Future Generation Computer Systems 139 (2023) 126–138

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Energy efficient HPC network topologies with on/off links

Francisco J. Andújar [a,*], Salvador Coll [b], Marina Alonso [c], Juan-Miguel Martínez [c], Pedro López [c], José L. Sánchez [d], Francisco J. Alfaro [d]

[a] Departamento de Informática, Universidad de Valladolid, Valladolid, Spain
[b] Instituto de Instrumentación para Imagen Molecular, Centro Mixto CSIC - Universitat Politècnica de València, Valencia, Spain
[c] Department of Computer Engineering, Universitat Politècnica de València, Valencia, Spain
[d] Computing System Department, University of Castilla-La Mancha, Albacete, Spain

ARTICLE INFO

Article history:
Received 3 January 2022
Received in revised form 10 September 2022
Accepted 16 September 2022
Available online 28 September 2022

Keywords:
Interconnection networks
Energy and performance evaluation
Energy consumption
Power consumption
Topology configuration
n-dimensional torus
Fat-tree
Dragonfly

ABSTRACT

Energy efficiency is a must in today HPC systems. To achieve this goal, a holistic design based on the use of power-aware components should be performed. One of the key components of an HPC system is the high-speed interconnect. In this paper, we compare and evaluate several design options for the interconnection network of an HPC system, including torus, fat-trees and dragonflies. State of the art low power modes are also used in the interconnection networks. The paper does not only consider energy efficiency at the interconnection network level but also at the system as a whole.

The analysis is performed by using a simple yet realistic power model of the system. The model has been adjusted using actual power consumption values measured on a real system. Using this model, realistic multi-job trace-based workloads have been used, obtaining the execution time and energy consumed. The results are presented to ease choosing a system, depending on which parameter, performance or energy consumption, receives the most importance.

1. Introduction

High Performance Computing (HPC) systems are based on the aggregation of multiple computing nodes (processors and memory banks) to provide the best possible computing support for computational problems not addressable by accessible commercial computers. The interconnection network is one of the basic building blocks of an HPC system, providing communication among potentially many thousands or even millions of computing nodes.

Interconnection networks are a significant performance limiting factor of HPC systems. For this reason, they are an active area of research where new topologies, routing algorithms or switch architectures are proposed. The goal is offering high-communication bandwidth, low latency and good scalability to make it possible to interconnect all nodes.

The United Nations in 2015 approved the 17 Sustainable Development Goals (SDG) to transform our world. Energy is the main contributor to climate change and accounts for around 60% of all global greenhouse gas emissions [1]. Thus, the efficient use of energy is one of the key aspects to consider to help combat climate change and global warming. HPC systems cannot be left out and must be energy efficient in order to contribute to energy savings that facilitate a more sustainable planet.

While significant contributions have been made to increase interconnection network performance, much lower attention has been paid to its energy consumption. HPC systems are significant energy consumers, e.g. the current most powerful supercomputer, Fugaku, reaches nearly 30 MW [2,3]. This is high enough to supply energy to more than 33,000 homes, according to U.S. Energy Information Administration standard. The most-energy efficient system, NVIDIA DGX SuperPOD, achieves 26.195 GFlops/W [2,4]. A projection of that metric to an exascale supercomputer with equivalent efficiency predicts power consumption will peak 473 MW, soon reaching gigawatt figures. In order to mitigate this trend, energy-proportional computing nodes are being deployed. The strategy is modulating energy consumption at server level according to processors' utilization. Under these conditions, interconnection networks contribution to overall system energy consumption significantly increases, ranging from 12% to 50% depending on servers utilization (higher budget for lower server utilization) [5,6].

As a consequence, a trade-off between performance and cost, which is greatly conditioned by power consumption, must be considered in the design of an HPC system and its interconnection network. From this perspective, as network performance

* Corresponding author.
E-mail addresses: fandujarm@infor.uva.es (F.J. Andújar), scoll@upv.es (S. Coll), malonso@upv.es (M. Alonso), jmmr@upv.es (J.-M. Martínez), plopez@upv.es (P. López), jose.sgarcia@uclm.es (J.L. Sánchez), fco.alfaro@uclm.es (F.J. Alfaro).

https://doi.org/10.1016/j.future.2022.09.012

is greatly conditioned by its topology, an appropriate topology selection will contribute to the overall system performance. Nevertheless, that decision also impacts on the ability of the network to show an energy-proportional behavior, contributing to an energy-efficiency design.

