño de envío $x = 4Mbytes$. La latencia para este valor de x es, aproximadamente, $t = 3545530\mu s$ independientemente del número de nodos implicados en la comunicación. El *ancho de banda global* vendrá dado por la expresión:

$$ABG = \frac{n * 4Mbytes}{3545530\mu s} = \frac{n * 4 * 8 * 1048576bits}{3545530 * 10^{-6}s} = n * 9{,}4Mbps$$

Tabla 4.6: ABG-MultiPingPong con Switch-10$Mbps$

nº nodos	**2**	**4**	**8**	**16**
n	1	2	4	8
ABG(Mbps)	9,4	18,8	37,6	75,2

A la vista de la tabla 4.6, se comprueba, de nuevo, que cuando para la interconexión se utiliza un switch, el hecho de que haya comunicaciones simultáneas hace que aumente el *ancho de banda global*.

Se observa que los valores de ancho de banda obtenidos para la red conectada mediante el switch de 100$Mbps$ son 10 veces los obtenidos para la red conectada mediante el switch de 10$Mbps$.

Hub: Con las figuras 4.11 y 4.12 se puede comprobar que el comportamiento que presenta la red conectada mediante un hub es diferente al de las redes conectadas mediante un switch. En el caso del hub, se observa que el tiempo de envío y el ancho de banda obtenidos varían con el número de

nodos: al duplicar el número de nodos se duplica el tiempo de envío y el ancho de banda (local) se reduce a la mitad.

Figura 4.11: *Multi-PingPong* Hub

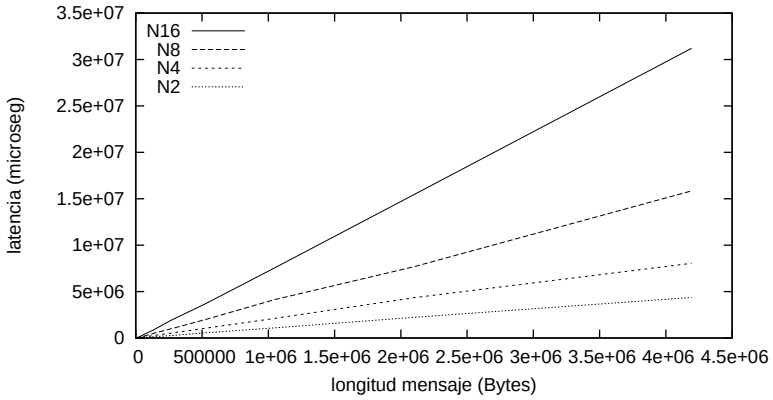
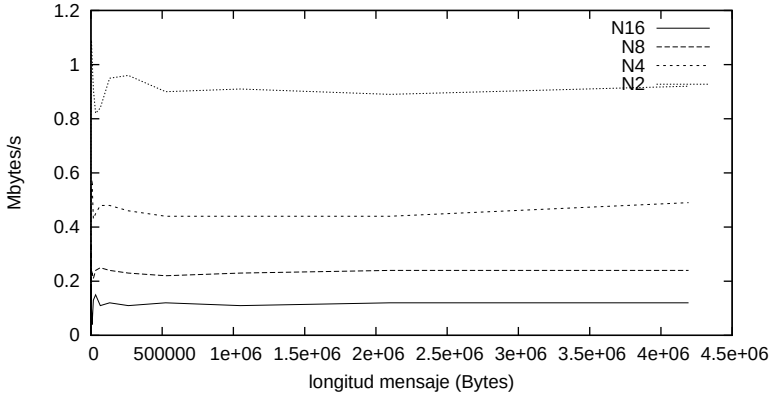


Figura 4.12: *Multi-PingPong* Hub



Calcularemos el ancho de banda del sistema cuando $2n$ nodos ejecutan simultaneamente *IMB-PingPong* y comprobaremos que en este caso no se produce la ganancia de los casos anteriores.

Como puede observarse en la gráfica 4.11, para un envío de $x = 4Mbytes$, la latencia (t_n), depende del número de comunicaciones simultáneas (n) que hay en un momento dado. El *ancho de banda global* del sistema viene dado por la expresión:

$$ABG = \frac{n * 4Mbytes}{t_n \mu s} = \frac{n * 4 * 8 * 1048576bits}{t_n * 10^{-6} s}$$

Tabla 4.7: *ABG-MultiPingPong* con Hub-10*Mbps*

nº nodos	**2**	**4**	**8**	**16**
n	1	2	4	8
latencia(μs)	4363298	8040792	15846112	31202804
ABG($Mbps$)	7,69	8,34	8,47	8,60

Con los datos de la tabla 4.7 comprobamos que, en el caso de utilizar un hub para la interconexión, el *ancho de banda global* no aumenta al aumentar el número de conexiones simultáneas, sino que se mantiene constante.

Se hace notar que en este caso tampoco ha habido pérdida de generalidad al considerar un tamaño concreto de mensaje ya que al igual que en los otros casos, para un valor de n dado, el tiempo de envío es directamente proporcional al tamaño de mensaje.

Wifi: Las gráficas correspondientes a la red wifi (4.13 y 4.14) muestran un comportamiento bastante parecido al de la red conectada mediante un hub. Es decir esta red pierde ancho de banda en cada nodo a medida que aumenta el número de comunicaciones simultáneas.

Figura 4.13: *Multi-PingPong* Wifi

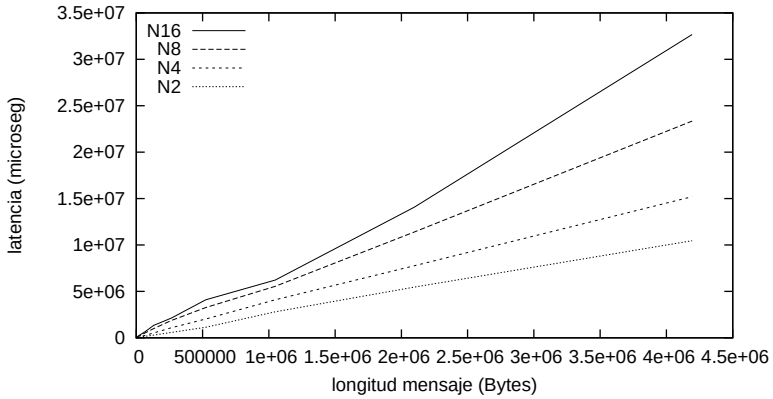
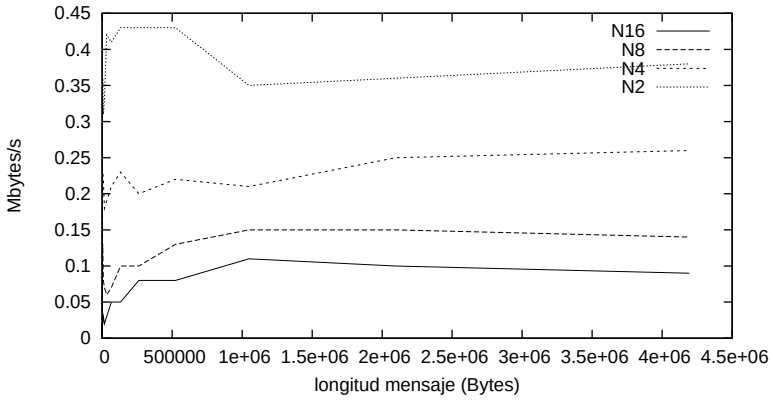


Figura 4.14: *Multi-PingPong* Wifi



Realizamos, para la red inalámbrica, cálculos análogos a los realizados para las otras redes. Consideramos envíos simultáneos de $x = 4Mbytes$:

$$ABG = \frac{n * 4Mbytes}{t_n \mu s} = \frac{n * 4 * 8 * 1048576bits}{t_n * 10^{-6} s}$$

Tabla 4.8: *ABG-MultiPingPong* con Wifi

nº nodos	**2**	**4**	**8**	**16**
n	1	2	4	8
latencia(μs)	10464152	15195967	23351171	32669240
ABG($Mbps$)	3,20	4,42	5,75	8,22

Aunque se observa (tabla 4.8) que el *ancho de banda global* aumenta con el número de comunicaciones simultáneas, no se trata de un aumento comparable al observado en las redes conectadas mediante un switch, sino que se trata de un pequeño aumento, más parecido al observado en el caso del hub.

En la figuras 4.15 y 4.16 (la segunda es un zoom de la primera) se comparan los tiempos obtenidos al ejecutar *IMB-MultiPingPong* sobre cada una de las redes consideradas, para un tamaño de mensaje de $4MBytes$. A la vista de éstas, se comprueba, de nuevo, el efecto que tiene el número de comunicaciones simultáneas sobre el tiempo necesario para llevarlas a cabo. Vemos cómo este tiempo se mantiene constante en las redes conectadas mediante un switch y cómo aumenta de forma proporcional al número de comunicaciones en el caso del hub. Respecto a la red wifi, en principio parece comportarse de forma bastante similar a la red conectada mediante un hub, pero, tal vez debido a la elevada variabilidad detectada en la red wifi, no es posible concluir nada definitivo al respecto.

Figura 4.15: Multi-PingPong, comparación para $4MBytes$

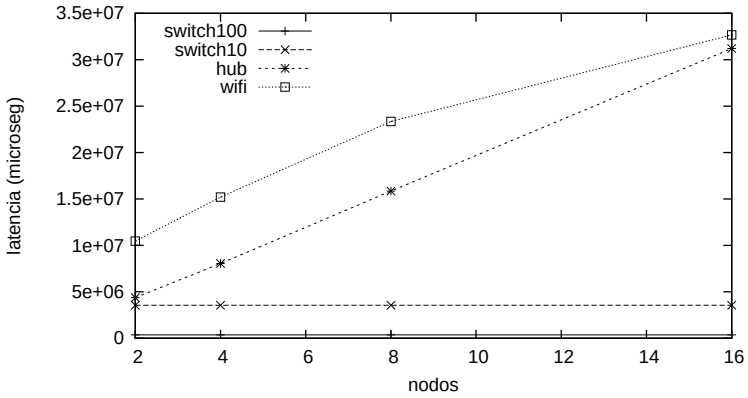
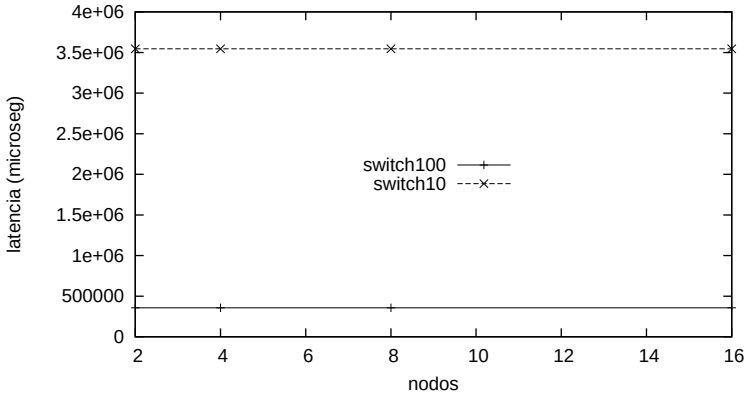


Figura 4.16: Multi-PingPong, comparación para $4MBytes$



4.2.4. Multi-PingPing

Switch-100Mbps: Las figuras 4.17 y 4.18 muestran la latencia y ancho de banda proporcionados por *IMB-MultiPingPing*.

Figura 4.17: *Multi-PingPing* Switch-100*Mbps*

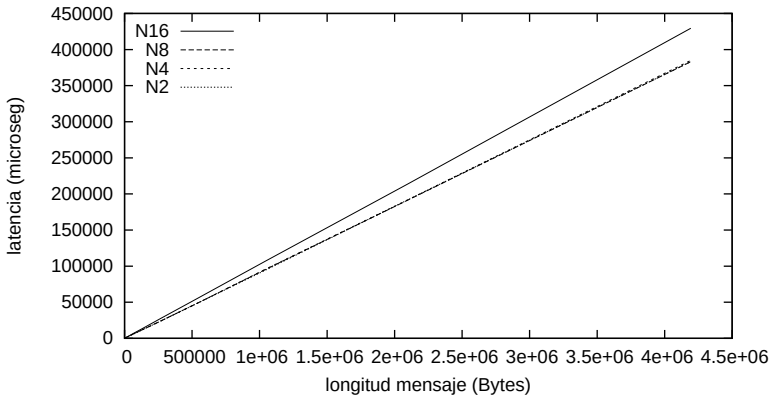
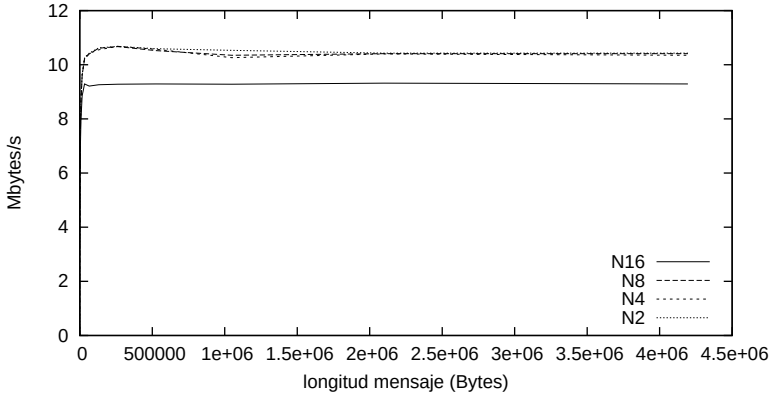


Figura 4.18: *Multi-PingPing* Switch-100*Mbps*



Con los datos obtenidos de *IMB-PingPing* (sección 4.2.2) e *IMB-MultiPingPong* (sección 4.2.3) cabría esperar que el número de nodos apenas afectase al tiempo de envío. Sin embargo, la gráfica 4.17 muestra que para 16 nodos (8 parejas ejecutando *IMB-PingPing*), el tiempo es ligeramente mayor que para 8, 4 y 2 nodos. Por otro lado, la figura 4.21, correspondiente a la ejecución de *IMB-MultiPingPing* en la red conectada mediante un switch de 10*Mbps*, sí presenta el comportamiento esperado. Dado que, hasta el momento, las dos redes conectadas mediante un switch se han comportado de forma análoga frente a envíos múltiples, nos planteamos la posibilidad de que la anomalía observada en la gráfica 4.17 se deba a alguna particularidad de funcionamiento del switch de 100*Mbps*. Para comprobar esta hipótesis se repite la misma prueba utilizando un swtich de 1*Gbps* trabajando a 100*Mbps* y se obtienen los resultados que se muestran en la figura 4.19

Figura 4.19: *Multi-PingPing* Switch-1*Gbps*

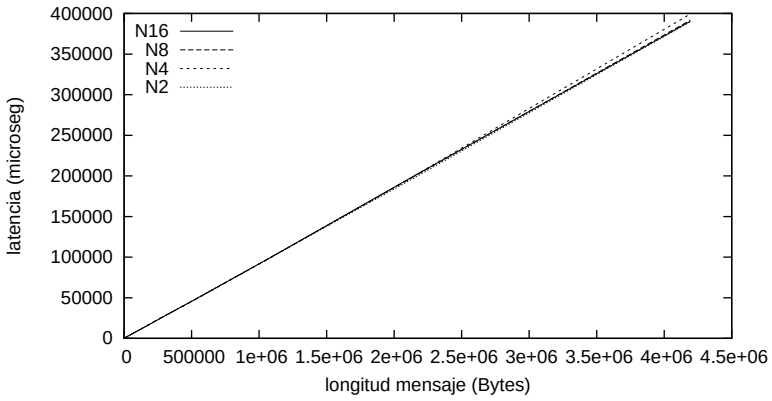
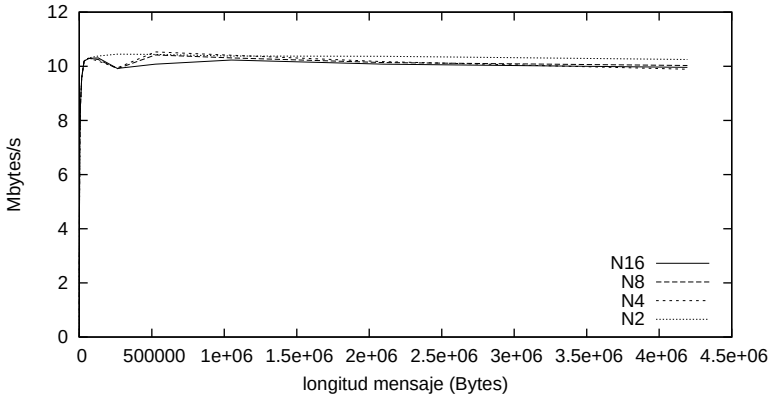


Figura 4.20: *Multi-PingPing* Switch-1*Gbps*



En la nueva ejecución de *IMB-MultiPingPing* en una red conectada con un switch funcionando a $100Mbps$ (figura 4.19), obtenemos el resultado esperado, confirmando nuestra hipótesis. Por otra parte, en la figura 4.20 se puede observar que para todas las parejas de nodos, el ancho de banda es similar, a diferencia de la figura 4.18, donde se empleaba el switch de $100Mbps$.

Calcularemos ahora, tal como hicimos en la sección 4.2.3, la ganancia de ancho de banda que se obtiene cuando n parejas de nodos ejecutan *IMB-PingPing* de forma simultánea. Cada par de nodos realiza dos envíos de longitud $xbytes$, en un tiempo t_n que dependerá del tamaño del mensaje y del número de nodos. Se tiene la siguiente expresión para el *ancho de banda global* (ABG):

$$ABG = \frac{n * 2 * xbytes}{t_n \mu s}$$

Tomando $x = 4Mbytes$ se tiene:

$$ABG = \frac{n * 2 * 4Mbytes}{t_n \mu s} = \frac{n * 2 * 4 * 8 * 1048576}{t_n * 10^{-6}} Mbps$$

Los valores de *ancho de banda global* obtenidos se muestran en la tabla 4.9. Observamos que se duplica el *ABG* que se había obtenido al ejecutar *MultiPingPong* para esta misma red. Esto es debido a que la red conectada mediante un switch permite comunicaciones fullduplex entre dos nodos, por lo que los dos envíos que realiza cada par de nodos pueden llevarse a cabo de forma simultánea.

Tabla 4.9: *ABG-MultiPingPing* con Switch-100*Mbps*

nº nodos	**2**	**4**	**8**	**16**
n	1	2	4	8
latencia(μs)	389970	399322	391760	390326
ABG($Mbps$)	172	336	685	1375

Switch-10Mbps: En las figuras 4.21 y 4.22 se puede observar que el número de comunicaciones simultáneas no influye en los resultados de *IMB-PingPing*. En concreto, se comprueba que el ancho de banda en cada nodo se mantiene constante independientemente del número de comunicaciones paralelas.

Figura 4.21: Switch-10*Mbps*/Multi-PingPing

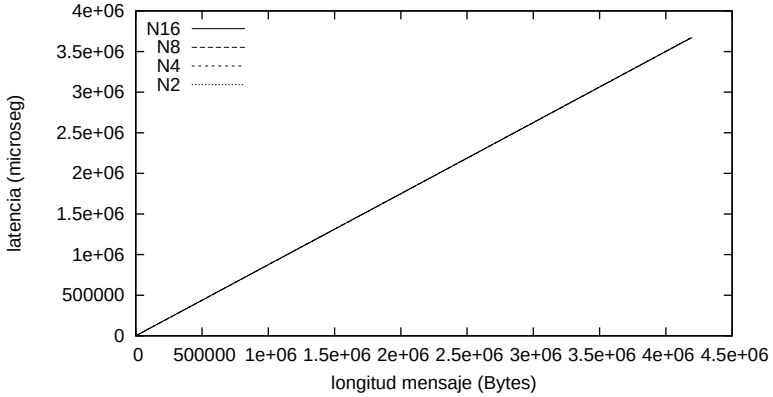
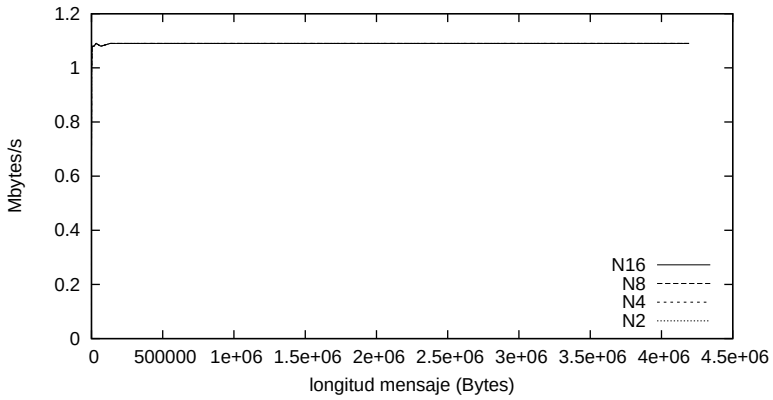


Figura 4.22: Switch-10$Mbps$/Multi-PingPing



La tabla 4.10 muestra los resultados del cálculo del *ancho de banda global* considerando $x = 4Mbytes$.

$$ABG = \frac{n * 2 * 4Mbytes}{t_n \mu s} = \frac{n * 2 * 4 * 8 * 1048576}{t_n * 10^{-6}} Mbps$$

Tabla 4.10: *ABG-MultiPingPing* con switch-10$Mbps$

nº nodos	**2**	**4**	**8**	**16**
n	1	2	4	8
latencia(μs)	3673237	3672678	3670617	3670796
ABG($Mbps$)	18,3	36,5	73,1	146,2

A la vista de los resultados de la tabla 4.10, comprobamos, por un lado que el *ABG* obtenido es el doble que el correspondiente al *IMB-MultiPingPong* y por otro, que al duplicar el número de comunicaciones se duplica el *ancho de banda global*.

El *ABG* correspondiente a la red con switch de 100$Mbps$ es casi 10 veces el de la red conectada mediante switch de 10$Mbps$.

Hub: Las figuras 4.23 y 4.24 muestran los datos proporcionados en la ejecución de *MultiPingPing* sobre la red conectada mediante un hub.