Historically, several topologies for interconnection networks have been proposed. Nevertheless, almost all high-performance interconnection networks in practical use over the last two decades use topologies derived from three families: tori (k-ary n-cubes) [7], fat-tree (k-ary n-tree) [8] and, recently, dragonfly [9]. For any given network size, any topology can be chosen using similar link technology and applying different power-saving strategies (pursuing energy-proportionality).

Selecting the topology that offers the best trade-off between performance and energy consumption is not straightforward. The energy consumed by a computing system depends on power consumption over time. Energy-efficiency strategies may reduce power during low utilization periods but increase overall execution time. Eventually this might provide a non-acceptable performance degradation and/or an undesired increase of the energy consumption of the system.

In this paper, we present a comparative performance-wise and energy-efficiency-wise analysis of different network topologies. We characterize network behavior and its impact on the overall HPC system when applying dynamic power saving techniques. Performance of the different configurations is determined by using real application execution traces. The main contribution of this paper is a thorough comparative evaluation of three family of topologies: tori, fat-trees and dragonflies. In all cases, several topology parameters are considered, leading to a representative set of evaluated topologies. In addition, the Low Power Idle proposed on the IEEE Energy Efficient Ethernet standard is used as dynamic power saving mechanism. Being a standard, the obtained results can be more easily generalized to commercial interconnection networks. Evaluation results are shown by using combined performance–energy plots which allow an easy comparison and selection of the proper configuration according to either the criterion of performance-first or energy-first.

The rest of the document is organized as follows. Section 2 describes the power consumption model. Section 3 presents the system and evaluation model. After that, we‘ analyze and discuss network performance and energy evaluation results in Section 4. Other works studying energy consumption in high-speed interconnection networks are reviewed in Section 5. Finally, in Section 6 we outline the conclusions and future work.

2. Power model

As mentioned in Section 1, the main goal of this work is to show the impact of the energy consumption of an HPC platform for different configurations of the interconnection network topology.

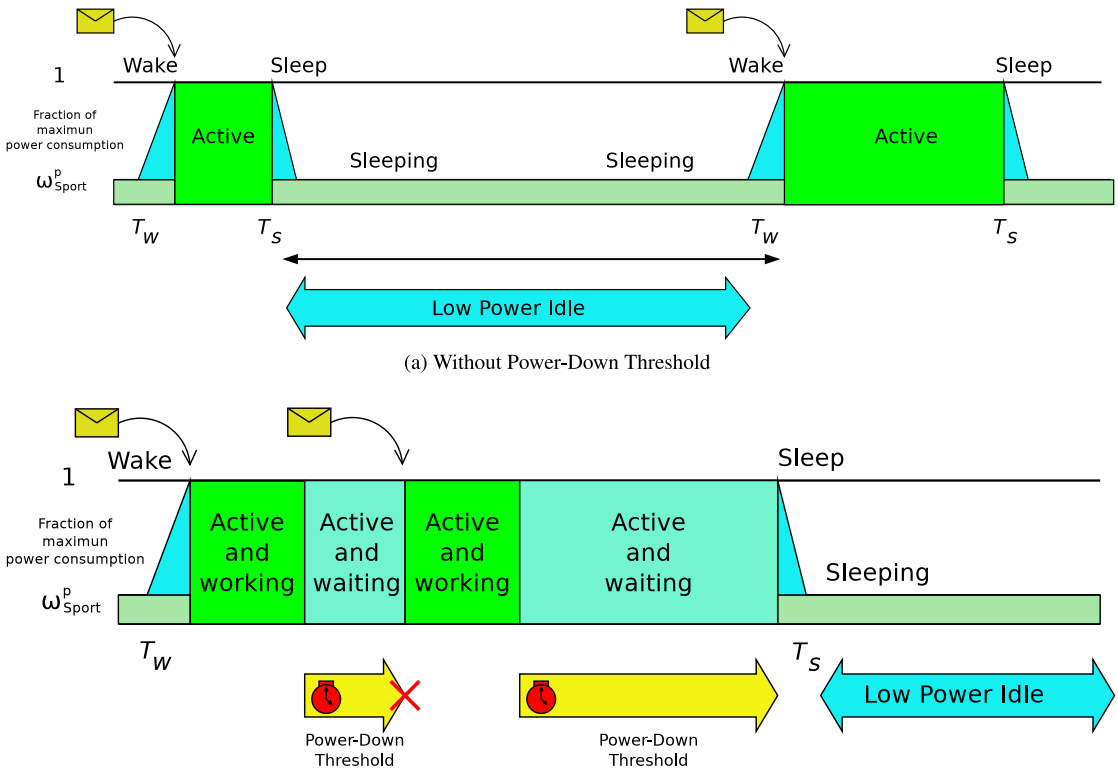
In previous works, we defined a power consumption model to compare networks with different number of switches/ports [10,11]. This model selects one of the compared networks as a *reference network* and normalizes all the power results according the switch/port configuration of the *reference network*. This approach allows us to compare different network configurations when we do not have power measurements of the components of the HPC system.

However, in this work we have made power measurements of some current HPC nodes and HPC switches in order to characterize the power model parameters (Section 2.4). As we now have these power values in Watts, the model has been slightly modified to work directly with absolute power measurements. For this reason, and for the paper to be self-contained, we include a set of definitions and a brief explanation about the modified power model.

2.1. Initial hypotheses

Our model considers separately the power consumption of the computing nodes and the power consumption of the network switches. Moreover, the link power consumption and the switch logic power consumption are also studied separately, due to the relevance of the links in the performance and energy efficiency of the network. According to the state of the art, our model considers the following general hypotheses:

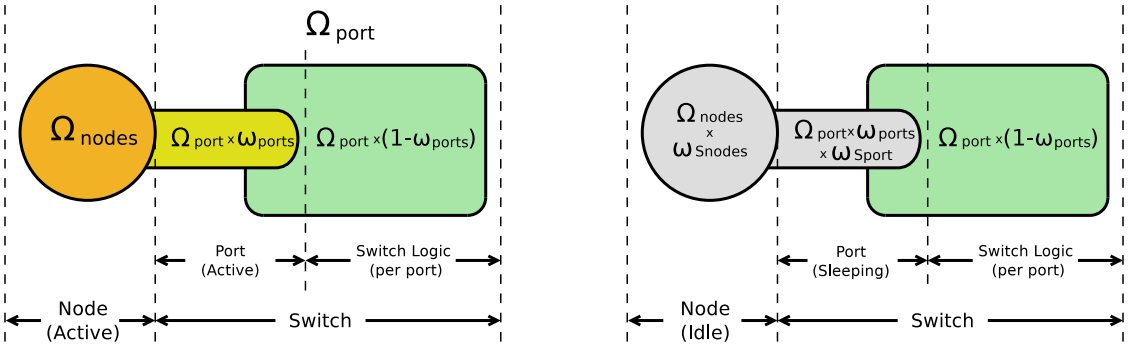
- The switch power consumption increases linearly with the number of ports [12], which has been experimentally verified in [13].
- A power-saving mechanism like Low Power Idle (LPI), proposed on the IEEE Energy-Efficient Ethernet (*EEE*) standard (IEEE 802.3az) [14] is assumed, and therefore, two switch port states are considered: *wake-up* (or *turned on*) and *sleep* (or *turned off*). Fig. 1(a) illustrates how a link with LPI operates under our hypotheses:
  - Since the transceiver is working independently of the port is transmitting data or not, the port power consumption is 100% when it is turned on.
  - When the transmission of a packet ends and no more packets are available for transmission, the link is switched to the low power mode. After a certain amount of time (T_s), the link is in LPI mode.
  - When the port is in *sleep* mode, the transceiver is frozen and it consumes a small part of its maximum power consumption [15]. As can be seen in Fig. 1, this fraction of power consumption is specified in our model by the ω^{p}_{Sport} parameter.
  - When packets need to be transmitted, the link is waked up to the active mode. This change of state also requires a certain amount of time (T_w) to finish.
  - During the transitions from one state to another, the port power consumption is 100%.
- As shown in other studies [11,16], using only LPI in the context of HPC applications considerably increases the execution time of these applications, and in some cases, the increase can be more than 100%. Not only this degrades the obtained performance, but also the total system consumption can be even higher than without using the LPI technique. However, LPI becomes an attractive technique for saving energy when it is combined with Power-Down Threshold (PDT) . Thus, in this work the PDT technique [16] is also implemented.
  Fig. 1(b) shows how the link with LPI and PDT works. Each link has a timer configured with the Power-Down Threshold value. After finishing a transmission, the timer starts. If one message is received, the timer is canceled. Since the link is active, it can immediately start the transmission of the message. Once the new transmission finishes, the timer is restarted again. Only when the timer finishes without receiving new messages, the link starts the transition to sleep mode.
- In order to simplify the model, we consider that the power consumption of the switch control logic is fixed regardless of the switch utilization. Moreover, this behavior is observed in the experiments performed in [13].
- The nodes always consume a fraction of its maximum power consumption, in spite of there are no running processes in the node (*idle* node). Then, a fraction of the node power consumption is fixed and the remaining power depends on the node utilization. This hypothesis has been also experimentally proven in [13].