Figura 4.23: *Multi-PingPing* Hub

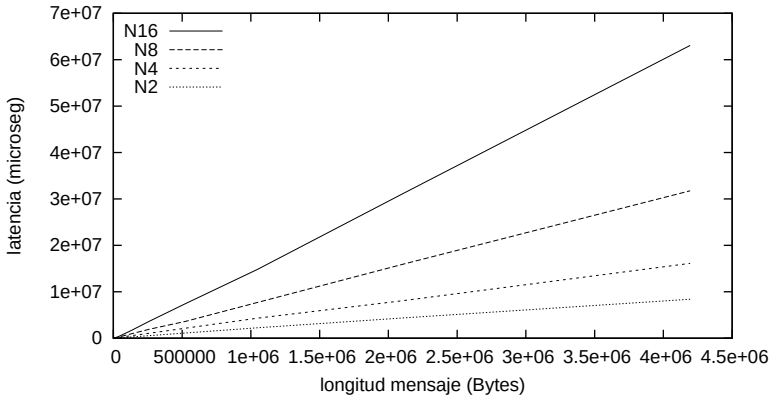
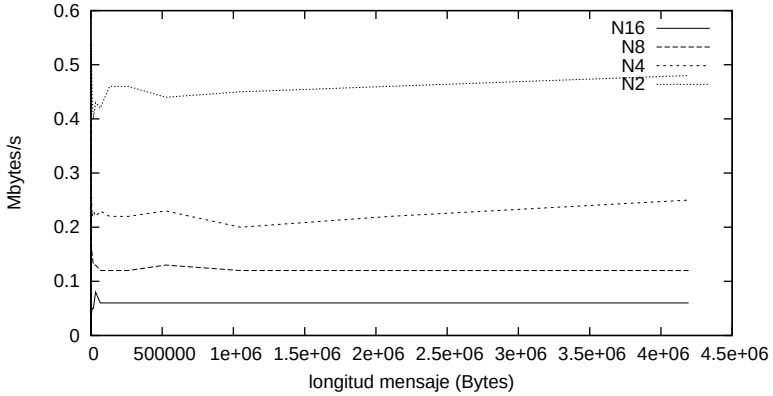


Figura 4.24: *Multi-PingPing* Hub



La tabla 4.11 muestra lo datos obtenidos en el cálculo del *ancho de banda global* para $x = 4Mbytes$ en la red hub.

$$ABG = \frac{n * 2 * 4Mbytes}{t_n \mu s} = \frac{n * 2 * 4 * 8 * 1048576}{t_n * 10^{-6}} Mbps$$

Tabla 4.11: *ABG-MultiPingPing* con Hub-10$Mbps$

nº nodos	**2**	**4**	**8**	**16**
n	1	2	4	8
latencia(μs)	8398121	16107594	31762929	63063357
ABG($Mbps$)	7,99	8,33	8,45	8,51

A la vista de las figuras 4.23 y 4.24, y de la tabla 4.11, comprobamos que se obtienen los resultados esperados:

- La latencia y el ancho de banda (local) varía con el número de nodos. Al duplicar el número de nodos se duplica la latencia y se reduce el ancho de banda a la mitad.
- El ancho de banda en cada nodo es la mitad que el obtenido para *IMB-MultiPingPong*.
- El *ancho de banda global* no varía con el número de nodos y coincide con el obtenido para *IMB-MultiPingPong*.

Wifi: Las figuras 4.25 y 4.26 muestran el comportamiento que presenta esta red al ejecutar *IMB-MultiPingPing*.

Figura 4.25: *Multi-PingPing* Wifi

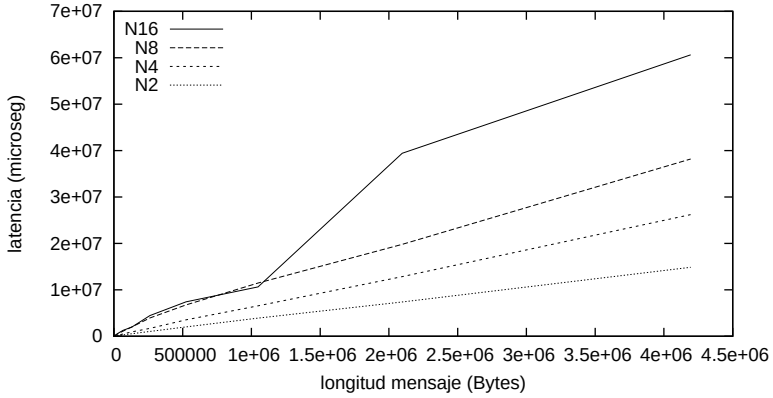
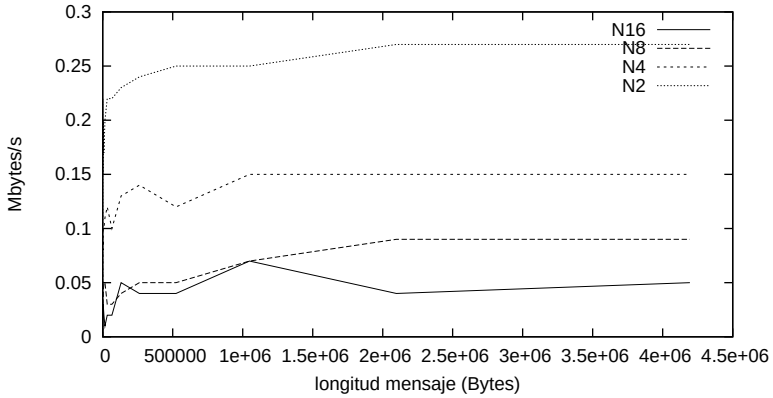


Figura 4.26: *Multi-PingPing* Wifi



Se realizan, para la red wifi, cálculos del *ancho de banda global* corespondiente a la función *IMB-MultiPingPing*, de forma similar a los efectuados para las otras redes. Los resultados obtenidos se muestran en la tabla 4.12.

Tabla 4.12: *ABG-MultiPingPing* con wifi

nº nodos	2	4	8	16
n	1	2	4	8
latencia(μs)	14852572	26204922	38200406	60618160
ABG($Mbps$)	4,5	5,12	7,02	8,8

En las figuras 4.27 y 4.28, en las que se compara el comportamiento de las ditintas redes al ejecutar *IMB-MultiPingPing* para un tamaño de mensaje de $4MBytes$, observamos que el tiempo de envío se mantiene constante al variar el número de nodos en las redes conectadas por un switch, mientras que varía en las otras dos. En el caso del hub la variación es lineal. En el caso de la red wifi, aunque aparenta un comportamiento similar al de la red con hub, la variación no presenta una tendencia clara.

Figura 4.27: *Multi-PingPing*, comparación para $4MBytes$

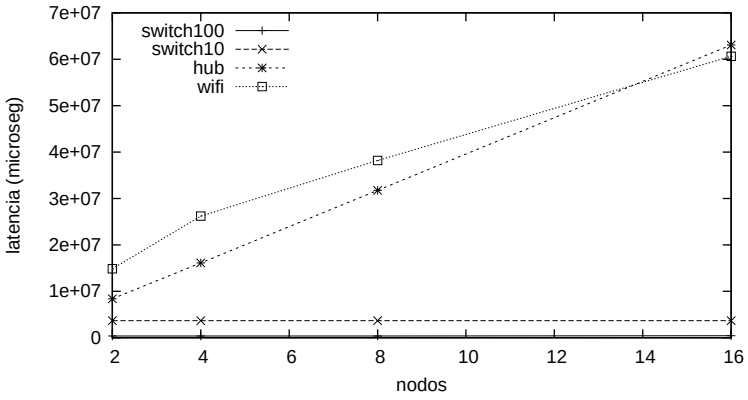
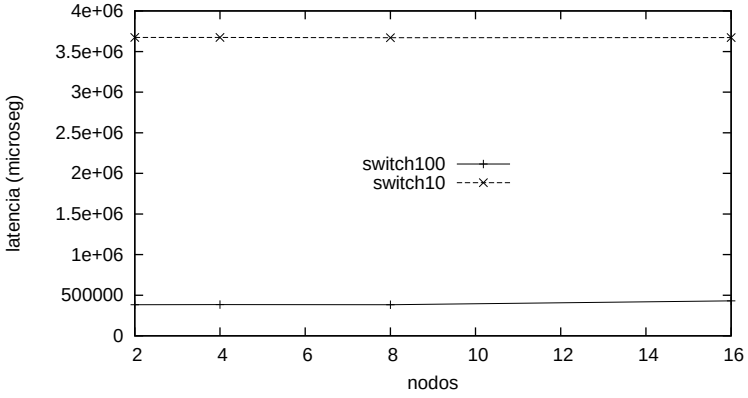


Figura 4.28: *Multi-PingPing*, comparación para $4MBytes$



4.2.5. Sendrecv

Con el fin de analizar el comportamiento que presentan las distintas redes al ejecutar *IMB-Sendrecv*, en las figuras 4.29 a 4.36 se muestran los resultados obtenidos en la ejecución de dicho benchmark sobre cada una de las redes.

Figura 4.29: *Sendrecv* Switch-100*Mbps*

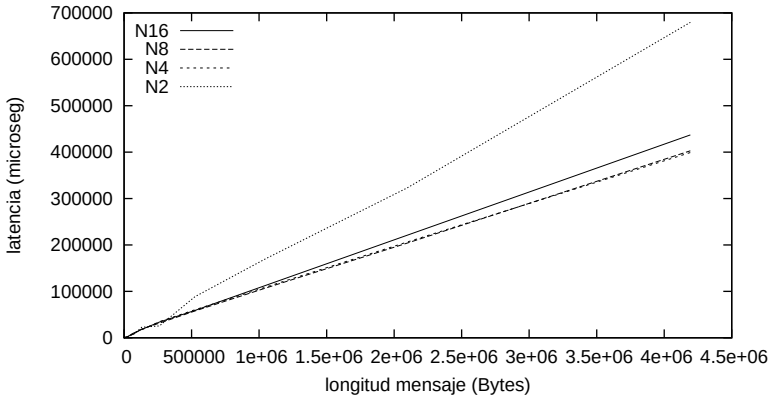


Figura 4.30: *Sendrecv* Switch-100*Mbps*

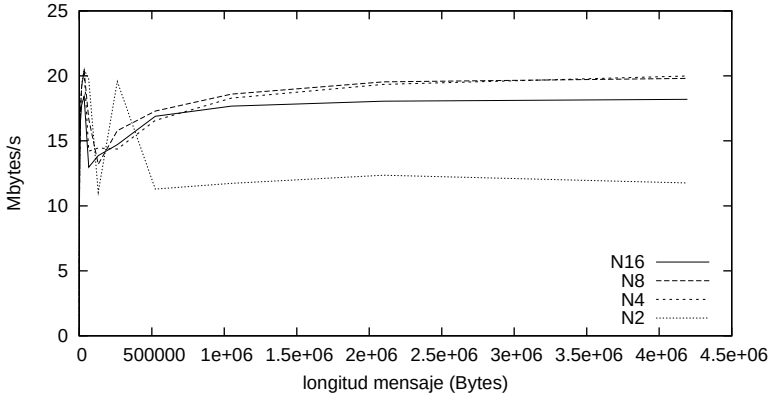


Figura 4.31: *Sendrecv* Switch-10*Mbps*

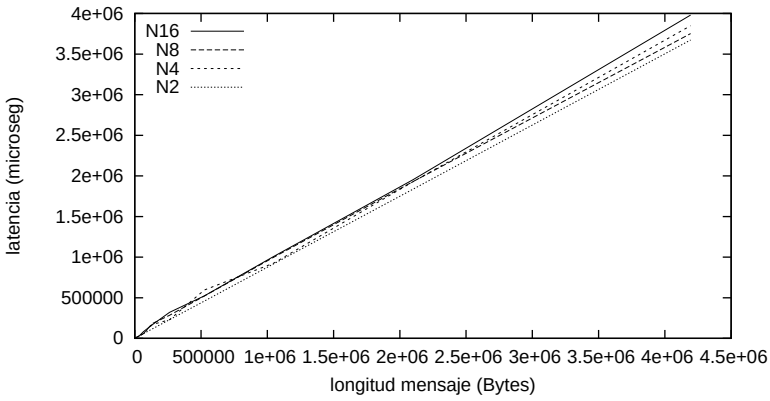


Figura 4.32: *Sendrecv* Switch-10*Mbps*

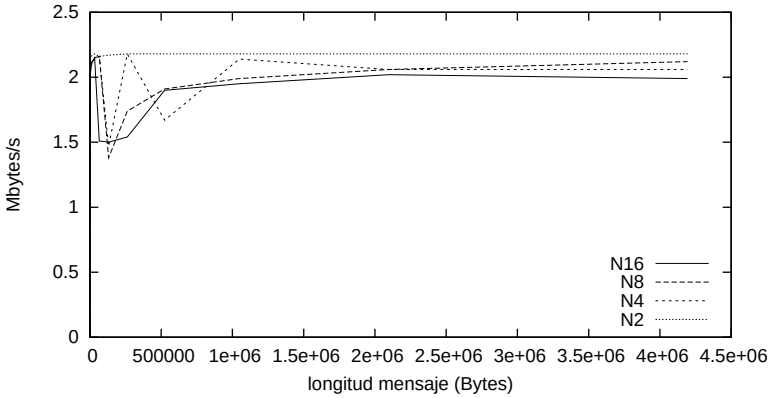


Figura 4.33: *Sendrecv* Hub

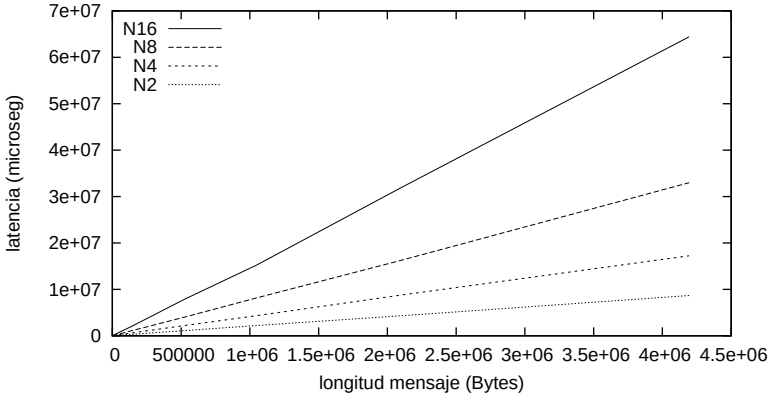


Figura 4.34: *Sendrecv* Hub

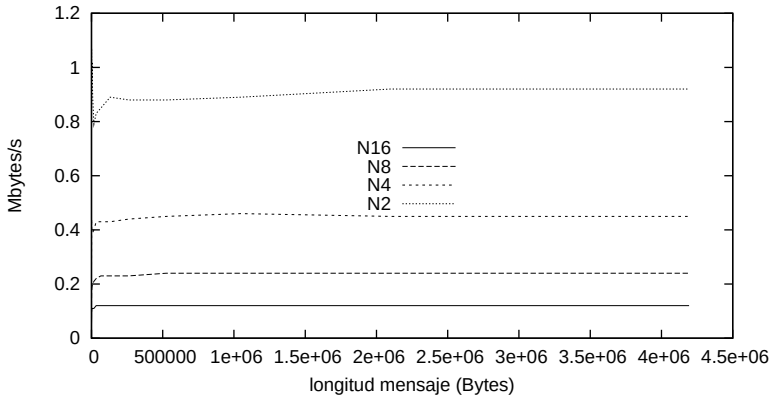


Figura 4.35: *Sendrecv* Wifi

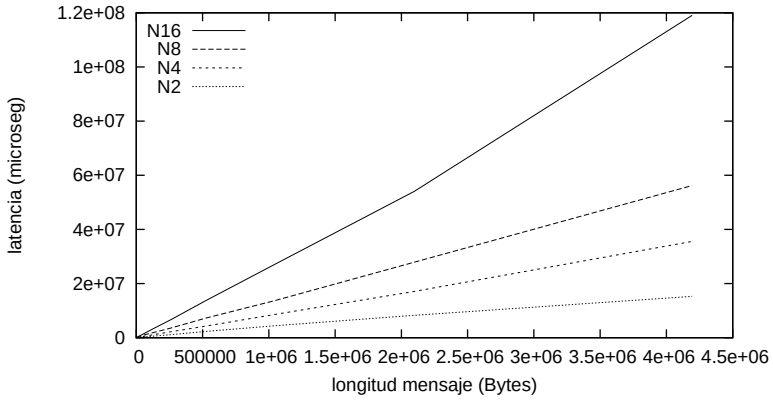
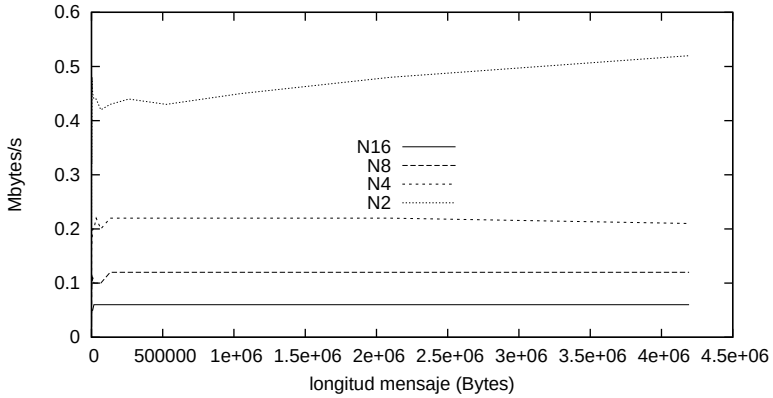


Figura 4.36: *Sendrecv* Wifi



Para el cálculo del *ancho de banda global* (correspondiente a N nodos),

se considera que en un tiempo t_N se realizan N envíos de $x = 4Mbytes$. Los resultados se muestran en las tablas 4.13 a 4.16

$$ABG = \frac{N * 4Mbytes}{t_N \mu s} = \frac{N * 4 * 8 * 1048576}{t_N * 10^{-6}} Mbps$$

Tabla 4.13: *ABG*-Sendrecv con Switch-100*Mbps*

nº nodos (N)	**2**	**4**	**8**	**16**
latencia(μs)	679481	399195	402837	437136
ABG(*Mbps*)	99	336	666	1228

Tabla 4.14: *ABG*-Sendrecv con Switch-10*Mbps*

nº nodos (N)	**2**	**4**	**8**	**16**
latencia(μs)	3673248	3849091	3751472	3980619
ABG(*Mbps*)	18,26	34,87	71,55	134,87

Tabla 4.15: *ABG*-Sendrecv con Hub-10*Mbps*

nº nodos (N)	**2**	**4**	**8**	**16**
latencia(μs)	8681823	17237575	32993052	64428851
ABG(*Mbps*)	7,73	7,78	8,13	8,33

Tabla 4.16: *ABG*-Sendrecv con Wifi

nº nodos (N)	**2**	**4**	**8**	**16**
latencia(μs)	15285366	35564079	56214253	119075487
ABG(*Mbps*)	4,39	3,77	4,77	4,5

Observamos que los resultados obtenidos son muy similares a los obtenidos para *IMB-MultiPingPing* (sección 4.2.4).

Se hacen las siguientes anotaciones:

- Las gráficas 4.29 y 4.30, que hacen referencia a la red conectada mediante un switch de 100*Mbps*, muestran un comportamiento inesperado para 2 nodos que podría ser debido a alguna anomalía de funcionamiento del switch de 100*Mbps* como la que se detectó en *IMB-MultiPingPing*. Para confirmar esta suposición, comprobamos que la

red conectada mediante un switch de $10Mbps$ no presenta ese comportamiento.

- Se observa cierta variabilidad, con el número de nodos, en las redes conectadas mediante un switch, que podría ser debida al hecho de que aunque la función *MPI-Sendrecv* se compone de un conjunto de funciones *MPI-Isend* y *MPI-Irecv* (igual que *IMB-MultiPingPing*), para llevarla a cabo se precisa una coordinación adicional entre todos los nodos.

- Los datos de ancho de banda que proporciona este benchmark tienen en cuenta dos envíos por nodo, mientras que *IMB-MultiPingPing* sólo considera uno de ellos. Esto hace que los valores de ancho de banda que proporciona *IMB-Sendrecv* sean el doble que los de *IMB-MultiPingPing*.

- En este ejemplo la red wifi muestra un comportamiento análogo al de la red conectada con un hub:

  - En las gráficas 4.35 y 4.36 se observa que al duplicar el número de nodos se duplica la latencia mientras que el ancho de banda (local) se reduce a la mitad.

  - El *ancho de banda global* se mantiene constante al variar el número de nodos (tabla 4.16).

Figura 4.37: *Sendrecv*, comparación para $4MBytes$

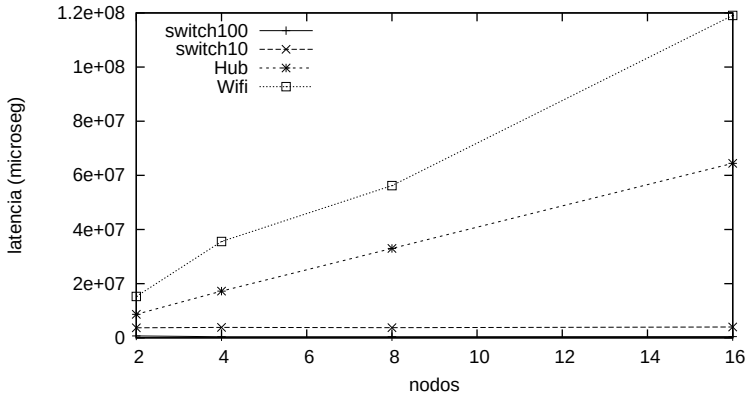
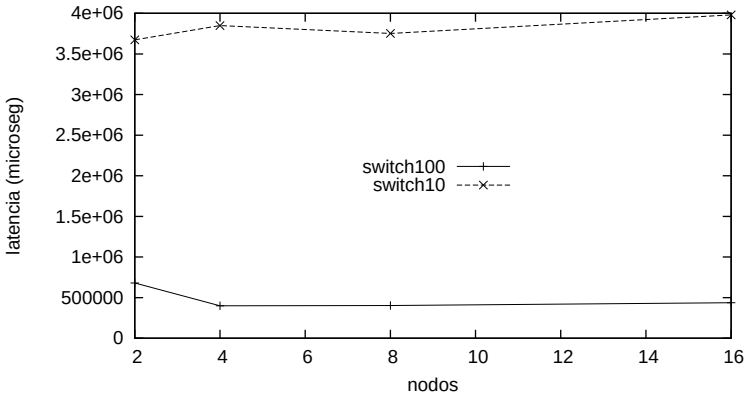


Figura 4.38: *Sendrecv*, comparación para $4MBytes$



Las gráficas que se muestran en las figuras 4.37 y 4.38 son similares a las obtenidas para *IMB-MultiPingPing* (Fig. 4.27). Aunque en este caso parece confirmarse la similitud, que no había podido ser determinada en otros ejemplos, entre el comportamiento de la red wifi y la red que utiliza un hub.

4.2.6. Conclusiones sobre los resultados de comunicación paralela

La idea principal que extraemos tras el análisis de los benchmarks que estudian las comunicaciones paralelas, es que las redes que utilizan un switch para la interconexión mejoran considerablemente su rendimiento ante comunicaciones simultáneas, mientras que no es así para las que utilizan un hub o wifi.