(a) Without Power-Down Threshold

(b) With Power-Down Threshold

Fig. 1. Operation of a link with Low Power Idle Mode.



(a) Active hardware

(b) Idle/sleeping hardware

Fig. 2. Power parameters of the power model.

2.2. Definitions

Table 1 introduces the parameters used to quantify both, the main components of the system and their contribution to power consumption and total energy, as well as the variables calculated by the power model.

Moreover, Fig. 2 shows graphically the main power parameters of the model, in order to help the reader to understand these parameters. The figure shows a node connected by a port to the switch, showing the power consumption when the hardware is active (Fig. 2(a)) and when it is inactive (Fig. 2(b)).

Then, in Fig. 2(a) we can see the power consumption when all the hardware is active. The node power consumption is Ω_{nodes} W, while the switch power consumption per port is Ω_{port} W. Moreover, we can distinguish two power components within Ω_{port}: the power of the port, and the power of the switch logic. The contribution of the ports to the total switch power consumption is indicated by ω_{ports} parameter, therefore, the port power consumption is $\Omega_{port} \times \omega_{ports}$ W, while the switch logic consumption is $\Omega_{port} \times (1 - \omega_{ports})$ W.

Finally, Fig. 2(b) shows the power consumption when the node is idle and the port is sleeping. Both components only consume a fraction of its maximum power consumption. While ω_{Sport} indicates the fraction of power consumption of a sleeping port, ω_{Sport} indicates the fraction consumed by an idle node. Therefore, the power consumption of the port is $\Omega_{port} \times \omega_{ports} \times \omega_{Sport}$ W and the power consumption of the node is $\Omega_{nodes} \times \omega_{Snodes}$ W. The power consumption of the switch logic power consumption remains the same as in Fig. 2(a).

2.3. Power consumption model

Let W^s_{ports} be the fraction of the power consumption that all the ports consume in a switch s with respect to the maximum power consumption of those ports. When a port p is in *sleep* mode, it

Table 1
Power model parameters and variables.

Network parameters	
N_{ports}	Number of ports per switch
N_{sw}	Number of switches in the network
N_{nodes}	Number of nodes in the system
Power parameters (based on literature or power measurements)	
ω^p_{Sport}	Fraction of the maximum port power consumption that a *sleep* port p consumes
ω^s_{ports}	Contribution of the ports in a switch s to the total switch power consumption
Ω_{port}	Switch power consumption per port (Watts)
Ω_{nodes}	Max. power consumption of a comp. node (Watts)
ω_{Snodes}	Fraction of the max. node power consumption that an *idle* node consumes
Variables obtained from simulation	
T_{exec}	Execution time of the HPC application (seconds)
U^p_{port}	Fraction of T_{exec} that a port p is turned on
U^n_{cpu}	Fraction of T_{exec} that a node n is running
Variables calculated by the power model	
W^s_{ports}	Fraction of the maximum power consumption of all switch ports consumed by the ports of switch s
W^s_{sw}	Power consumption of a switch s (Watts)
W_{net}	Power consumption of the network (Watts)
W_{nodes}	Power consumption of the compute nodes (Watts)
$W_{cluster}$	Power consumption of the cluster system (Watts)
E_{net}	Energy consumption of the network (Joules)
$E_{cluster}$	Energy consumption of the cluster system (Joules)