Las redes conectadas mediante un switch obtienen una importante ganancia de ancho de banda cuando se llevan a cabo comunicaciones simultáneas entre parejas de nodos. Esto es debido a que un switch permite varios canales de comunicación simultáneos. No ocurre lo mismo en la red conectada con un hub ni en la red wifi. Podría decirse que dado que en estas redes no son posibles los envíos simultáneos, ante cualquier intento de comunicación en paralelo, ésta quedaría serializada. Las transmisiones se ordenan transformando una comunicación paralela en secuencial.

4.2.7. Gather

Pasamos a analizar el primer ejemplo de comunicación colectiva.

Figura 4.39: *Gather* Switch-100*Mbps*

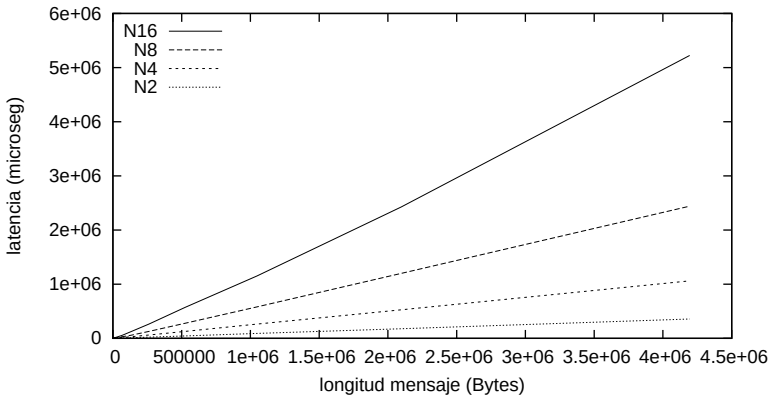


Figura 4.40: *Gather* Switch-10*Mbps*

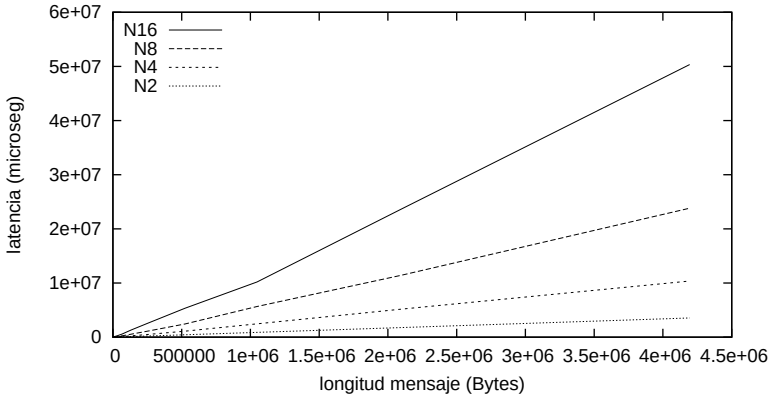


Figura 4.41: *Gather* Hub

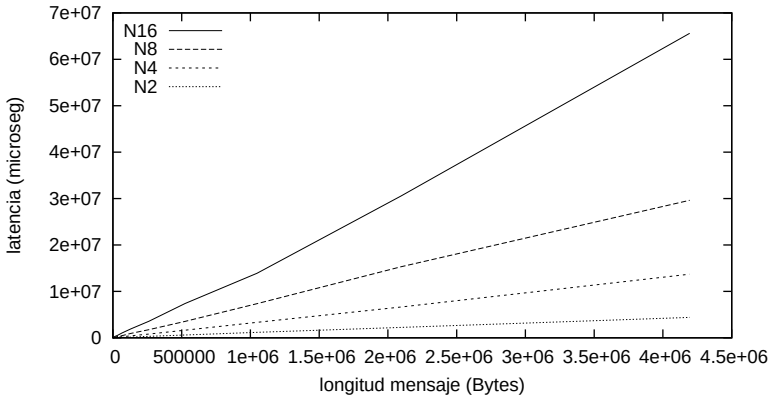
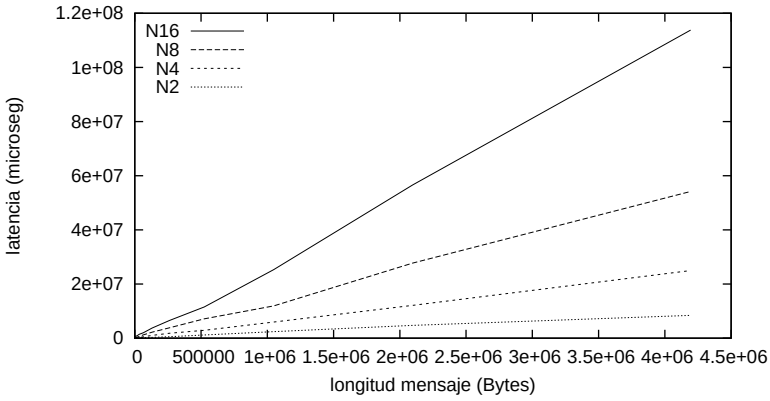


Figura 4.42: *Gather* Wifi



En este ejemplo, tal como puede observarse en las figuras 4.39 a 4.42, las cuatro redes muestran el mismo comportamiento: La latencia crece linealmente con el tamaño de envío (rectas) y es proporcional al número de nodos. Este comportamiento coincide con el que hasta ahora presentaba la red conectada mediante un hub.

Cuando N nodos invocan la función MPI-Gather, se llevan a cabo $N-1$ envíos de datos desde diferentes nodos a un nodo maestro (N_0). Por otro lado, en cualquiera de las redes que estudiamos, dado que los envíos se realizan todos a un mismo nodo N_0, éste sólo podrá recibir datos de uno de los nodos en un momento determinado. Es decir, los envíos no podrán ser simultáneos en ningún caso, siempre serán secuenciales. Este hecho hace que, al ejecutar *MPI-Gather*, las redes conectadas mediante un switch no tengan en este caso ganancia alguna y presenten el mismo comportamiento que las otras redes.

Figura 4.43: *Gather*, comparación para $4MBytes$

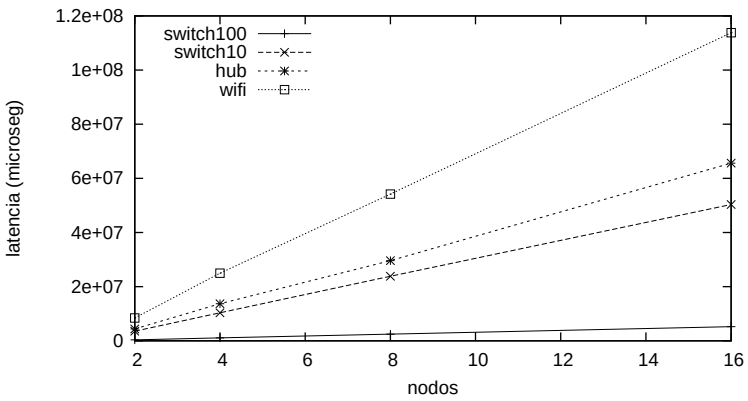
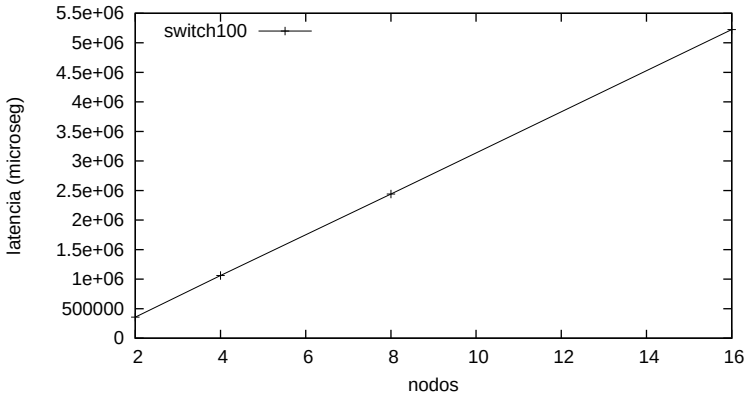


Figura 4.44: *Gather*, comparación para $4MBytes$



A la vista de las gráficas 4.43 y 4.44, se vuelve a comprobar que las cuatro redes tienen el mismo comportamiento al invocar *IMB-Gather*. Observamos que la latencia crece de forma lineal con el número de nodos. Más concretamente, la latencia es propocional al número de envíos que, en este caso, es proporcional al número de nodos.

4.2.8. Scatter

Se trata de otro ejemplo de comunicación colectiva que tiene un comportamiento similar a la función anterior.

Figura 4.45: *Scatter* Switch-$100Mbps$

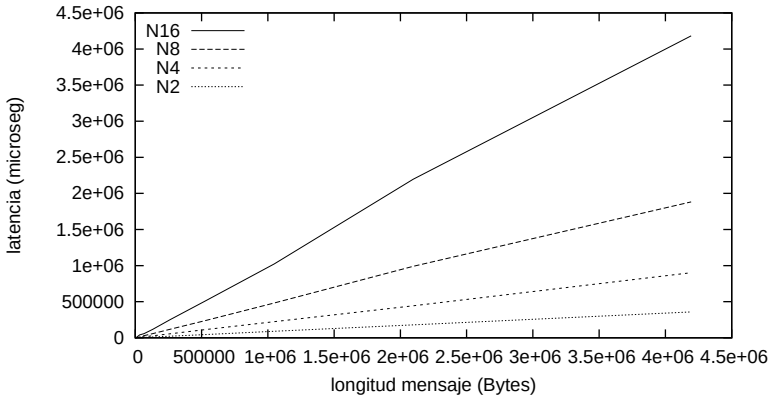


Figura 4.46: *Scatter* Switch-10$Mbps$

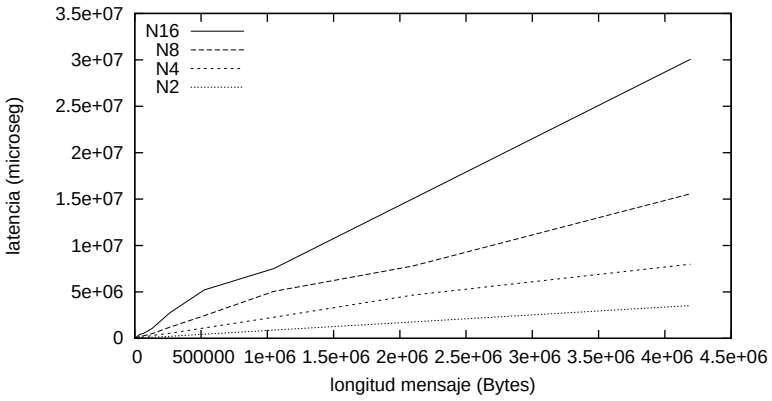


Figura 4.47: *Scatter* Hub

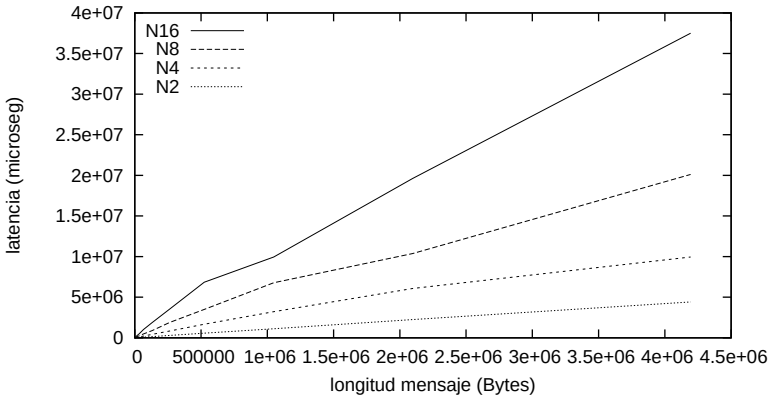
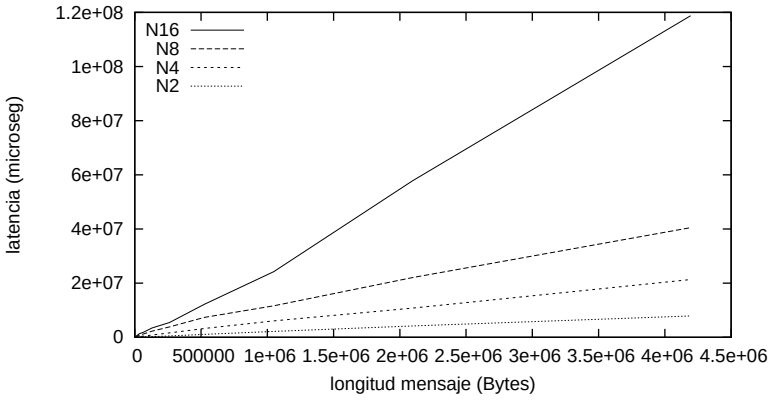


Figura 4.48: *Scatter* Wifi



Las gráficas correspondientes a este benchmark (figuras 4.45 a 4.48) mues-

tran que en la ejecución de la función *MPI-Scatter*, al igual que ocurría con *MPI-Gather* (4.2.7) no hay envíos simultáneos, ya que las redes conectadas mediante switch se comportan igual que la conectada mediante hub.

Al ejecutar la función *MPI-Scatter* con N nodos, un nodo maestro (N_0) reparte un vector de datos entre los $(N-1)$ nodos restantes. Es decir , se producen $N-1$ envíos desde un único nodo al resto. Dado que un nodo sólo puede realizar un envío en un momento dado, la comuncación no será simultánea sino secuencial.

Figura 4.49: Scatter, comparación para $4MBytes$

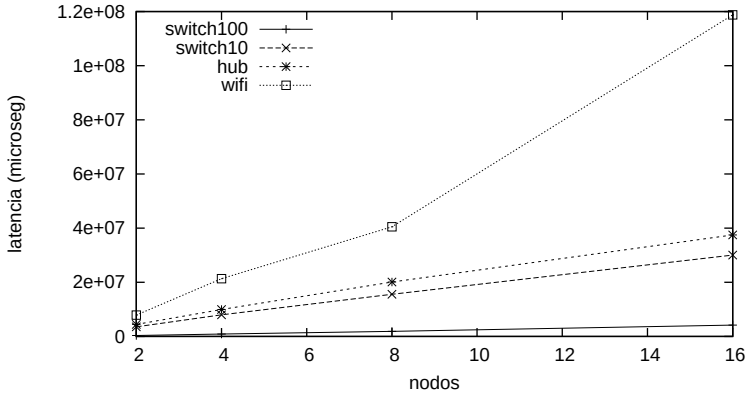
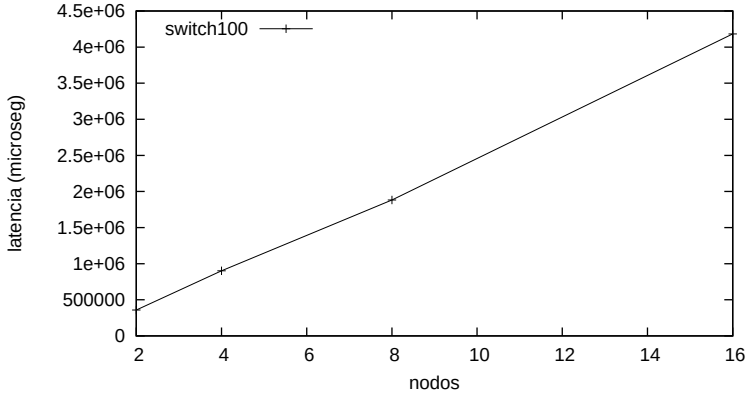


Figura 4.50: Scatter, comparación para $4MBytes$



En los ejemplos *Gather* y *Scatter*, se ha podido comprobar que cuando las distintas comunicaciones se hacen de forma secuencial, las cuatro redes estudiadas tienen un comportamiento similar, en el que la latencia es múltiplo del número de mensajes.

4.2.9. Bcast

Este benchmark proporciona un ejemplo en el que una parte de las comunicaciones son secuenciales y otras paralelas.

Figura 4.51: *Bcast* Switch-100*Mbps*

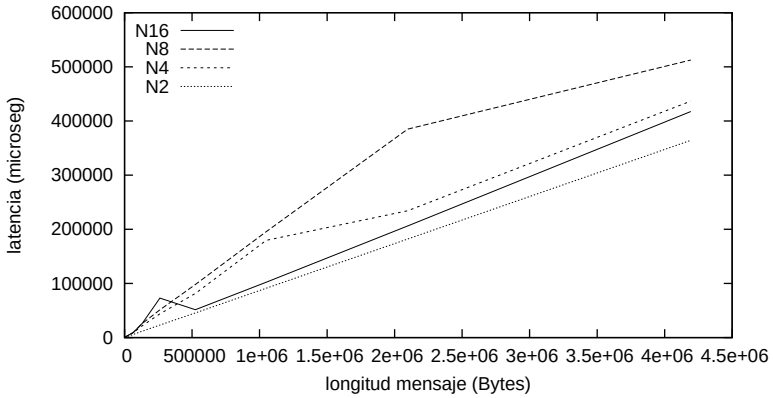


Figura 4.52: *Bcast* Switch-10*Mbps*

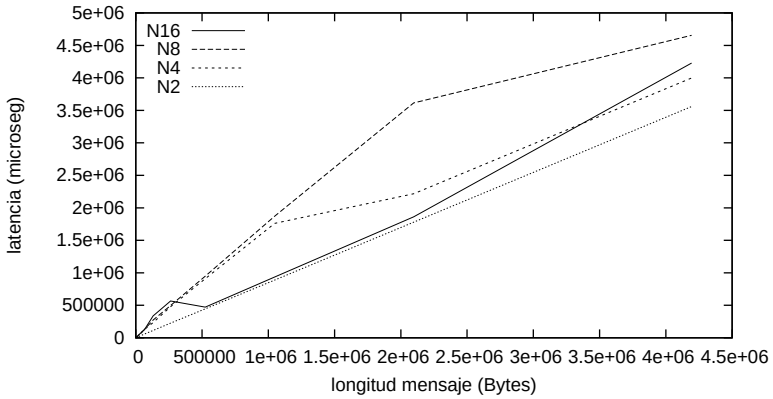


Figura 4.53: *Bcast* Hub

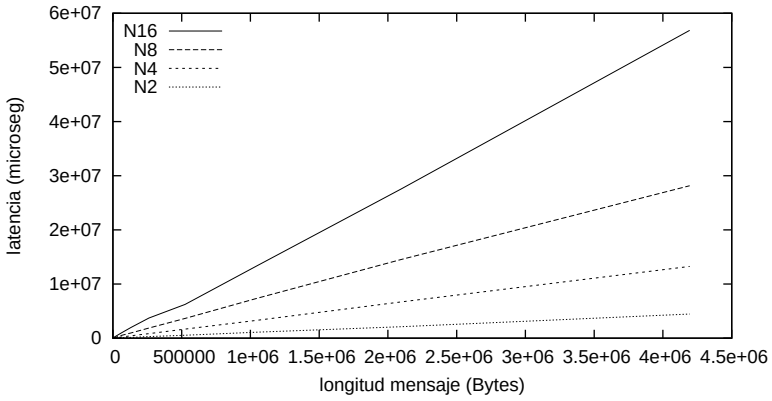
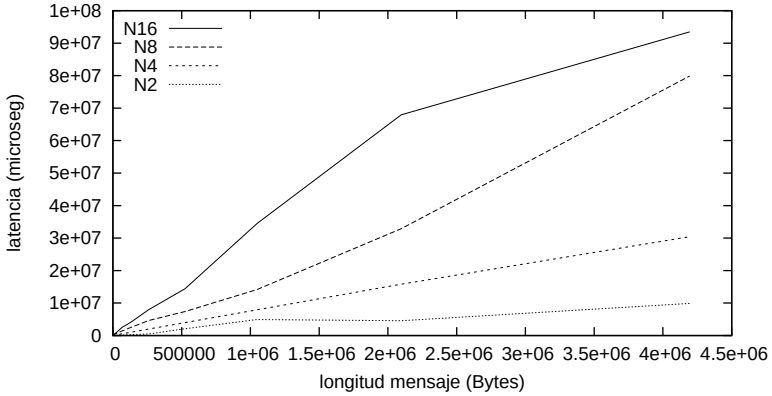


Figura 4.54: *Bcast* Wifi



En las figuras de este ejemplo (4.51 a 4.54), observamos, por un lado, que la red conectada mediante un hub muestra una gráfica típica de comunicaciones secuenciales, similar al resto de gráficas correspondientes a esta red. Es decir, la latencia crece linealmente con el tamaño de envío y al duplicar el número de nodos se duplica (aproximadamente) la pendiente de la recta correspondiente. Por otro lado las redes conectadas mediante un switch muestran una gráfica que no se corresponde con una comunicación simultánea ni con una secuencial. En estas gráficas llama la atención el hecho de que la latencia no es lineal con el tamaño del envío, es decir, el comportaniento de la función *MPI-Bcast* depende del tamaño del mensaje a enviar. Esta diferencia de comportamiemto vendrá determinada por la forma en que la función está implementada.

Al ejecutar *MPI-Bcast*, un nodo maestro envía un mismo mensaje al resto de nodos. Una forma de implementar la función sería que el nodo N_0 enviase, de forma secuencial, el mensaje a cada uno de los nodos. Si fuese así, las gráficas deberían ser similares a las correspondientes a la función *MPI-*

Scatter (sección 4.2.8). Además, en ese caso, en el que no se producirían comunicaciones simultáneas, las redes conectadas mediante un switch no obtendrían ganancia de ancho de banda, por lo que sería una implementación poco eficiente.

Otra posibilidad es hacer el envío utilizando un árbol de comunicaciones. Dado que el mensaje a transmitir es siempre el mismo, una vez que el nodo maestro ha realizado el primer envío, el nodo que lo ha recibido puede, a su vez, reenviar el mensaje. De esta forma, cada nodo que recibe el mensaje lo reenviará a otro nodo consiguiendo así que parte de las comunicaciones sean simultáneas. Supongamos que N nodos ejecutan *MPI-Bcast* para un mensaje de tamaño x, que tarda un tiempo t_x en ser enviado de un nodo a otro. En un tiempo $t = t_x$ han recibido el mensaje 2 nodos, para $t = 2t_x$ lo han recibido 4 nodos, para $t = 3t_x$ 8 nodos. En general, para $t = nt_x$ lo habrán recibido 2^n nodos. Si tenemos N nodos, se precisa un tiempo $t = \log_2(N)\, t_x$ para completar la función. Aunque la explicación anterior justificaría el hecho de que las comunicaciones son en parte secuenciales y en parte paralelas, no justifica la forma observada en las gráficas, en las que la latencia no es proporcional al tamaño del mensaje. Tampoco queda justificado el hecho de que para 16 nodos se obtenga menor latencia que para 8 ó 4.

Supongamos ahora que el mensaje a enviar, de tamaño x, es dividido en q mensajes, tendremos que cada nuevo mensaje tarda un tiempo $t_q = \frac{t_x}{q}$ en ser enviado de un nodo a otro. Dado que $t_q < t_x$, será menor el intervalo de tiempo necesario para que se multiplique el número de envíos simultáneos.

A modo de ejemplo, consideremos $N = 4$. El tiempo necesario para llevar a cabo 4 envíos viene dado por la expresión:

$$t = \log_2(N)\, t_x = \log_2(4)\, t_x = 2t_x \tag{4.1}$$

En cambio, si en el ejemplo anterior dividimos cada mensaje en 2 partes ($q = 2$), podremos enviar, de forma equivalente, $2 * 4$ mensajes de longitud $\frac{x}{q} = \frac{x}{2}$. Dado que cada nuevo mensaje tarda $t_2 = \frac{t_x}{2}$, el tiempo necesario para realizar los 8 nuevos envíos será:

$$t = \log_2(qN)\, t_q = \log_2(8)\, \frac{t_x}{2} = \frac{3}{2} t_x \tag{4.2}$$

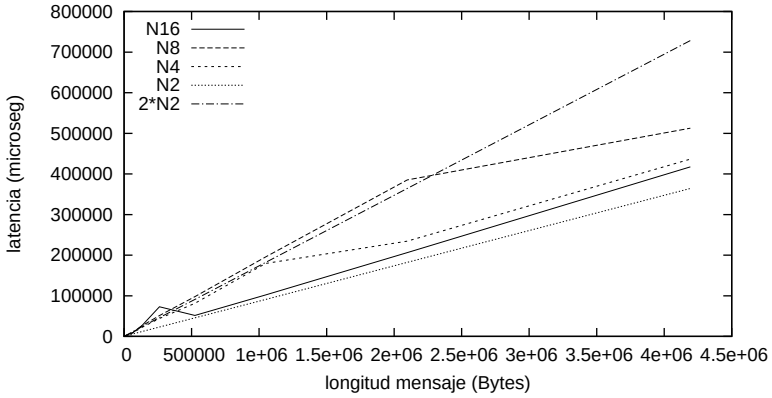
A la vista de las ecuaciones 4.1 y 4.2 vemos cómo al dividir el mensaje se consigue una ganancia debida al aumento de las comunicaciones simultáneas.

Hemos visto que estructurando los envíos en forma de árbol y dividiendo el mensaje en otros más pequeños se obtiene una mayor eficiencia en la ejecución de la función *MPI-Bcast*. Ahora bien, el hecho de dividir los mensajes y multiplicar el número de ellos añade una sobrecarga de tiempo que sólo será rentable si t_x es lo suficientemente grande, es decir, si el tamaño del mensaje es suficientemente grande. En resumen, que dependiendo del tamaño de envío se dividirá o no el mensaje original.

El razonamiento anterior podría justificar el hecho de que en las gráficas 4.51 y 4.52 la latencia no varíe de forma lineal al variar el tamaño de envío. En ellas observamos que, como era de esperar, para 2 nodos la latencia sí crece de forma lineal con el tamaño del envío. Por otro lado, las curvas correspondientes a 4, 8 y 16 nodos están aproximadamente comprendidas entre la gráfica que se corresponde con 2 nodos, y el doble de ésta. Esto se debe a que mediante la división del mensaje original, tal como se ha explicado antes, se consigue que el tiempo de envío no supere $2t_x$, siendo t_x el tiempo necesario para enviar el mensaje original de un nodo a otro.

En la gráfica 4.55 se muestra de nuevo la gráfica 4.51, a la que se le ha añadido, a modo de referencia, la línea $2 \times N2$ obtenida multiplicando por 2 la línea correspondiente a 2 nodos.

Figura 4.55: Switch-100$Mbps$/Bcast



A continuación se describe, de una forma aproximada, un posible modo en que se dividirían los mensajes iniciales, dependiendo del tamaño de éstos y del número de nodos, para obtener una gráfica similar a la representada en la figura 4.55. Consideremos la fórmula, para N nodos, que nos permite calcular el tiempo de envío necesario cuando se divide un mensaje de tamaño x en q partes:

$$t = \log_2(qN)\, t_q = \log_2(qN)\, \frac{t_x}{q} \tag{4.3}$$

16-nodos: Para un tamaño de mensaje x, menor y próximo a 262144$bytes$, la curva de 16 nodos está por encima de línea $2 \times N2$, aunque aparentemente no se aleja demasiado. Posiblemente para estos valores de x el mensaje se divide en 2 para mejorar el rendimiento. Utilizando la ecuación (4.3) con $N = 16$ y $q = 2$ tendremos $t = \log_2(32)\, \frac{t_x}{2} = \frac{5}{2} t_x = 2{,}5t_x$. Este valor explicaría que la curva esté ligeramente por encima de la línea $2 \times N2$.

Para $x = 524288bytes$ la curva baja por debajo de la línea de referencia $2 \times N2$. En este caso, se habría producido una nueva división

de mensajes y se tendría $N = 16$ y $q = 4$. Aplicando la fórmula (4.3) nuevamente, tendremos $t = \log_2(64)\frac{t_x}{4} = \frac{6}{4}t_x = 1{,}5t_x$. A la vista de la gráfica 4.55 parece claro, sin embargo, que la reducción de la latencia es mayor, por lo que el valor de q tendría que ser mayor que el supuesto. Si tomamos $q = 8$ tendríamos $t = \frac{7}{8}t_x = 0{,}9t_x$ que parece ajustarse mejor a la curva. Se hace notar que aunque $t = 0{,}9t_x$ supondría que la curva correspondiente a 16 nodos estuviera por debajo de la de 2 nodos, esto no sucede debido a la sobrecarga que indudablemente se añade al dividir el mensaje original en 8 nuevos mensajes.

8-nodos: Para un tamaño de mensaje x menor de $2097152bytes$, la curva de 8 nodos es similar a la línea $2 \times N2$. Esto se consigue dividiendo el mensaje original entre 2. En efecto, si consideramos la ecuación 4.3 con $N = 8$ y $q = 2$ obtenemos $t = \log_2(16)\frac{t_x}{2} = \frac{4}{2}t_x = 2t_x$, tal como se esperaba. Para $x = 4194304bytes$ vuelve a haber una nueva división. Tomamos $N = 8$ y $q = 4$ y obtenemos $t = 1{,}25t_x$ que se ajusta (considerando la sobrecarga) a lo observado en la gráfica.

4-nodos: Para este número de nodos, la primera división de mensaje se hace para $x = 2097152bytes$. Considerando $N = 4$, $q = 2$ se tiene $t = 1{,}5t_x$ que no coincide con lo observado. Tomando $N = 4$, $q = 4$ se tiene $t = 1t_x$, que una vez considerada la sobrecarga correspondiente, sí se ajusta a la curva de 4 nodos.

Se hace notar que dado que la red conectada mediante un hub secuencializa cualquier intento de comunicación simultánea, no va a obtener niguna ventaja de la implementación de la función *bcast*. Aunque los mensajes se dividan, no se enviarán de forma simultánea, sino que serán enviados uno tras otro. En el caso de esta red, esta implementación empeora por tanto los tiempos de latencia ya que la sobrecarga añadida por la fragmentación de los mensajes, no se ve compensada en modo alguno. Respecto a la red wifi tiene un comportamiento análogo a la red conectada con un hub, con el inconveniente añadido de que la sobrecarga parece acusarse más en esta red.

El estudio del modo en que la función *MPI-bcast* ha sido implementada, nos permite deducir que la versión de *MPI* que utilizamos no ha sido diseñada para ser ejecutada utilizando una red del tipo hub o wifi.

Figura 4.56: Bcast, comparación para $4MBytes$

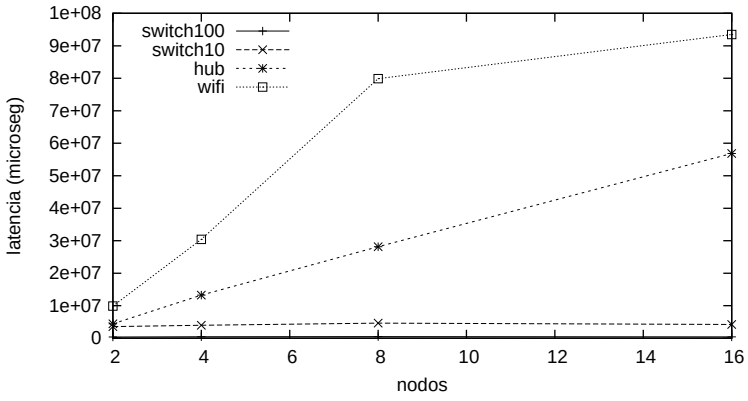


Figura 4.57: Bcast, comparación para $4MBytes$

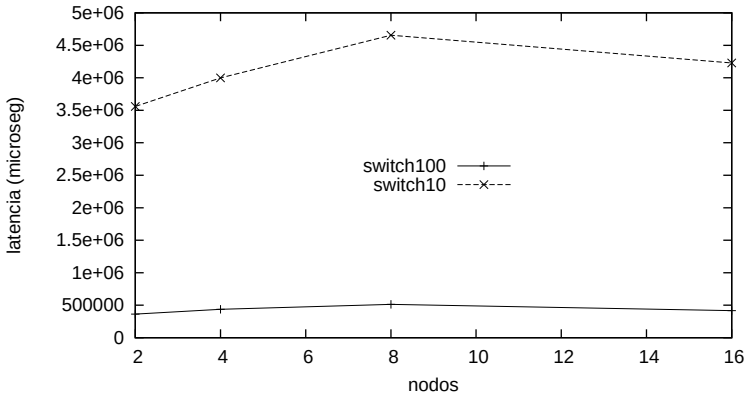
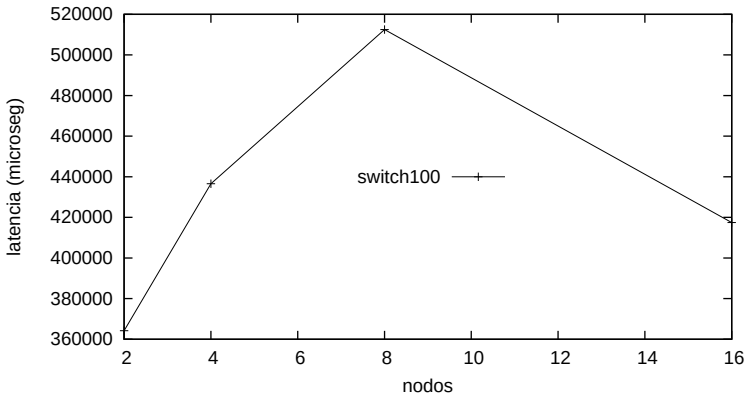


Figura 4.58: Bcast, comparación para $4MBytes$



En este ejemplo, en el que la latencia no varía linealmente con el tamaño

de envío, no se puede generalizar a partir de datos particulares, correspondientes a un determinado tamaño de mensaje. Las gráficas anteriones (4.56 a 4.58) posiblemente variarían de forma considerable al considerar otro tamaño de mensaje.

Es de destacar que la implementación de la función *MPI-Bcast* no aprovecha la capacidad de broadcast de las redes ethernet. Esto se deduce al observar que el número de nodos influye en la latencia.

4.2.10. Reduce

Este ejemplo es similar a *bcast* en el hecho de presentar comunicaciones tanto paralelas como secuenciales.

Figura 4.59: Switch-100$Mbps$/Reduce

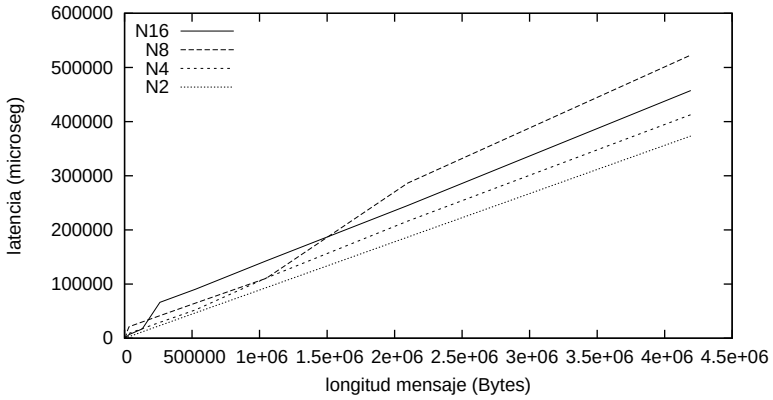


Figura 4.60: Switch-10$Mbps$/Reduce

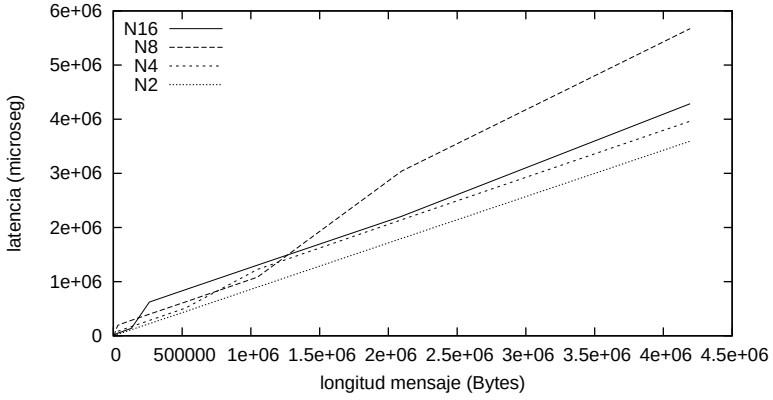


Figura 4.61: Hub/Reduce

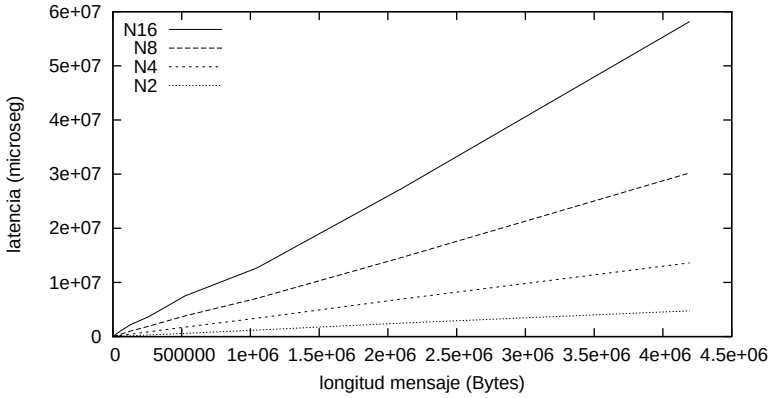


Figura 4.62: Wifi/Reduce

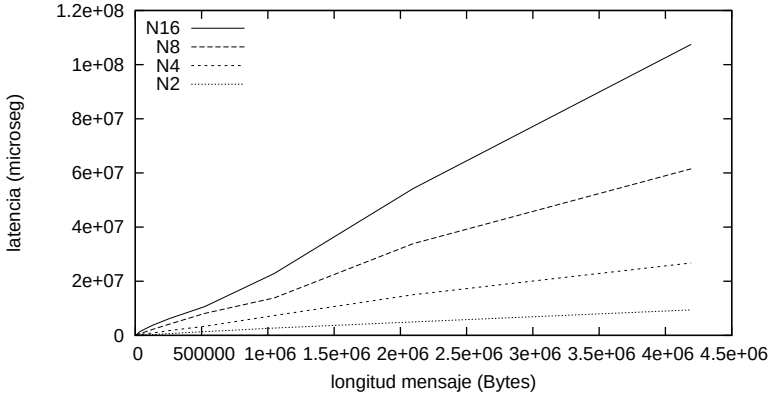


Figura 4.63: Reduce, comparación para $4MBytes$

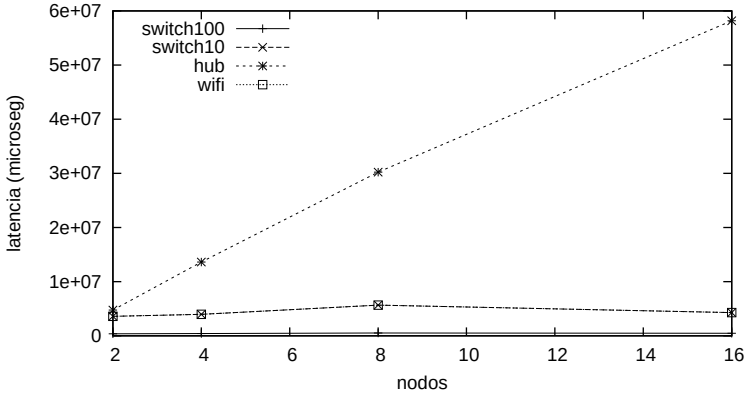


Figura 4.64: Reduce, comparación para $4MBytes$

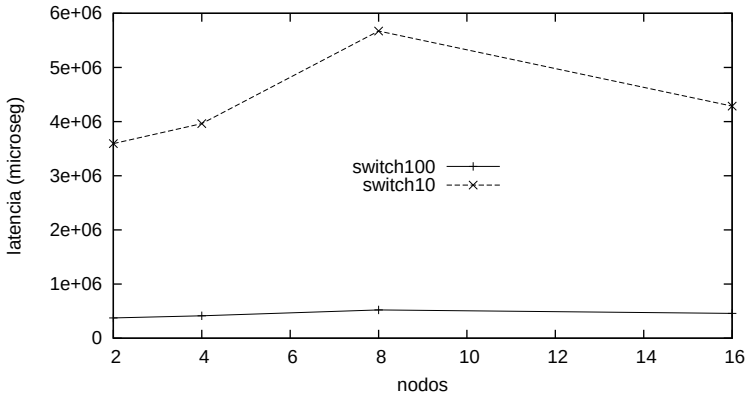
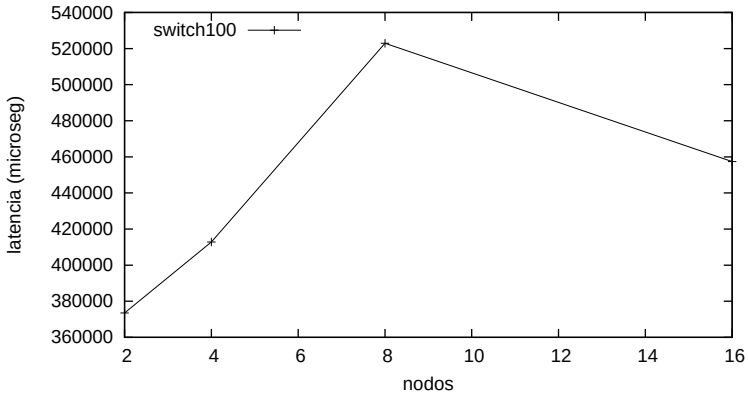


Figura 4.65: Reduce, comparación para $4MBytes$



Nuevamente los switch presentan, ambos, la misma forma de gráfica que vendrá determinada por la forma en que está hecha la implementación de la función *reduce*. Muy posiblemente en forma de árbol, de forma similar a la función *bcast* (sección 4.2.9).

Se hace notar que en este caso se observa un peor rendimiento para 8 nodos, igual que sucedía en la función *bcast*.

4.2.11. ReduceScatter

La función *MPI-Reducescatter* lleva a cabo una operación *reduce* seguida de otra *scatter*. Como es de esperar, se observa que las gráficas de esta función son una combinación de las dos operaciones de que consta. Los tiempos mostrados en estas gráficas resultan de sumar los tiempos correspondientes a las gráficas de las funciones *Scatter* (4.2.8) y *Reduce* (4.2.10), teniendo en cuenta que cuando se ejecuta Reducescatter para un tamaño de mensaje x

determinado, se ejecuta Reduce para un tamaño x y Scatter para un tamaño $\frac{x}{N}$, siendo N el número de nodos.