only consumes ω^p_{Sport} of its maximum power consumption. Then, a port p always consumes ω^p_{Sport}, plus $(1-\omega^p_{Sport})\cdot U^p_{port}$ when it is turned on.

Since the ports in a switch have the same characteristics, ω^p_{Sport} is the same for all ports (ω_{Sport} will be used hereafter), and the relative ports power consumption is:

$$W^s_{ports} = \omega_{Sport} + (1-\omega_{Sport})\frac{1}{N_{ports}}\sum_{i=1}^{N_{ports}} U^i_{port} \quad (1)$$

Once we have the ports power consumption, we can calculate the switch power consumption. According to the initial hypotheses, the switch logic always consumes $(1-w^s_{ports})$ of the total switch power consumption. Therefore, the fraction of the maximum power consumed by a switch is:

$$(1-\omega^s_{ports}) + \omega^s_{ports}\cdot W^s_{ports}$$

and therefore, the absolute switch power consumption, measured in Watts, is:

$$W^s_{sw} = N_{ports}\cdot\Omega_{port}\cdot\left((1-\omega^s_{ports}) + \omega^s_{ports}\cdot W^s_{ports}\right) \quad (2)$$

By considering all switches in the network, we obtain the network power consumption. Without loss of generality, and reasoning at network level in the same way as switch level, we consider that the contribution of switch ports to the total switch power consumption is the same for all switches, i.e. ω^s_{ports} is the same for all the switches, and we use ω_{ports} hereafter. Then:

$$\begin{aligned} W_{net} &= \sum_{i=1}^{N_{sw}} W^i_{sw} \\ &= \sum_{i=1}^{N_{sw}} \left(N_{ports}\cdot\Omega_{port}\cdot\left((1-\omega_{ports}) + \omega_{ports}\cdot W^i_{ports}\right)\right) \\ &= N_{ports}\cdot\Omega_{port}\cdot\left(N_{sw}\cdot(1-\omega_{ports}) + \omega_{ports}\cdot\sum_{i=1}^{N_{sw}} W^i_{ports}\right) \end{aligned} \quad (3)$$

Table 2
Power model parameter characterization.

Parameter	w_{Sport}	w_{ports}	Ω_{port}	Ω_{nodes}	ω_{Snodes}
Value	0.1	0.816	5 W	300 W	0.5

In order to obtain the total energy of the HPC platform, we need to consider the power consumption of the computing part, mainly due to the compute nodes. Taking into account the definitions above and the initial hypotheses, a node always consumes w_{Snodes} of the total node power consumption. Therefore, the power consumption of a node n is:

$$\Omega_{nodes}\cdot\left(w_{Snodes} + (1-w_{Snodes})\cdot U^n_{cpu}\right)$$

and then, the power consumption of the cluster nodes is:

$$\begin{aligned} W_{nodes} &= \sum_{i=1}^{N_{nodes}} \Omega_{nodes}\cdot\left(w_{Snodes} + (1-w_{Snodes})\cdot U^i_{cpu}\right) \\ &= \Omega_{nodes}\cdot\left(w_{Snodes} + (1-w_{Snodes})\cdot\sum_{i=1}^{N_{nodes}} U^i_{cpu}\right) \end{aligned} \quad (4)$$

Then, considering the nodes and the network power consumption, we can calculate the HPC platform power consumption:

$$W_{cluster} = W_{net} + W_{nodes} \quad (5)$$

and the energy consumed by an application:

$$E_{net} = W_{net}\cdot T_{exec} \quad (6)$$

$$E_{cluster} = W_{cluster}\cdot T_{exec} \quad (7)$$

2.4. Characterization of power model parameters

To use the power model defined in this section, values must be given to the parameters that characterize the model, e.g. the fraction of power dissipated by a port in *sleep* mode, the switch port power consumption or the compute node power consumption.