Figura 4.66: Switch-100$Mbps$/ReduceScatter

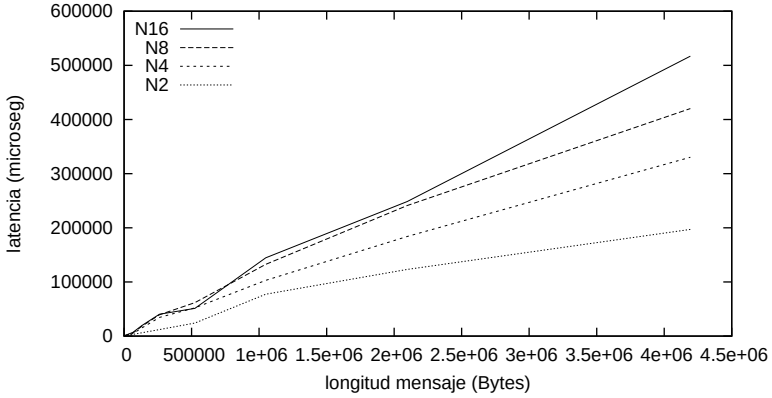


Figura 4.67: Switch-10$Mbps$/ReduceScatter

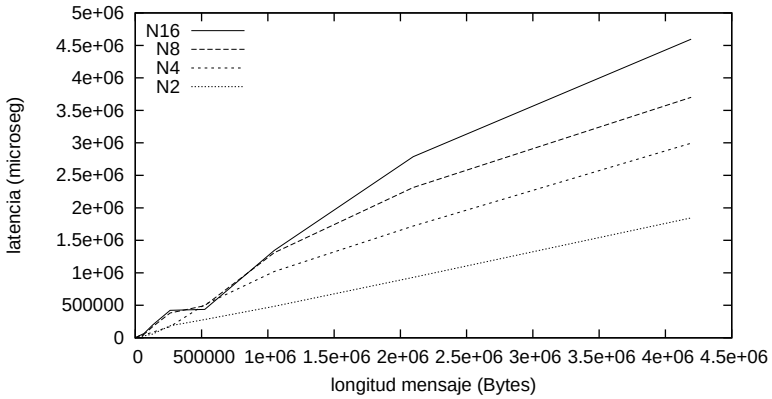


Figura 4.68: Hub/ReduceScatter

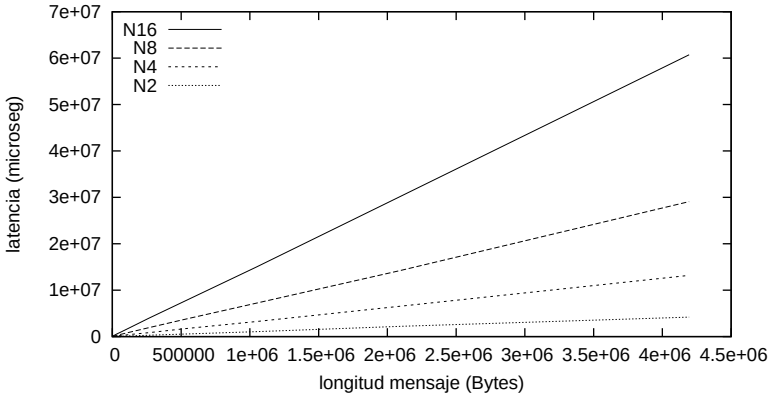


Figura 4.69: Wifi/ReduceScatter

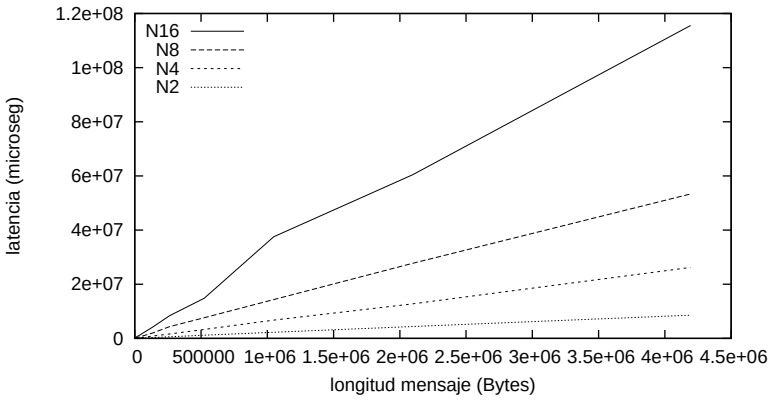


Figura 4.70: ReduceScatter, comparación para $4MBytes$

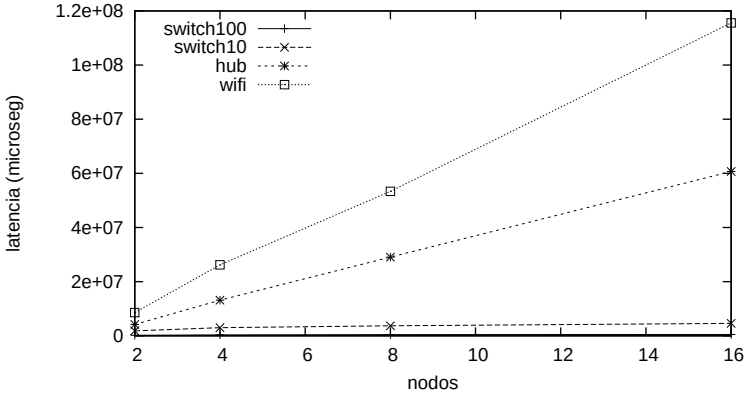


Figura 4.71: ReduceScatter, comparación para $4MBytes$

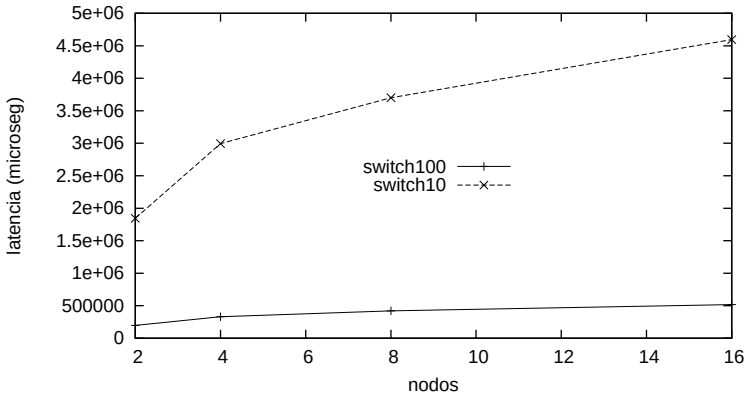
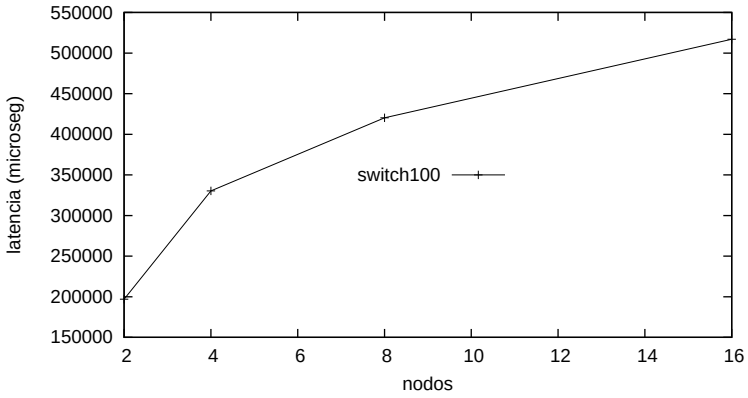


Figura 4.72: ReduceScatter, comparación para $4MBytes$



4.2.12. Alltoall

En secciones anteriores hemos comprobado que cuando las comunicaciones generadas en la ejecución de una determinada función se hacen de forma secuencial (como ocurre siempre en la red conectada mediante el hub y en la red wifi), la latencia es proporcional al número de mensajes que se envían. Dado que en todos los ejemplos vistos hasta ahora los mensajes enviados coincidían con el número de nodos, se hablaba, de forma equivalente, de que la latencia era proporcional al número de nodos, cuando en realidad se refería al número de mensajes enviados. En este ejemplo, tendremos presente que en la ejecución de la función *Alltoall* el número de mensajes enviados no coincide con el de nodos.

En la ejecución de la función *MPI-Alltoall* cada nodo envía un mensaje, en principio diferente, a cada uno de los nodos restantes. Esto supone que si la función es invocada por N nodos, para su ejecución se genera un total de N^2 mensajes.

Figura 4.73: *Alltoall* Switch-100$Mbps$

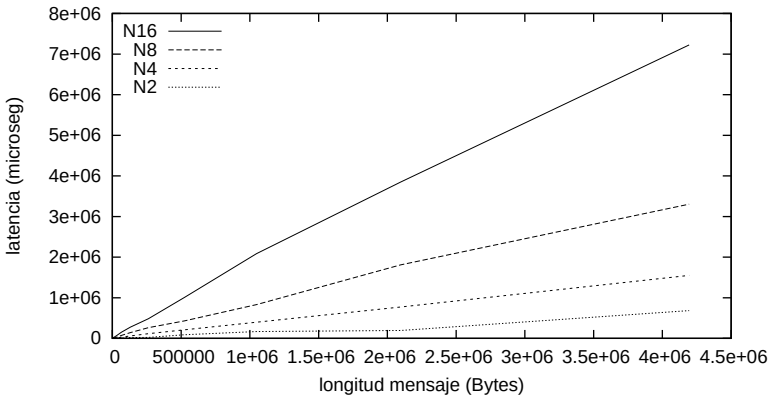


Figura 4.74: *Alltoall* Switch-10$Mbps$

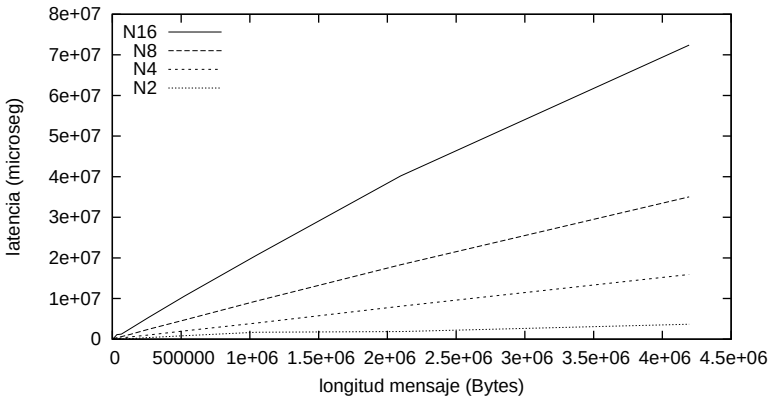


Figura 4.75: *Alltoall* Hub

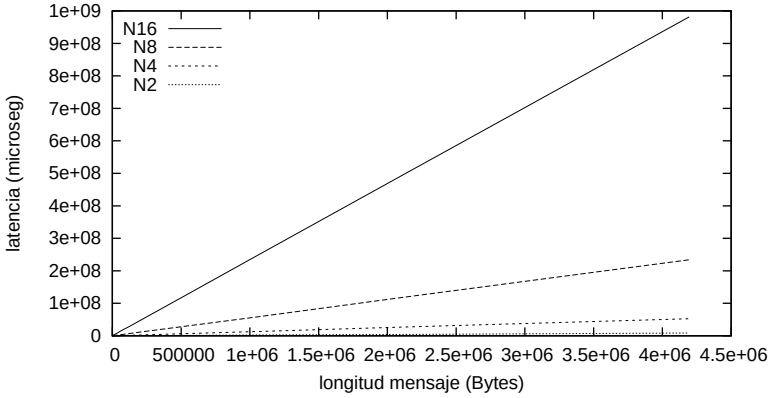
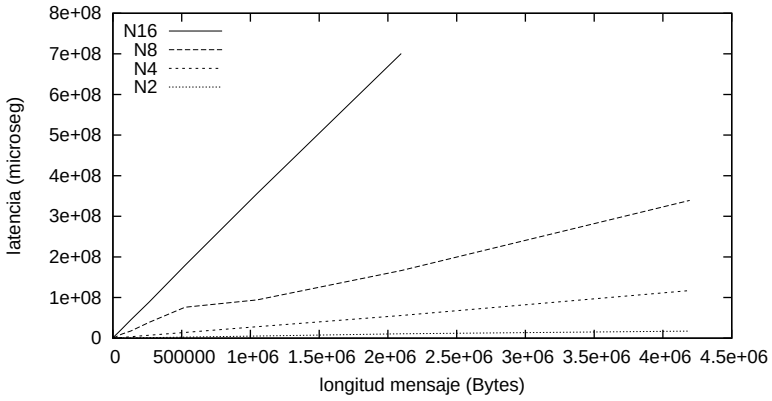


Figura 4.76: *Alltoall* Wifi



Si observamos la gráfica 4.75, correspondiente a la red conectada mediante un hub, comprobamos que la latencia no se duplica al duplicar el número de nodos como había ocurrido en todos los ejemplos anteriores. Esto es debido, como ya se ha comentado, a que la latencia es proporcional al número de envíos, que en este caso crece de forma cuadrática respecto al número de nodos. Así, al duplicar el número de nodos, se cuadruplicará la latencia, hecho que sí se observa en la gráfica 4.75 y en la correspondiente a la red wifi (gráfica 4.76). Estas redes, que en un principio parecían mostrar un comportamiento diferente a lo visto hasta ahora, siguen siendo por tanto un reflejo del comportamiento secuencial que presentan en las comunicaciones.

En las gráficas 4.73 y 4.74, correspondientes a las redes conectadas mediante un switch, se observa que la latencia es proporcional al número de nodos. Para entender este comportamiento, analizaremos cómo se producen los N^2 envíos de mensajes, para determinar hasta qué punto éstos se producen o no de forma simultánea.

Dado que un nodo no puede enviar mensajes a dos nodos distintos a la vez, los $N-1$ mensajes que un nodo determinado debe enviar al resto sólo pueden hacerse de forma secuencial. Por tanto, el tiempo necesario para la ejecución de la función será, como mínimo $N-1$ veces el tiempo necesario para transmitir el mensaje de un nodo a otro (t_x). Veamos que este tiempo, a parte de ser necesario, es suficiente para completar todos los envíos que la función requiere. Imaginemos los N nodos ordenados a modo de cadena circular y que realizan los envíos de forma ordenada como sigue: en un primer tiempo $t = t_x$ cada nodo n_i (con $1 \leq i \geq N$) transmite al nodo vecino (n_{i+1}). En un tiempo $t = pt_x$ (con $1 \leq p \geq N$) cada nodo n_i transmite al nodo n_{i+p}. De esta forma, dado que los canales son fullduplex, en un tiempo $t = Nt_x$ se habrán realizado todas las comunicaciones pendientes, obteniendo así, una latencia proporcional al número de nodos tal y como muestran las gráficas correspondientes.

Figura 4.77: Alltoall, comparación para 4MBytes

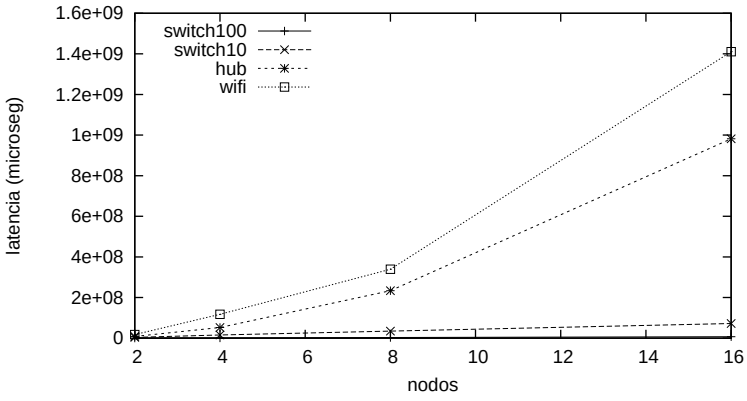


Figura 4.78: Alltoall, comparación para 4MBytes

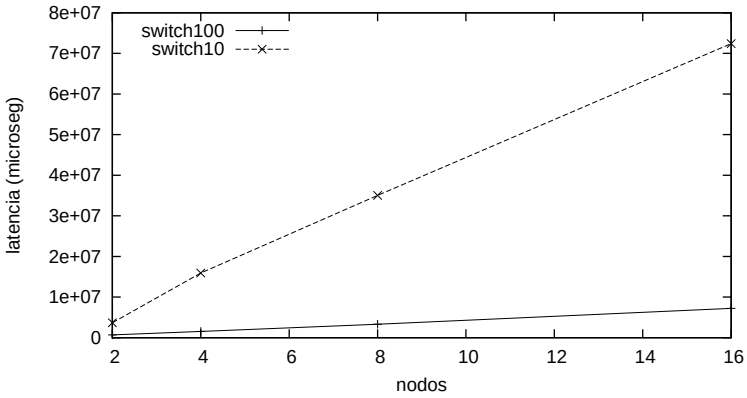
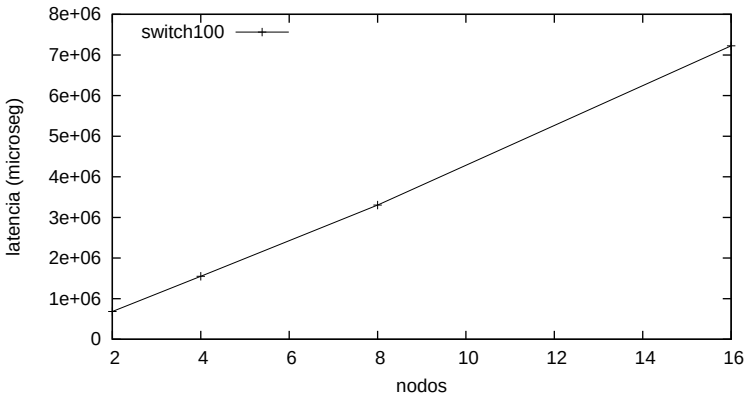


Figura 4.79: Alltoall, comparación para 4MBytes



En las figuras 4.77 a 4.79, puede observarse de forma clara el crecimiento

que presenta la latencia en función del número de nodos. Puede verse que el crecimiento es lineal en el caso de las redes conectadas mediante un switch y que es cuadrático (forma de parábola) en el caso de la red conectada mediante un hub y en la red wifi.

4.2.13. Allgather

Tal como puede observarse en las figuras 4.80 a 4.86, este ejemplo es similar a *Alltoall*. Esto es debido a que ambas funciones coinciden en el número de mensajes que generan y en el modo en que deben llevarse a cabo.

Figura 4.80: *Allgather* Switch-100*Mbps*

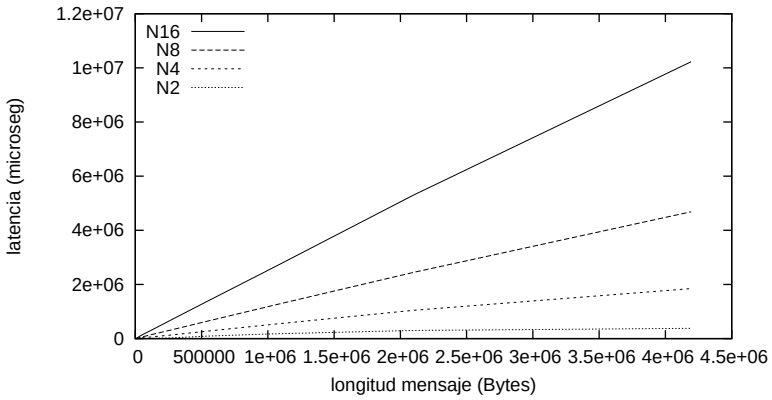


Figura 4.81: *Allgather* Switch-10*Mbps*

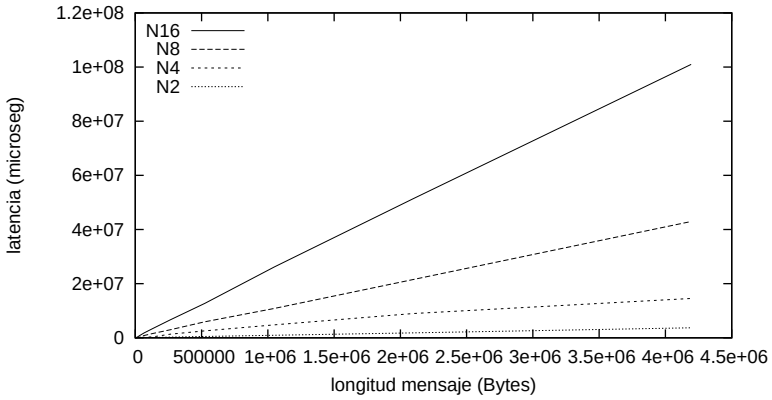


Figura 4.82: *Allgather* Hub

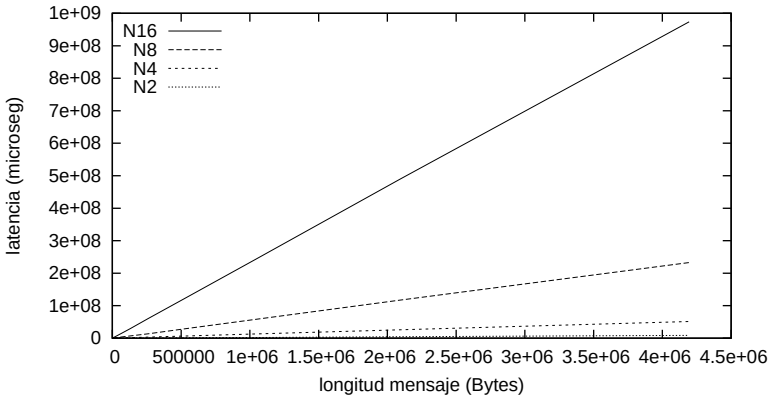


Figura 4.83: *Allgather* Wifi

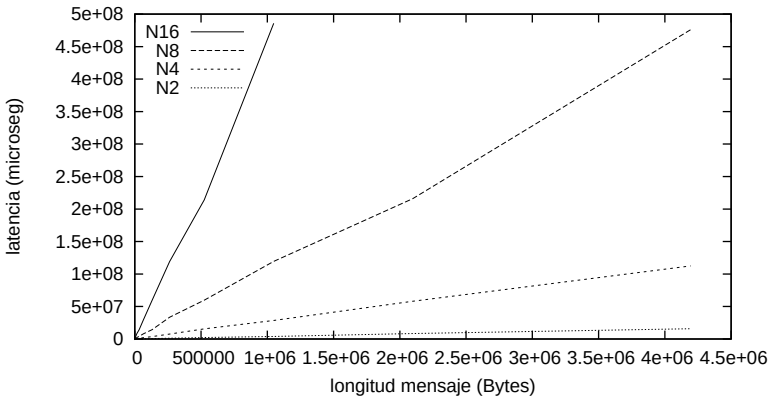


Figura 4.84: Allgather, comparación para $4MBytes$

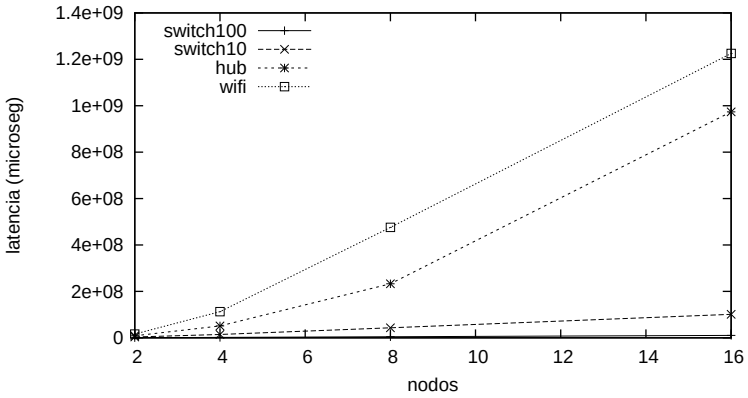


Figura 4.85: Allgather, comparación para $4MBytes$

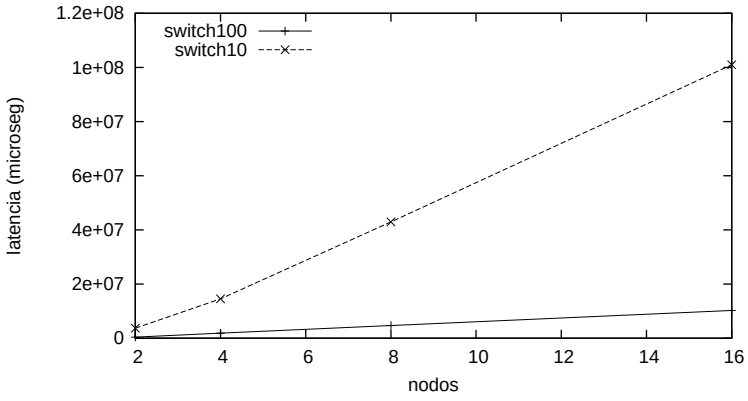
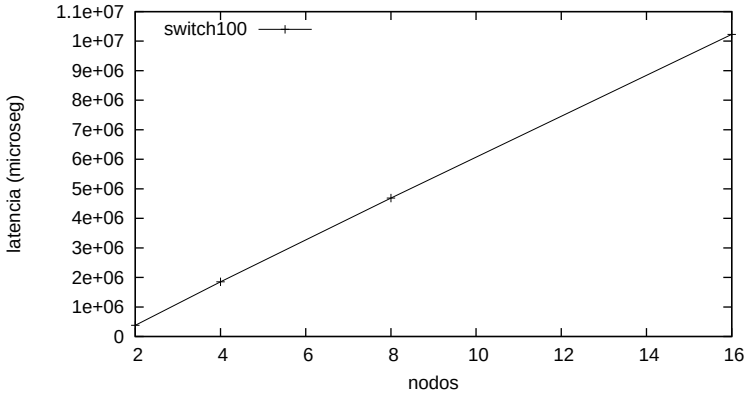


Figura 4.86: Allgather, comparación para $4MBytes$



4.2.14. Conclusiones a los resultados de las pruebas con benchmarks MPI de Intel

La red conectada mediante un hub siempre mantiene constante el *ancho de banda global*, o equivalentemente, el ancho de banda local se divide entre el número de comunicaciones simultáneas. Por otro lado, las redes conectadas mediante un switch multiplican su *ancho de banda global* por el número de comunicaciones simultáneas (siempre que no se supere el número máximo de conexiones simultáneas que permite el switch), o lo que es lo mismo, el ancho de banda local no se ve afectado por el hecho de que haya otras comunicaciones paralelas.

A lo largo de la sección 4.2 hemos podido comprobar que algunas redes obtienen un importante aumento del ancho de banda al ejecutar determinadas funciones *MPI*. Es decir, que dependiendo de la función y de la red concreta sobre la que ésta se ejecuta, el ancho de banda nominal de la red puede verse multiplicado.

A modo de ejemplo de la ganancia que se puede obtener en la ejecución de determinadas funciones *MPI*, la siguiente tabla muestra un resumen del *ancho de banda global (ABG)* obtenido al ejecutar en cada una de las redes disponibles, para 16 nodos, los diferentes benchmarks estudiados. De forma general, el ABG viene dado por la expresión:

$$ABD = \frac{m * x}{t_x}$$

Siendo m el número total de envíos, x el tamaño del mensaje y t_x la latencia asociada al envío de un mensaje de longitud x. Para los cálculos de la tabla 4.17, consideraremos $x = 4Mbytes$.

Tabla 4.17: *ancho de banda global* para 16 nodos

(**Mbps**)	**switch**$100Mbps$	**switch**$10Mbps$	**hub**$10Mbps$	**wifi**
MultiPingPong	752	75,2	8,60	8,22
MultiPingPing	1375	146,2	8,51	8,8
Sendrecv	1228	134,9	8,33	4,5
Gather	96,34	9,99	7,67	4,42
Scatter	120,35	16,7	13,4	4,23
Bcast	1205	119,03	8,85	5,38
Reduce	1100	117,44	8,65	4,68
Reduce-scatter	1034	116,37	8,20	4,62
Alltoall	1114	111,22	8,20	5,7
Allgather	787	79,75	8,26	6,57

4.3. Aplicaciones

En la sección 4.2 se ha visto que las características de la red utilizada para la interconexión de los PCs determina, en gran medida, la latencia en la ejecución de las distintas funciones MPI. En esta sección se quiere comprobar hasta qué punto la diferencia de comportamiento observada afecta a la ejecución de aplicaciones paralelas reales. Para ello se miden tiempos de ejecución de varias aplicaciones reales, que se ejecutan en paralelo sobre cada una de las redes consideradas, y para un número de nodos variable. Mediante la computación en paralelo, se busca reducir el tiempo de ejecución de aplicaciones que precisan enormes cantidades de operaciones. Al repartir el trabajo entre distintos procesadores se consigue reducir el tiempo de cómputo en cada uno de ellos. Esto no supone necesariamente una reducción del tiempo de ejecución, ya que al tiempo que los procesadores necesitan para realizar las operaciones propias de la aplicación, se le añade el tiempo que éstos van a emplear en comunicarse.

Con el fin de facilitar el análisis de los resultados obtenidos en las distintas pruebas realizadas, se hacen las siguientes suposiciones:

1. El tiempo que la aplicación necesita para efectuar sus operaciones (tiempo de cómputo C_1) es el tiempo que se obtiene al ejecutar la aplicación para un nodo ($N = 1$).

2. Al ejecutar la aplicación en n nodos, el tiempo de computo en uno de los nodos (C_n) viene dado por la expresión $C_n = \frac{C_1}{n}$, es decir el tiempo de cómputo se reparte entre los distintos nodos.

3. Estimamos el tiempo empleado en comunicación, como la diferencia entre el tiempo de ejecución y el de cómputo. Esta estimación se refiere al tiempo mínimo de comunicación, ya que, en algunos casos estos tiempos podrían solaparse, con lo que el tiempo de comunicación real sería aún mayor.

Estas suposiciones nos permiten desglosar el tiempo de ejecución en tiempo de cómputo y tiempo de comunicación entre los distintos nodos. De esta forma, podremos evaluar la sobrecarga que supone la comunicación en el tiempo de ejecución.

Como hemos visto en la sección 4.2, hay varios factores que influyen en la latencia de las funciones *MPI*. Estos pueden resumirse en:

1. Características de la red. Principalmente, ancho de banda y si permite comunicaciones simultáneas.

2. Tipo de comunicación. Por ejemplo, ante una comunicación secuencial, no habrá diferencia entre la red que permite las comunicaciones paralelas de aquella que no las permite.

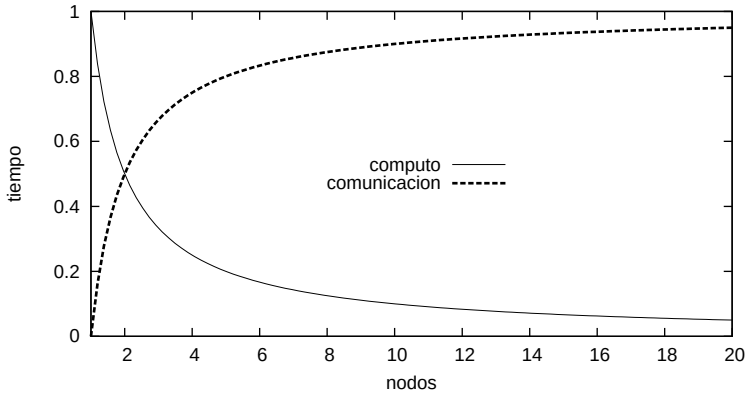
3. Tamaño del mensaje a enviar. En la seción 4.2 se ha podido comprobar que para cualquiera de las funciones estudiadas, independientemente de la red o del número de nodos, la latencia aumenta con el tamaño del mensaje a enviar.

Considerando una red concreta, para analizar la evolución que la latencia de comunicación presenta al aumentar el número de nodos, tendremos en cuenta, por un lado, que la latencia puede aumentar (o no) debido al aumento del número de comunicaciones. Por otro, que la latencia podría disminuir si el aumento de nodos conlleva la reducción del tamaño de los mensajes. Como consecuencia, el aumento o no de la latencia de comunicación, dependerá, en gran medida, de las características de la aplicación.

El objetivo de la comunicación paralela es reducir el tiempo de ejecución de las aplicaciones. Esto se producirá si la disminución en el tiempo de cómputo puede compensar el posible aumento de la latencia de comunicaciones. En la figura 4.87 se muestra, por un lado, el modo en que disminuye el

tiempo de ejecución al aumentar el número de nodos, y por otro, el umbral máximo para la latencia de comunicación, que no deberá ser sobrepasado para obtener tiempos inferiores a la ejecución en un único nodo.

Figura 4.87: Evolución del tiempo de cómputo y del umbral de latencia de comunicación



Las aplicaciones que hemos utilizado para el presente estudio son las siguientes:

1. Gromacs
2. Lammps
3. NBP
4. Multiplicación matricial

4.3.1. Gromacs

Para esta aplicación se han considerado dos ejemplos (dppc y poly) de distinto tamaño.

Gromacs-dppc Este es el ejemplo de aplicación paralela real de mayor tamaño que se ha utilizado. Los resultados obtenidos con esta aplicación se muestran en las figuras 4.88 a 4.90.

Figura 4.88: *gromacs_ dppc*

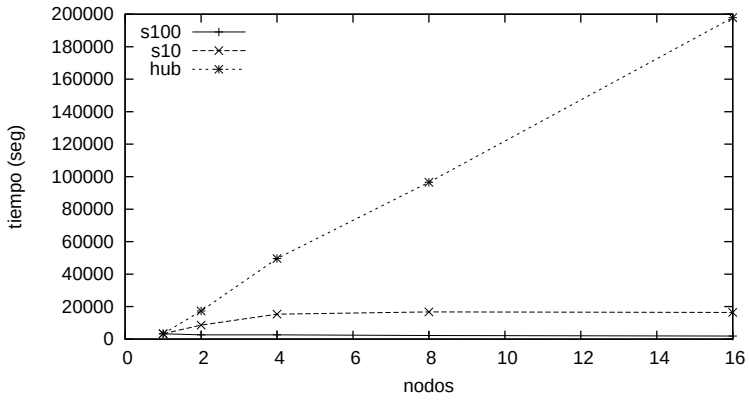


Figura 4.89: *gromacs_ dppc*

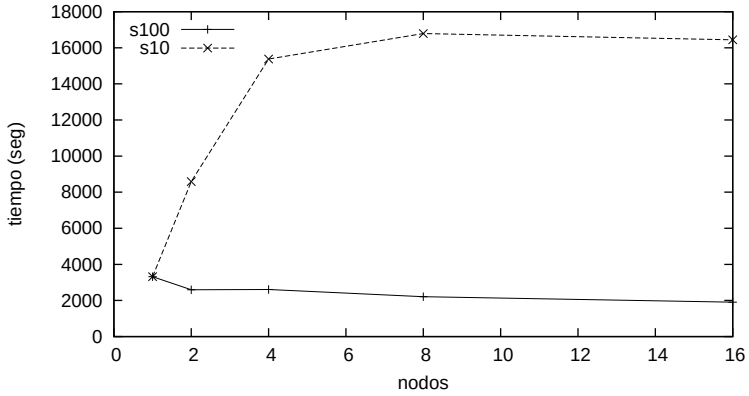
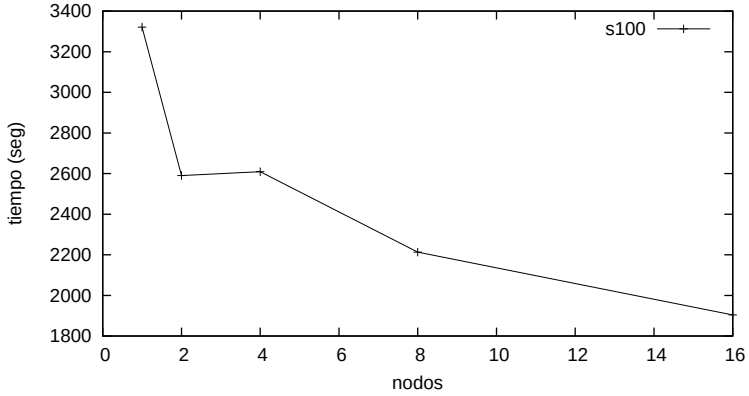


Figura 4.90: *gromacs_ dppc*



A la vista de estos primeros datos de ejecución de aplicaciones reales, se

pone de manifiesto la importancia que tiene la red utilizada para la interconexión de los nodos empleados para la ejecución en paralelo. Se comprueba que una mala elección en la red de interconexión puede hacer que no se consiga el objetivo deseado de reducir el tiempo de ejecución. Comprobamos que este aumento de tiempo de ejecución puede llegar a ser considerablemente mayor al tiempo de ejecución en un único nodo. Por ejemplo, al ejecutar *gromacs-dccp*, con 16 nodos, en la red conectada con un hub, obtenemos un tiempo de ejecución 60 veces mayor que el tiempo de ejecución en un nodo.

Consideramos que el tiempo de cómputo de la aplicación es $3320seg$ (ejecución para un nodo). A partir de este dato hacemos una estimación de la latencia de comunicación que presenta cada una de las redes, al ejecutar esta aplicación. Los resultados se muestran en la tabla 4.18

Tabla 4.18: gromacs_dppc latencia de comunicación (seg)

nº nodos	**1**	**2**	**4**	**8**	**16**
switch-$100Mbps$	0	930	1779	1798	1697
switch-$10Mbps$	0	6925	14549	16372	16237
hub	0	15658	47960	95282	197680

Puede observarse que en las redes conectadas mediante un switch la latencia de comunicación se duplica al pasar de 2 a 4 nodos mientras que se mantiene constante con 4, 8 y 16. Por otro lado, la red conectada mediante un hub triplica su latencia de comunicación al pasar de 2 a 4 nodos y se va duplicando al pasar de 4 a 8 y de 8 a 16. Este comportamiento vendrá determinado por la propia implementación de la aplicación. Como ya se comentó al principio de la sección 4.3, dependiendo de la implementación de cada aplicación paralela, y de las características de la red utilizada, la latencia de comunicación podía aumentar, mantenerse constante o incluso disminuir a medida que aumenta el número de nodos.

En las figuras 4.91 a 4.93, se representa, para cada una de las redes, el tiempo de latencia de comunicación, con la referencia del umbral de comunicación definido al inicio de esta sección.

Figura 4.91: *gromacs-dppc* latencia de comunicación

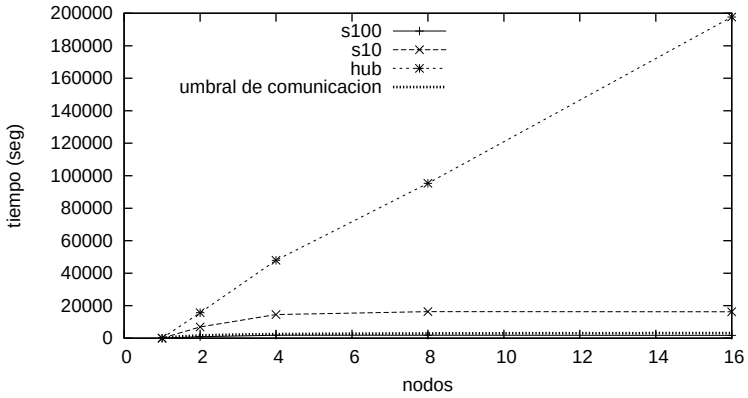


Figura 4.92: *gromacs-dppc* latencia comunicación

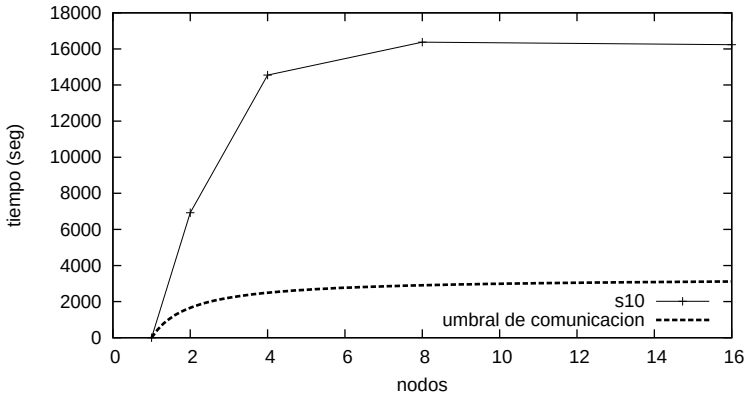
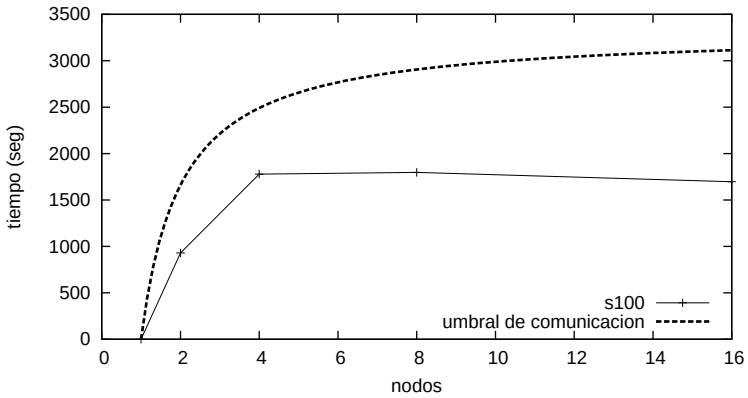


Figura 4.93: *gromacs-dppc* latencia comunicación



La gráfica 4.92 muestra un ejemplo en el que se supera el umbral máximo

de comunicación, mientras que la gráfica 4.93 muestra un ejemplo en el que la latencia de comunicación se mantiene por debajo del umbral. El comportamiento observado en estas gráficas coincide con los datos que se muestran en las gráficas 4.89 y 4.90 en las que se puede comprobar que para la red conectada mediante el switch de $100Mbps$ mejoramos el tiempo de ejecución a medida que aumenta el número de nodos, mientras que en la red del switch de $10Mbps$ este tiempo empeora.

Gromacs-poly Ante los elevados tiempos de ejecución obtenidos en la aplicación gromacs-dppc, se plantean pruebas con un problema de menor tamaño que permita evaluar la red wifi. Los resultados obtenidos con esta aplicación se muestran en las figuras 4.94 a 4.96.

Figura 4.94: *gromacs_poly*

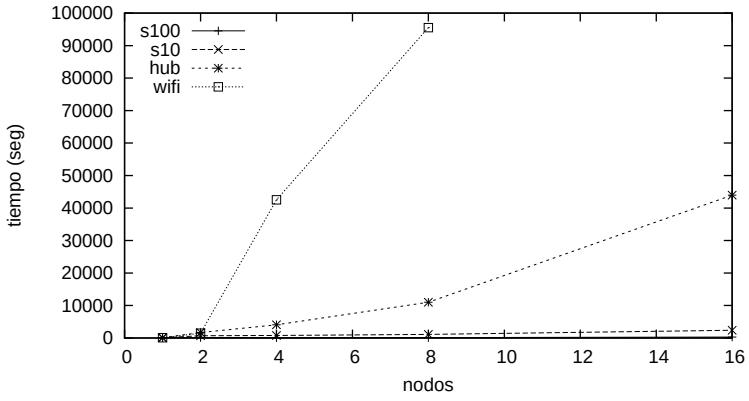


Figura 4.95: *gromacs_poly*

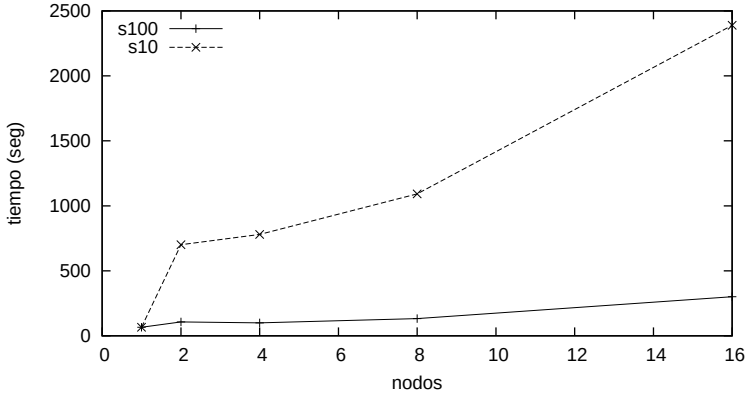
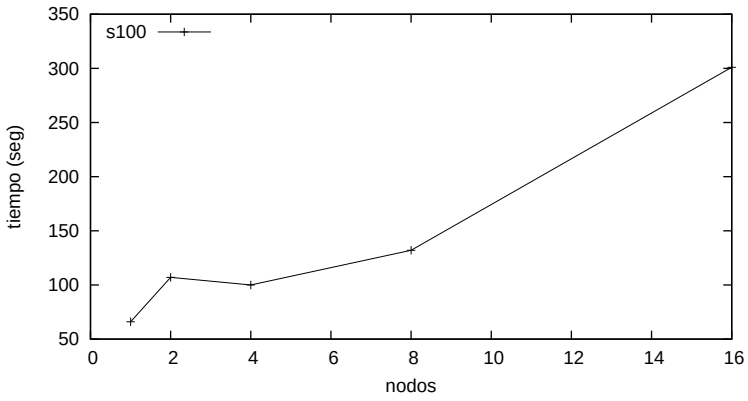


Figura 4.96: *gromacs_poly*



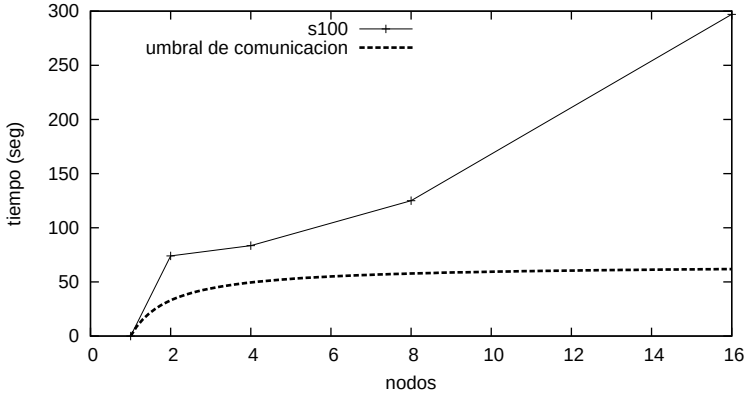
Tiempo de computo $C1 = 66seg$

Tabla 4.19: *gromacs_poly* latencia de comunicación

nº nodos (N)	**1**	**2**	**4**	**8**	**16**
switch-$100Mbps$	0	74	83.5	125	297
switch-$10Mbps$	0	668	763	1084	2385
hub	0	1668	4055	10993	43967
wifi	0	1461	42526	95511	

En este ejemplo podemos observar que, en ninguna de las redes se mejora el tiempo de ejecución de un nodo. La sobrecarga de comunicación es muy elevada. Veamos, por ejemplo (figura 4.97), cómo la latencia de comunicación en la red del switch de $100Mbps$ está por encima del umbral de comunicación:

Figura 4.97: *gromacs-poly* latencia de comunicación



A la vista de los resultados obtenidos, podemos concluir que las redes que

habitualmente se utilizan para la ejecución de aplicaciones como ésta, tienen un mayor ancho de banda que cualquiera de las redes que consideramos.

Lammps

En el caso de esta otra aplicación paralela real, se ha escogido un ejemplo de tamaño pequeño como es *Lammps-crack*. Los resultados obtenidos con esta aplicación se muestran en las figuras 4.98 a 4.100.