There are numerous documents that provide data on the power consumption of the main components of an HPC system or of the system as a whole. However, these data, for the same type of components, offer significant differences, and thus it is not easy to choose one in particular. Therefore, we have decided to collect power consumption data in a real and modern HPC cluster, using a meter for this purpose. Except for w_{Sport}, all other power model parameters have been selected according to the results obtained. In [13] we have included the characteristics of the measuring instrument, the tests carried out and the results obtained.

The power model parameters are summarized in Table 2. According to the EEE standard, the power consumption of an idle link is estimated to be 10% of the link power consumption [14,15], and therefore, w_{Sport} is set to 0.1. The experiments in [13] show that a modern 36-port Mellanox switch consumes approximately 33 W without connected ports, and 180 W with all its ports connected. Therefore, w_{ports} is set to $(180-33)/180 = 0.816$ and Ω_{port} is set to $180/36 = 5$ W. Regarding the compute nodes, the power consumption of a fully-loaded node is 300 W, while in a node with an unique thread running is only 150 W. Then, Ω_{nodes} is set to 300 W and ω_{Snodes} is $150/300 = 0.5$.

3. System model

After presenting the power model, we describe the system model used in the evaluation. Section 3.1 outlines the network model while Section 3.2 shows the topologies selected for our experiments. Finally, Section 3.3 briefly explains the network load model.

3.1. Network model

The network has been modeled using the simulation tool Hiperion (HIgh PERformance InterconnectiOn Network) [17]. Hiperion is an open-source simulation tool available for researchers and companies. The simulator main goal is to perform comparative studies and it has a large range of configurable parameters, e.g. topology, routing, queue sizes, output scheduling algorithms, etc. Hiperion is capable of running simulations using synthetic traffic (e.g. random uniform, bit-reversal, bit-complement, etc.) and MPI traffic using the *VEF trace framework* [18,19]. The Hiperion configuration used to perform all the experiments in the performance study is detailed below.

The Hiperion modeled architecture is realistic and representative of current state of the art HPC platforms. The chosen design parameters are based on several commercial networks [20–24].

Network switches are *IQ* (*Input Queued*) [25], with *virtual cut-through* [26], credit-based flow control and the three-stage allocation algorithm implemented in the IBM Blue Gene L [27].

Flit size is 16 bytes and packets are 8 flit long. The switch clock frequency is 625 MHz (i.e. clock cycle equal to 1.6 ns). Since the switch crossbar can deliver one flit per cycle, each switch port offers a peak bandwidth of 10 Gb/s.

We assume that all switch components have a fixed delay. The latency per hop is approximately 50 ns, slightly varying as a function of the number of ports. The unique switch component with a variable latency is the switch allocator. The allocator comprises several stages of round-robin arbiter, which latency increases logarithmically with the number of arbiter entries [28]. Therefore, the allocator latency increases logarithmically with the number of ports.

Each physical channel is multiplexed into 4 virtual channels (VCs). Each input port has an input buffer of 1024 flits, or 16 Kbytes, statically split between the VCs. The use of the VCs depends on the modeled topology:

- In the torus topologies, the VCs are employed to avoid deadlocks and to enable the use of adaptive routing. In this case, the switch implements a fully-adaptive routing algorithm [29]. Two of the four VCs are fully-adaptive VCs, while the remaining VCs are used as escape paths, implementing the DOR algorithm to avoid deadlocks [30].
- In the fat-tree topologies, since VCs are not required for avoiding deadlocks or providing adaptiveness, they are used to implement DBBM [31] and reduce the Head-Of-Line blocking, mapping the packets in the VCs using the function *Destination % Number_of_VCs*.
- In the dragonfly topology, UGAL-L routing algorithm [9] is used. The number of VCs has been modified, since this algorithm requires 3 VCs in local channels and 2VCs in global channels for avoiding deadlock.