Figura 4.98: *lammps_ crack*

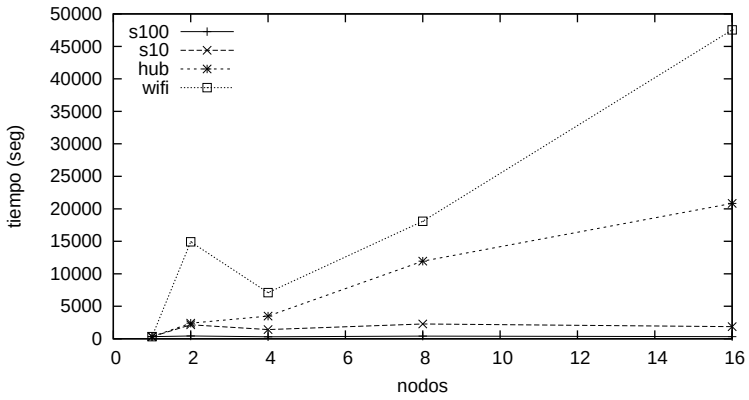


Figura 4.99: *lammps_ crack*

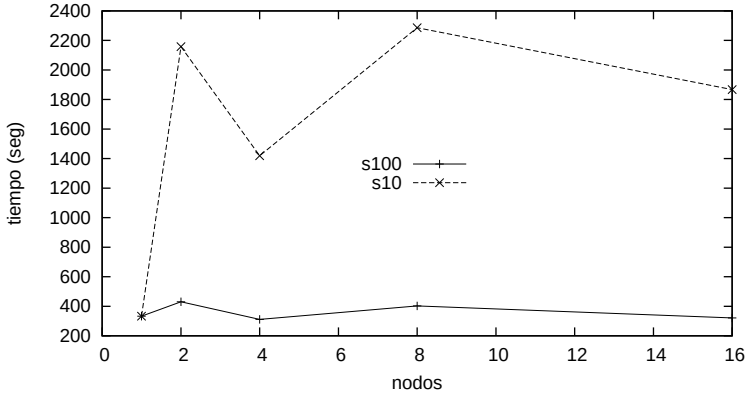
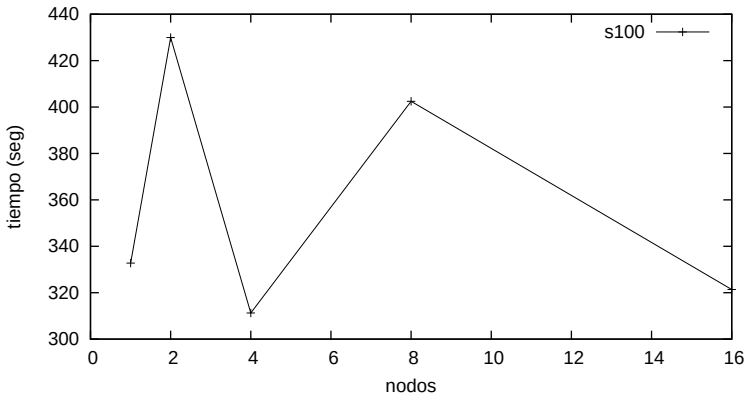


Figura 4.100: lammps_crack



Tiempo estimado de cómputo para un nodo: $C_1 = 332seg$

Tabla 4.20: lammps_crack. latencia de comunicación

nº nodos (N)	**2**	**4**	**8**	**16**
switch-$100Mbps$	263	288	360	300
switch-$10Mbps$	1992	1336	2244	1846
hub	2233	3414	11904	20803
wifi	14765	7028	18051	47521

Figura 4.101: lammps_crack. latencia en comunicación

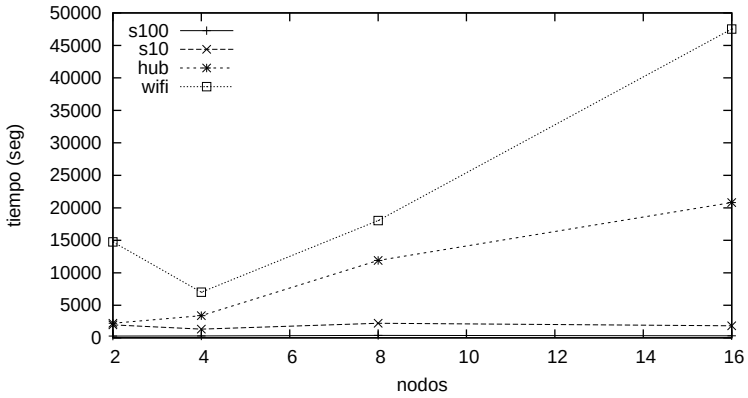


Figura 4.102: lammps_crack. latencia en comunicación

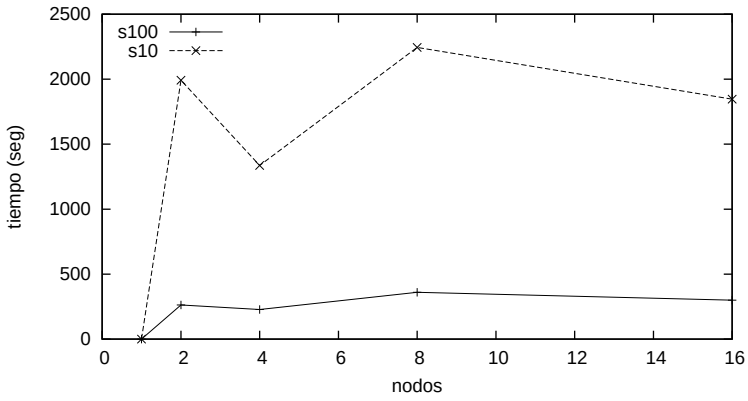
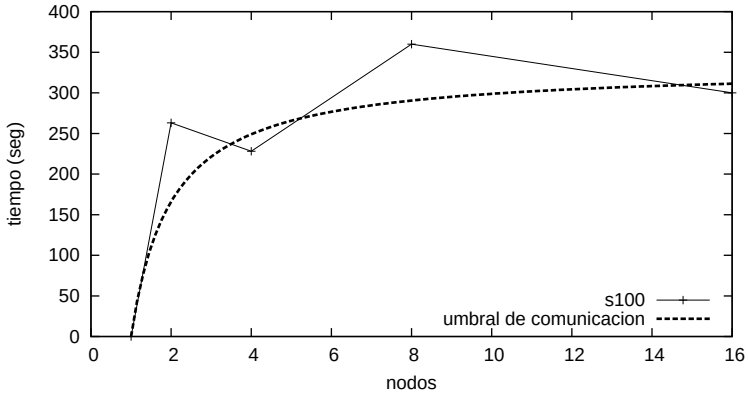


Figura 4.103: lammps_crack. latencia en comunicación



En la gráfica 4.103, al igual que en la figura 4.100, se comprueba que, con 4 y 16 nodos, se reduce el tiempo de ejecución obtenido para un nodo.

Como ya se ha comentado anteriormente, dependiendo de la implementación de la aplicación paralela, y de las características de la red utilizada, la latencia de comunicación puede aumentar, mantenerse constante o incluso disminuir a medida que aumenta el número de nodos. Éste es un ejemplo en el que la latencia de comunicación aumenta o disminuye en función del número de nodos. A la vista de las gráficas 4.101 a 4.103, podríamos deducir que la implementación de la aplicación *lammps-crack* hace que la latencia de comunicación sea notablemente menor cuando el número de nodos es cuadrado perfecto.

4.3.2. NPB

Los ejemplos que se estudian en esta sección no son aplicaciones paralelas reales, aunque el código que contienen forma parte de gran número de ellas.

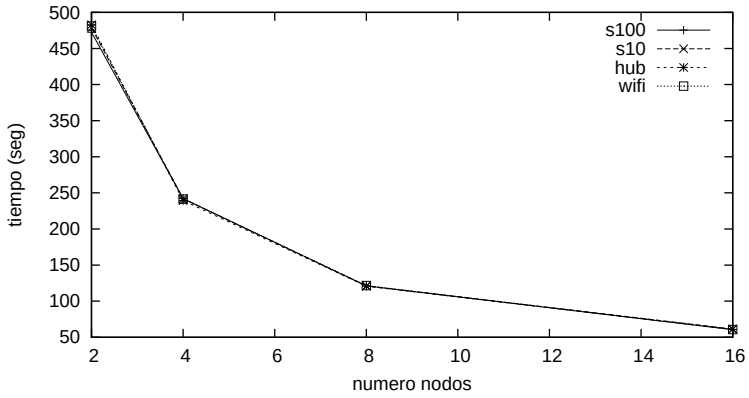
Los resultados obtenidos en la ejecución de aplicaciones paralelas reales parecen revelar que las redes consideradas para el estudio no son aptas para la computación paralela. Sólo en uno de los ejemplos y para una de las redes (figura 4.90) se conseguía mejorar el tiempo de ejecución al aumentar el número de nodos.

Esto nos lleva a buscar nuevos ejemplos que nos permitan estudiar si es posible conseguir mayor escalabilidad, y bajo qué condiciones. En concreto, se ejecutan los sigientes benchmarks NAS: EP y FT.

EP Ejemplo de función *altamente paralela.*

Se ha escogido esta función con muy poca comunicación, con el fin de confirmar que en tal caso, independientemente de la red utilizada, se consigue disminuir el tiempo de ejecución al aumentar el número de nodos. Se ejecuta el ejemplo de mayor tamaño (C) dentro de los disponibles para esta función.

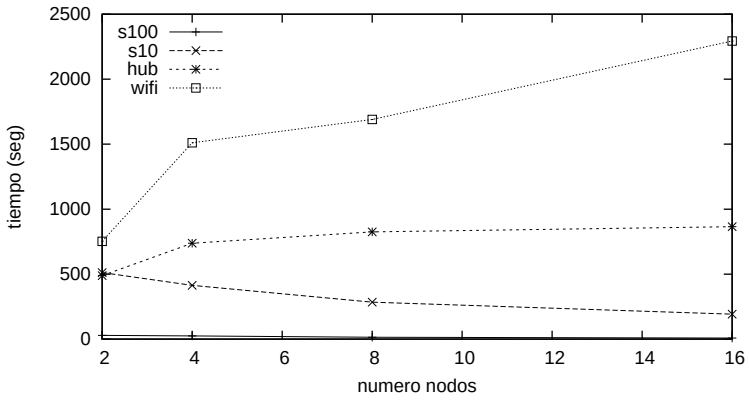
Figura 4.104: benchmarks-NAS_epC



Con este ejemplo (figura 4.104) comprobamos que, efectivamente, la red utilizada no ha influido en el tiempo de ejecución debido a la escasa comunicación. Puede observarse que las curvas correspondientes a todas redes coinciden entre sí y con la del tiempo de cómputo, que se mostró en la figura 4.87.

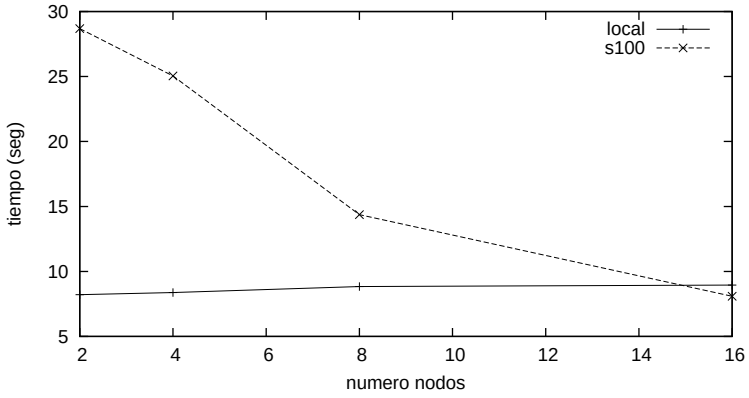
FT Ejemplo de función utilizada para cálculo paralelo de la transformada de Fourier. En este caso se ha optado por el de menor tamaño disponible (A).

Figura 4.105: benchmarks-NAS_ftA



Dado que no se dispone de tiempo de ejecución para un nodo (este ejemplo sólo puede ejecutarse para un número de nodos que sea potencia de 2), para saber si se mejora el tiempo de ejecución en un único nodo, se recurre a comparar con los tiempos obtenidos al ejecutar los distintos procesos en una única máquina (figura 4.105). Como los procesadores utilizados tienen dos cores, los tiempos obtenidos serán, aproximadamente, la mitad de lo que se obtendría si sólo tuvieran uno.

Figura 4.106: benchmarks-NAS_ftA



Los resultados mostrados en la gráfica 4.106 indican que para 8 y 16 nodos se mejora el tiempo de ejecución de un nodo (recordemos que los tiempos representados en la curva *local* son la mitad de los que se obtendrían con un sólo core).

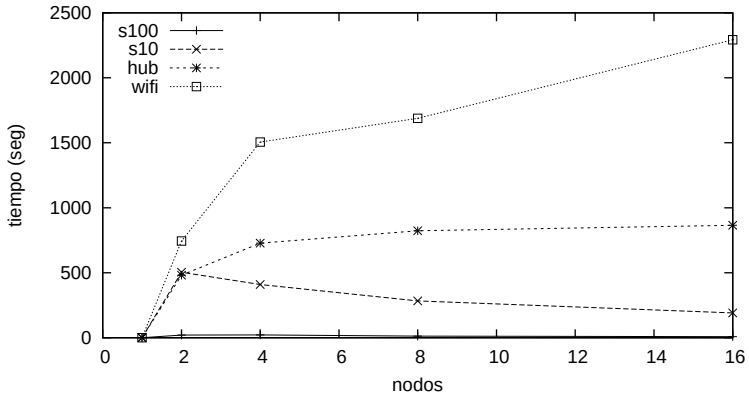
Con el fin de completar una tabla de sobrecarga por comunicación (tabla 4.21), de forma similar a otras anteriores (por ejemplo las tablas 4.20 y 4.18), estimamos el tiempo de ejecución en un nodo $C_1 = 16{,}24seg$, a partir del obtenido para dos procesos en un único nodo:

Tabla 4.21: NAS_ftA latencia de comunicación

nº nodos (N)	**2**	**4**	**8**	**16**
switch-100$Mbps$	20.6	21.04	12.37	7.95
switch-10$Mbps$	504	409	283	191
hub	481	728	823	865
wifi	744	1506	1688	2293

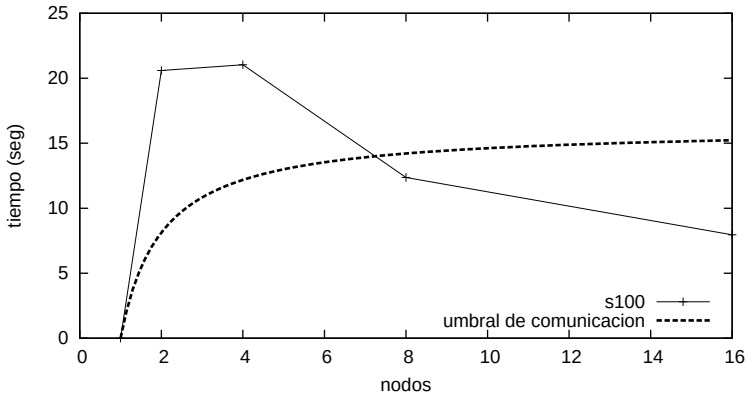
En este ejemplo, observamos (tabla 4.21) que para les redes conectadas mediante un switch, la latencia debida a la comunicación decrece a medida que aumenta el número de nodos. Esto es posible ya que al aumentar el número de nodos, aunque posiblemente aumentará el número de comunicaciones, también disminuirá el tamaño de éstas.

Figura 4.107: NBP-ftA latencia en comunicación



Si comparamos la gráficas 4.107 y 4.105, comprobamos que cuando el tiempo de comunicación es considerablemente mayor que el tiempo de cómputo, coinciden las gráficas de tiempo de ejecución y comunicación.

Figura 4.108: NBP-ftA latencia en comunicación



En la figura 4.108 se observa nuevamente el hecho de que para 8 y 16 nodos se mejora el tiempo de ejecución de un nodo.

4.3.3. Multiplicación de matrices

El motivo de utilizar esta aplicación es disponer de un ejemplo de aplicación real que tenga, por un lado, una implementación conocida, que nos permita saber el tipo y número de comunicaciones que realiza, y por otro lado, que no tenga demasiada comunicación (en relación al tiempo de cómputo), de modo que el tiempo de ejecución disminuya con el número de nodos.

En las pruebas realizadas, se ha considerado la multiplicación de matrices cuadradas, de tamaño M. Dado que buscamos el menor número de comunicaciones posibles, se ha adaptado el algoritmo utilizado, de modo que las dos matrices son la misma (cálculo del cuadrado de una matriz). La multiplicación se realiza considerando que la matriz está distribuida entre los nodos, en bloques.

Matriz 1000x1000 En este primer ejemplo se considera una matriz de tamaño 1000x1000. Los resultados obtenidos en la ejecución de esta aplicación se muestran en las gráficas 4.109 y 4.110.

Figura 4.109: Multiplicación Matriz 1000x1000

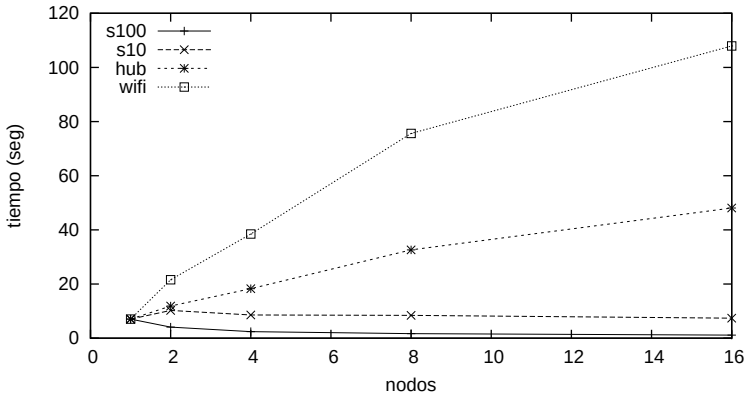
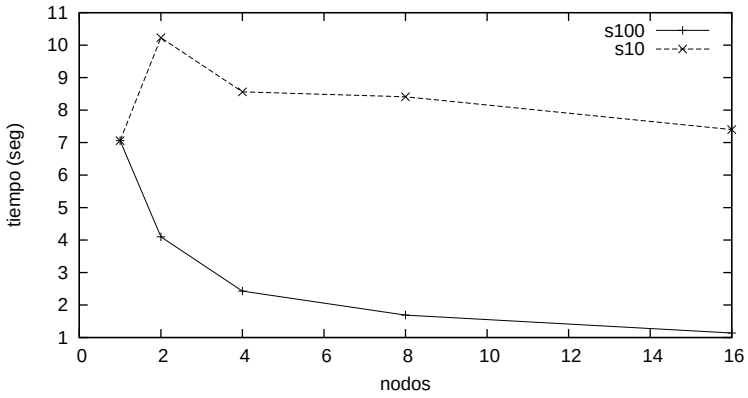


Figura 4.110: Multiplicación Matriz 1000x1000



Se puede observar que al utilizar la red conectada mediante un switch de $100Mbps$, obtenemos los resultados buscados. Es decir, el tiempo de ejecución se reduce a medida que aumentamos el número de nodos. Con el resto de redes no se obtiene el resultado deseado, por lo que se plantean nuevas pruebas considerando mayores tamaños de matriz.

Antes de ver resultados para otros tamaños de matriz, calcularemos la latencia de comunicación, como se ha hecho ya para otras aplicaciones. Después, haremos un análisis de las comunicaciones que se llevan a cabo al ejecutar el algoritmo de multiplicación y calcularemos (utilizando los tiempos obtenidos con IMB) una estimación del tiempo necesario para realizar las comunicaciones que el algoritmo precisa. Finalmente compararemos los dos tiempos de comunicación obtenidos.

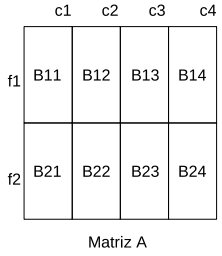
Cálculo de la latencia de comunicación. Se considera el tiempo de cómputo $C_1 = 6{,}83$. Como se ha hecho ya en otros ejemplos, se calcula la latencia de comunicación como la diferencia entre el tiempo de ejecución

y el de cómputo. Los resultados obtenidos para cada una de las redes, se muestran en la tabla 4.22

Tabla 4.22: Matriz-1000x1000 latencia de comunicación

nº nodos	**2**	**4**	**8**	**16**
switch-$100Mbps$	0.69	0.72	0.84	0.71
switch-$10Mbps$	6.82	6.85	7.56	6.97
hub	8.46	16.6	31.8	47.7
wifi	18.2	36.8	74.8	107.5

Análisis de las comunicaciones. La matriz se considera dividida en n bloques y distribuida entre los nodos. Por ejemplo, al ejecutar la multiplicación en $n = 8$ nodos, cada uno de ellos tendrá uno de los bloques B_{ij} siguientes:



Del análisis del algoritmo de la multiplicación considerado se desprende que, para cada fila, cada uno de sus nodos realiza MPI_Bcast para enviar los datos del bloque que tiene, al resto de nodos de su misma fila. Las distintas comunicaciones de $Bcast$ entre los nodos de una fila se llevan a cabo de forma secuencial, ya que entre cada dos envíos $Bcast$ de una misma fila, debe realizarse cómputo. Por otro lado, envíos correspondientes a filas distintas podrían realizarse, si la red lo permite, de forma simultánea. Por tanto, si una matriz está distribuida en $n = pq$ nodos, por cada fila se realizarán, de forma secuencial, q operaciones de $Bcast$ con q nodos (lo denotamos $Bcast_q$), y dado que hay p filas, hay p grupos ejecutando $Bcast_q$, lo denotamos $multi_p Bcast_q$. Por otro lado si el tamaño de la matriz es M, el tamaño de bloque será $\frac{M}{n}$. Dado que lo dicho para filas se repite, de forma análoga, para columnas, podemos resumir las comunicaciones que se realizan en la siguiente expresión:

$$\underbrace{q \times multi_p Bcast_q(\frac{M}{n})}_{filas} + \underbrace{p \times multi_q Bcast_p(\frac{M}{n})}_{columnas} \tag{4.4}$$

El estudio realizado con el benchmark Multi-PingPong (4.2.3) nos sirve ahora para saber el comportamiento que tendrá la operación $multi_q Bcast_p$ descrita, dependiendo de si la red sobre la que se ejecuta permite comunicaciones simultáneas o no:

$$multi_q Bcast_p(\frac{M}{n}) = \begin{cases} Bcast_p(\frac{M}{n}), & \text{switch} \\ q \times Bcast_p(\frac{M}{n}), & \text{hub y wifi} \end{cases} \quad (4.5)$$

Utilizando por un lado

las expresiones 4.4 y 4.5, y por otro, los tiempos obtenidos en la ejecución del IMB-Bcast (sección 4.2.9), podemos hacer una estimación del tiempo empleado para la comunicación, al ejecutar el algoritmo. Se debe tener en cuenta que al aumentar el número de nodos, el tamaño de mensaje a enviar irá disminuyendo. Por ejemplo, en el caso de una matriz 1000×1000, dado que los datos son números reales en doble precisión, se tiene un tamaño inicial $M = 1000 \times 1000 \times 8Bytes = \frac{8}{1{,}048576} MBytes$, que estará repartido entre los nodos, en bloques de tamaño $\frac{8}{n \times 1{,}048576} MBytes$. Los tiempos obtenidos se muestran en la tabla 4.23.

Tabla 4.23: Estimación de tiempo de comunicación. Matriz-1000x1000

nº nodos	2	4	8	16
switch-100$Mbps$	0.69	0.69	0.85	0.68
switch-10$Mbps$	6.79	6.79	8.41	6.93
hub	8.9	16.1	33.5	51.1
wifi	18.9	34.9	97.9	121.4

En las figuras 4.111 a 4.113 se representan conjuntamente, para su comparación, los datos mostrados en las tablas 4.22 y 4.23, que se corresponden con los tiempos de comunicación obtenidos con los dos métodos descritos.

Figura 4.111: Comparación tiempos comunicación. switch-100$Mbps$

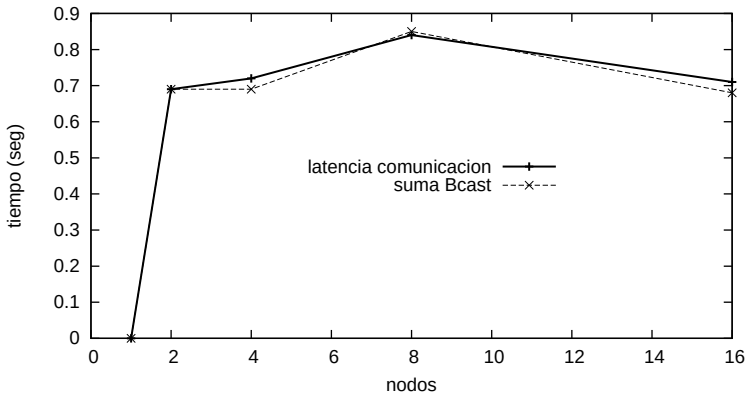


Figura 4.112: Comparación tiempos comunicación. switch-10$Mbps$

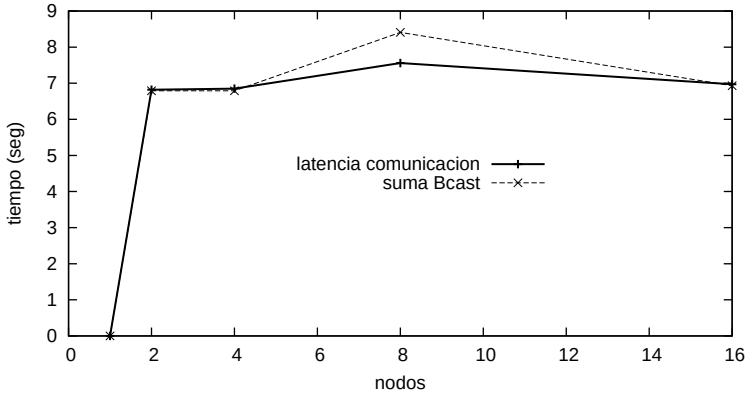
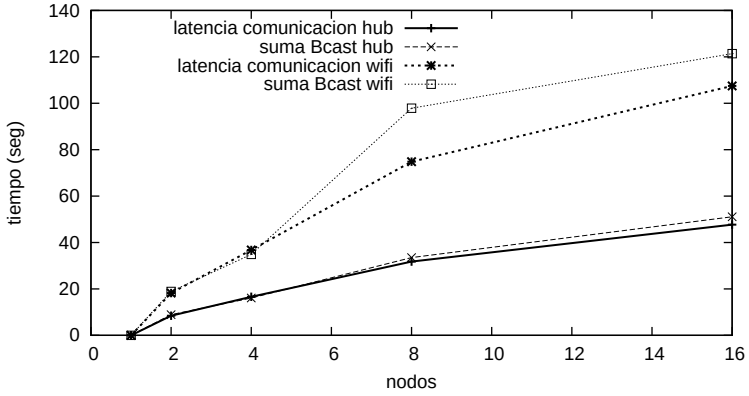


Figura 4.113: Comparación tiempos comunicación. Hub y Wifi



Puede observarse gran similitud entre las gráficas de *latencia de comu-*

nicación, obtenida a partir del tiempo de ejecución, y de *suma de Bcast* obtenida para estimar el tiempo de comunicación, a partir de la suma de tiempos de funciones *Bcast*. Sin embargo, en el caso de la red wifi, se observa una divergencia entre ambas curvas, que podría ser debida, en parte, a irregularidades detectadas en los datos de que se dispone de la ejecución de IMB-Bast para 2 nodos, y que se muestran a continuación.

Tabla 4.24: Datos de la ejecución de *IMB-Bcast*

Bytes	**2 nodos** (seg)	4 nodos (seg)
65536	127447.01	531953.14
131072	268508.82	1038381.60
262144	507681.01	2012707.11
524288	2082159.21	3979421.08
1048576	4929343.22	7913571.77
2097152	4576611.46	15794822.28
4194304	9910334.23	30419769.94

En los tiempos proporcionados en la tabla 4.24 se comprueba que, para 4 nodos, la latencia se duplica, aproximadamente, cuando también lo hace el tamaño del mensaje (ocurre lo mismo para otras redes). Sin embargo, para 2 nodos, el tiempo correspondiente a $524288 Bytes$ es 4 veces el de $262144 Bytes$ (debería ser el doble), y el correspondiente a $1048576 Bytes$ es mayor que el de $2097152 Bytes$(debería ser la mitad).

Matriz 2000x2000 Probamos con un tamaño mayor de matriz, buscando que, con cualquiera de las redes de que disponemos, el tiempo de ejecución mejore al aumentar el número de nodos. Los resultados obtenidos con esta aplicación se muestran en las figuras 4.114 y 4.115.

Figura 4.114: Multiplicación Matriz 2000x2000

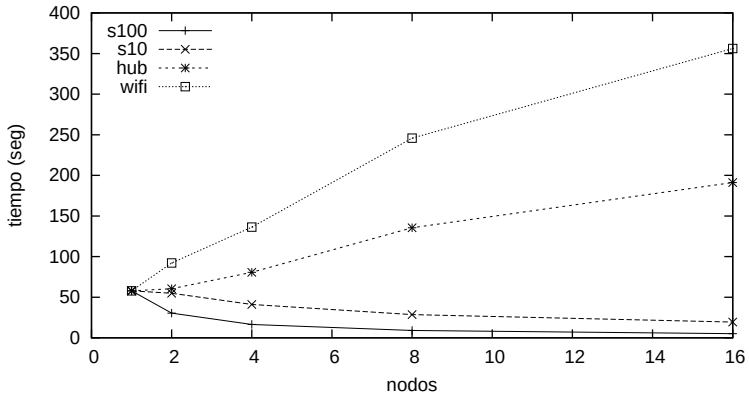
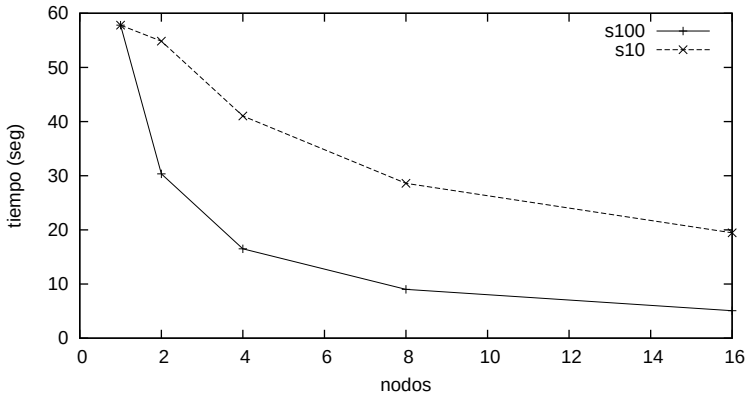


Figura 4.115: Multiplicación Matriz 2000x2000



Al aumentar el tamaño de matriz obtenemos el resultado que buscábamos para la red conectada con el switch de $10Mbps$. Sin embargo, en el caso de las redes hub y wifi, el tiempo de ejecución sigue aumentando con el número de nodos.

Matriz 4000x4000 Consideramos un tamaño mayor, buscando mayor escalabilidad.

Figura 4.116: Multiplicación Matriz 4000x4000

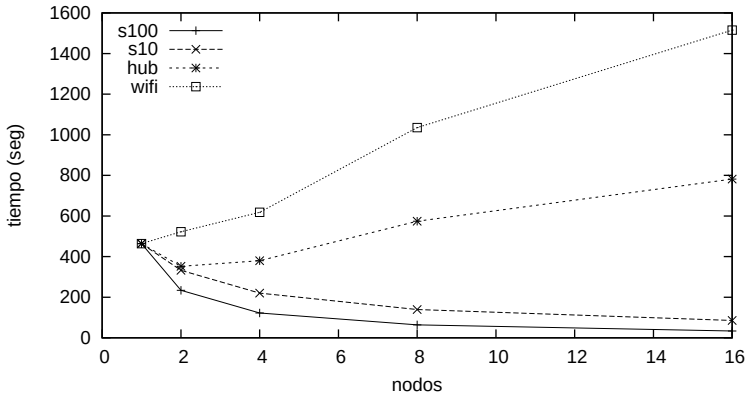
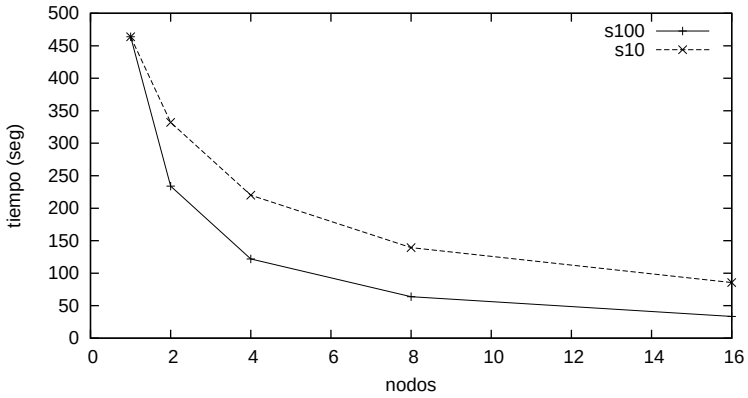


Figura 4.117: Multiplicación Matriz 4000x4000



Si bien en este ejemplo se observa que para la red conectada mediante un hub, el tiempo de ejecución para 2 nodos es menor que la ejecución en un sólo nodo, al aumentar más aún el número de nodos, vuelve a incrementarse el tiempo de ejecución, quedando éste por encima del valor para un nodo. En el caso de la red wifi, los tiempos obtenidos se encuentran siempre por encima de la ejecución en un nodo.

Matriz 8000x8000 Probamos un nuevo tamaño de matriz con el fin de aumentar la proporción entre tiempo de cómputo frente al de comunicación, buscando mayor escalabilidad.

Figura 4.118: Multiplicación Matriz 8000x8000

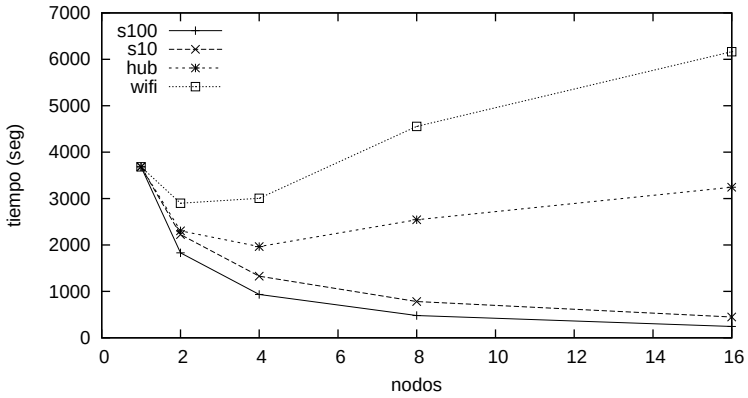
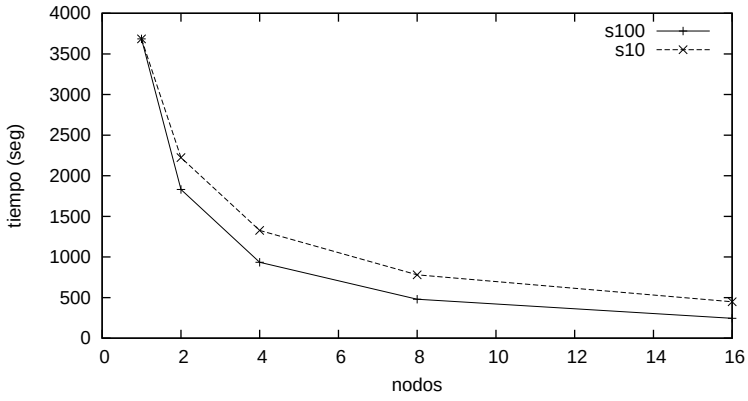


Figura 4.119: Multiplicación Matriz 8000x8000



En este tamaño de matriz se observa escalabilidad hasta 16 nodos, al menos, en las redes conectadas mediante switch, hasta 4 nodos en la red hub, y con 2 nodos en la wifi.

Multiplicación Matriz con procesador más lento A la vista de los resultados obtenidos, se comprueba que la escalabilidad que presenta la red conectada mediante un switch de $10Mbps$ es escasa, y prácticamente nula en las redes hub o wifi. Esto nos lleva a preguntarnos cómo sería la computación paralela cuando las redes de que se disponía eran más lentas de lo que son hoy en día, y si la falta de escalabilidad observada puede deberse a que se están combinando, por un lado, redes antiguas como lo es la conectada mediante un hub (wifi con comportamiento similar a hub) con procesadores rápidos (relativamente modernos). Se plantean nuevas pruebas en las que se ejecuta el algoritmo de la multiplicación, simulando procesadores más lentos. Para ello se ejecutan, de forma simultánea a la multiplicación, varios programas que consumen CPU.

Figura 4.120: Multiplicación M=1000x1000 + Sobrecarga, para Switch $10Mbps$

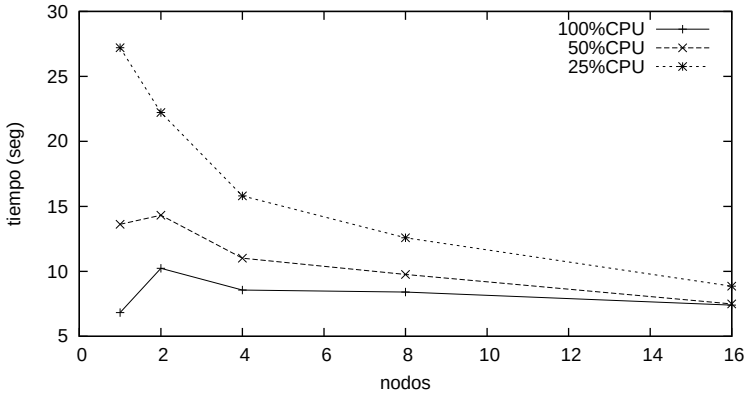


Figura 4.121: Multiplicación M=2000x2000 + Sobrecarga, para Switch $10Mbps$

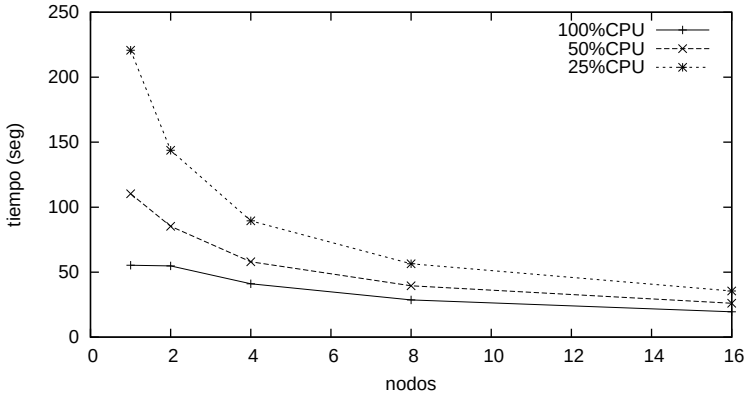


Figura 4.122: Multiplicación M=2000x2000 + Sobrecarga, para Hub

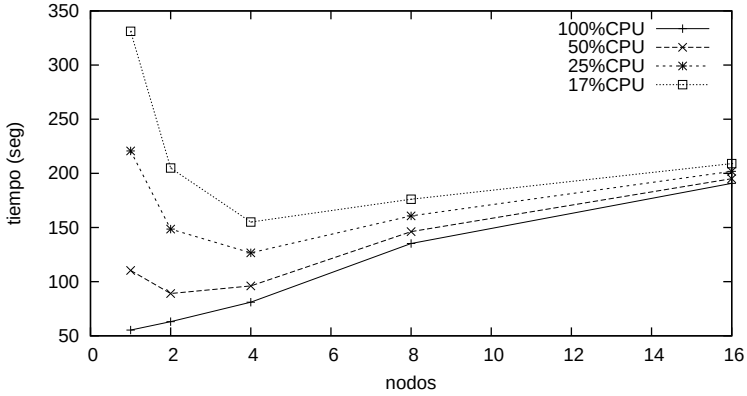
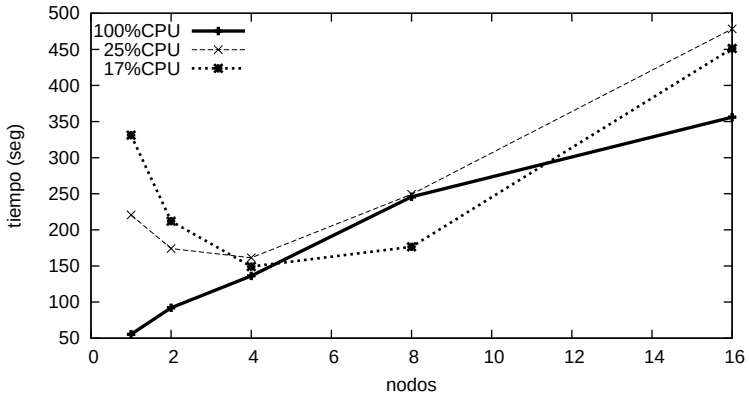


Figura 4.123: Multiplicación M=2000x2000 + Sobrecarga, para Wifi



Observamos (figuras 4.120 a 4.123) que con cualquiera de las redes, a medida que disminuye la capacidad de cómputo, se consigue mayor escalabilidad. Esto es debido a que se reduce el tiempo de comunicación respecto al de cómputo. Como ya se comentó al principio de la sección 4.3, para mejorar el tiempo de ejecución al aumentar el número de nodos, el tiempo añadido debido a la comunicación debe ser inferior a la reducción que sufrirá el tiempo de cómputo como consecuencia de repartir el *trabajo* entre más nodos. En las gráficas 4.122 y 4.123, correspondientes a las redes Hub y Wifi, en las que no son posibles las comunicaciones simultáneas, observamos que cuando el procesador es suficientemente lento, o equivalentemente, cuando el tiempo de comunicación es suficientemente pequeño (respecto al de cómputo), al aumentar el número de nodos disminuye el tiempo de ejecución. Aún así, en este caso, sólo escala hasta 4 nodos. Esto se debe a que, tal como hemos podido comprobar a lo largo del presente estudio, el tiempo de comunicación en estas redes crece considerablemente con el número de nodos. Por otro lado, el tiempo de cómputo decrece de forma exponencial, tal como se vió en la figura 4.87. Por tanto, aún cuando se da la condición de que el tiempo añadido de comunicación es menor que la reducción del tiempo de cómputo, y dado que éste último decrece muy rápidamente, mientras que el primero crece, más o menos rápido, con el número de nodos, la escalabilidad que presentan estas redes es escasa. Estas observaciones ponen de manifiesto que, para conseguir escalabilidad en la resolución de problemas en paralelo, la red utilizada debe, además de tener un determinado ancho de banda, permitir comunicaciones simultáneas.

Capítulo 5

Conclusiones

Tras realizar y analizar los diferentes experimentos presentados en esta memoria, podemos considerar que las principales conclusiones que se obtienen son:

1. El objetivo de la computación paralela es reducir el tiempo de ejecución de aplicaciones que precisan una enorme cantidad de operaciones. Su estrategia consiste en repartir el trabajo a realizar entre distintos nodos procesadores. Ahora bien, para que varios nodos lleven a cabo un trabajo común, necesitan comunicarse. Debe tenerse en cuenta que el tiempo empleado en esa comunicación se sumará al de cómputo para dar el tiempo final de ejecución de la aplicación paralela. Se debe poner un verdadero interés en reducir al máximo la latencia de comunicación, ya que el éxito o fracaso en la obtención de un buen tiempo de ejecución depende, en gran medida, de esta latencia. Una deficiente planificación de la red de interconexión podría llevarnos a casos extremos como el visto en la figura 4.88, en el que una aplicación, que un sólo ordenador ejecutaba en 55 minutos, al ser ejecutada en paralelo, dividiendo el trabajo entre 16 ordenadores, pasaba a tardar 2 días y 7 horas.

2. En el rendimiento influyen las características de red, no sólo de ancho de banda, sino determinadas características que permiten obtener ganancia de ancho de banda. Hemos podido comprobar que el hecho de que la red permita o no comunicaciones simultáneas, influye en gran medida en la latencia. Si la red utilizada no permite ningún tipo de comunicación simultánea, no es posible conseguir gran escalabilidad, ya que la latencia siempre crecerá con el número de nodos. En una situación como ésta, las únicas aplicaciones paralelas que presentarían un buen comportamiento serían aquellas que tuvieran muy poca comunicación, con lo que se limitaría enormemente el potencial de la computación paralela. Podemos decir, por tanto, que para obtener buenos resultados,

es imprescindible, no sólo que la red utilizada tenga ancho de banda suficiente, sino que, además, permita comunicaciones simultáneas.

3. Con el fin de mejorar la latencia en comunicación, es importante conocer la implementación de *MPI* que se utiliza, para poder comprobar si es adecuada, o no, para nuestra red. En algunos casos, utilizando para ello los parámetros adecuados, será posible que ésta sea adaptada para mejorar la latencia de comunicación. Habrá casos en los que será conveniente modificar algunos algoritmos contenidos en la implementación de *MPI* para que éstos se adecúen a las características de la red sobre la que va a ser ejecutado, con el fin de obtener un mayor rendimiento.

4. El estudio llevado a cabo y que ha sido expuesto en la presente memoria, ha permitido alcanzar el primer objetivo para el que fue planteado, que es proporcionar al proyectando una primera aproximación a las tareas de investigación científica, así como consolidar y ampliar sus conocimientos sobre el manejo de equipos informáticos y de dispositivos de red.