Although our switch is not referred to EEE or any specific technology, we have implemented Low Power Idle mechanism [14] and Power-Down Threshold (PDT) [16] for saving energy on the links. We have used the time values specified in IEEE Energy-Efficient Ethernet standard [15] to configure the delays for turning on (4.16 μs) and turning off (2.88 μs) a link.

In a previous work, we evaluated the impact of PDT value in torus and fat-tree topologies. The results showed that the greatest power savings with negligible performance penalties were obtained when the adaptive routing prioritizes the *wake-up* links and PDT is set to 10 μs [11]. Therefore, this value has been chosen for the experiments. However, the dragonfly topology does not have an adaptive routing. The routing algorithm selects a path without having information about the power state of the links in the chosen path. This may cause a poor performance setting PDT to 10 μs. For this reason, the dragonfly topologies are also tested setting PDT to 100 μs.

3.2. Case studies

To perform the evaluation, the first idea was to compare different systems with a fixed number of computing nodes (64 and 256). These system sizes are possible in fat-tree and torus topologies, but not in the dragonfly topology. For this reason, we have chosen dragonfly configurations that: (i) their number of computing nodes is closer to 64 and 256; and (ii) they fulfill the conditions $a \geq 2h$ and $2p \geq 2h$ to balance the network load [9]. Moreover, in the second case study (256 nodes), we have also added extra configurations for fat-tree and torus topologies which have a closer system size to dragonfly configurations. Tables 3 and 4 show the topologies evaluated in both case studies and other useful information, such as the number of nodes, switches, ports per switch and total network ports. Note that the torus topologies marked as *Trunk* in both tables use trunk links in each torus direction. Each trunk link comprises four independent links transmitting independent packets.

The labels used to identify each topology in Section 4 are included in the last column of Tables 3 and 4. When the network employs the power saving mechanism and PDT is set to 10 μs, the suffix *"-P"* is added to the topology label. The suffix *"-PH"* indicates that PDT is set to 100 μs.

3.3. Network load

The load on the system plays an important role when evaluating the energy consumed that will be obtained from the execution time and the power consumed. We have decided to use an open access trace-driven traffic model, called *VEF trace framework* [18, 19]. The parallel applications inject MPI traffic that will be captured in a trace file that we will use to generate the traffic in the network simulator. The VEF traces model both point-to-point communications and MPI collective communication primitives, making use of the collective communication algorithms developed in Open MPI [32].

For modeling the network traffic, we have run the following applications in the *GALGO* supercomputer [33], trying to make the scenarios as real as possible:

- *HPCC Linpack* [34] is used to solve a dense system of linear equations. This application follows a specific pattern in which a given task always communicates with the same tasks, following a ping-pong traffic pattern.
- *Namd* is a parallel application for simulating large biomolecular systems [35]. The application maps logically the tasks in a 3D grid and the tasks communicate mainly with their neighbor tasks in the grid. For this reason its traffic pattern shows a great spatial locality. Our traces correspond to the *apoa1* benchmark.
- *Gromacs* [36] is a scientific application to perform molecular dynamics. Similarly to the previous one, this application shows a great spatial locality. We generated the trace using the input "d.poly-ch2" available in the Gromacs benchmark.[1]
- *Graph500 benchmark* using the *replicated-csr* implementation, a scale factor of 20 and an edge factor of 16 [37]. All the communications are generated by MPI collective communications that generate a great exchange of data among tasks. This application generates the highest network load of the all applications tested.
- *HPCC* ***MPI Random Access*** [34] (or MPIRA). Most of the communications are performed by MPI point-to-point primitives. The messages are uniformly distributed among all the tasks, being the traffic pattern very close to a uniform traffic pattern.

[1] http://www.gromacs.org/About_Gromacs/Benchmarks.

Table 3
First case study: 64-node tori, fat-trees, and near-sized dragonflies.

	Topology configuration	Num. nodes	Num. switches	Switch ports	Net. ports	Base name
Torus	3D 4 × 4 × 4	64	64	7	448	3D
	2D 4 × 4 Trunk	64	16	20	320	2D
Fat-tree	k = 4 n = 3	64	40	8	320	k4-n3
	k = 8 n = 2	64	12	16	192	k8-n2
Dragon-	a = 4 h = 2 p = 2	72	36	7	252	D72
fly	a = 4 h = 1 p = 4	80	20	8	160	D80

Table 4
Second case study: 256-node tori, fat-trees and near-sized dragonflies.

	Topology configuration	Num. Nodes	Num. Switches	Switch Ports	Net. Ports	Base Name
Torus	4D 4 × 4 × 4 × 4	256	256	9	2304	4D
	3D 4 × 4 × 4 Trunk	256	64	28	1792	3D
	4D 5 × 4 × 4 × 4	320	320	9	2880	4D-320
	3D 5 × 4 × 4 Trunk	320	80	28	2240	3D-320
Fat-tree	k = 4 n = 4	256	224	8	1792	k4-n4
	k = 16 n = 2	256	24	32	768	k16-n2
	k = 18 n = 2	324	27	36	972	k18-n2
Dragonfly	a = 8 h = 2 p = 2	272	136	11	1496	D272
	a = 6 h = 2 p = 4	312	78	11	858	D312
	a = 6 h = 3 p = 3	342	114	11	1254	D342

Trace scheduler evaluation

Since the number of tasks per trace is limited by the size of the *GALGO* supercomputer, we have developed an oblivious trace scheduler to solve this limitation and to be able to evaluate the most realistic network traffic.

Given a set of traces, the scheduler operates as follows: (i) the scheduler checks the number of available cores, (ii) the scheduler verifies the number of tasks in each trace and marks the traces as eligible if their number of tasks is lower than the number of available cores, (iii) the scheduler randomly chooses an eligible trace and allocates it to the first free nodes. Until all the traces are mapped or there are no free nodes to map another trace, the previous process must be repeated. When a trace is finished and its resources are released, the scheduler will run again.

We consider all nodes have 8 cores in the follow-up scheduler evaluation, e.g. the 64-node networks have 512 cores, the 256-node networks have 2048 cores, etc.

We have evaluated three different sets of traces, combining traces from the five applications shown in Section 3.3 with three different number of tasks: 128, 256 and 512 tasks.

All trace sets have the same number of applications (15 traces: Three task sizes per application). Trace sets have been configured to provide increasing communication requirements. This is achieved by removing Namd and Gromacs in sets *Mid* and *High*, respectively, while including additional instances of Graph500 that generates a higher network load.

The three trace sets are the following:

- **Set Low**: Namd, Gromacs, Linpack, MPI Random Access and Graph500. This trace set generates approximately 715 Gb and injects 645 Gb into the network.[2]
- **Set Mid**: Gromacs, Linpack, MPI Random Access and Graph500 (× 2). This trace set generates approximately 1.1 Tb and injects 1.05 Tb into the network.
- **Set High**: Linpack, MPI Random Access and Graph500 (× 3). This trace set generates approximately 1.58 Tb and injects 1.52 Tb into the network.

The amount of traffic generated by each trace and its contribution to the total traffic generated by each set can be found in Table 5.

Finally, it should be noted that this scheduler is an oblivious scheduler, i.e, the scheduler does not apply any strategy to optimize the use of the nodes or the network. The scheduler randomly selects an application to be run taking only into account the number of available cores. Note that designing an scheduler that optimizes the available resources is far away of the scope of this work. Therefore, for each set of traces and each topology, we have carried out 30 different executions varying the random seed, and the average value and the 95% confidence interval is shown in the evaluation figures.

4. Evaluation

This section presents the results of the performance and energy consumption evaluation. Section 4.1 shows the results for 64-node networks, while Section 4.2 shows the results for 256-node networks.

4.1. 64-node network evaluation

Fig. 3 shows *Runtime*, E_{net} and $E_{cluster}$ results for each topology, connecting at least 64 computing nodes. Every plot depicts three groups of contiguous bars, each one corresponding to the same topology (namely, torus, fat-tree and Dragonfly). Systems based on different network topologies have been tested with a power saving mechanism based on Low Power Idle and Power

[2] Note that the traffic between two tasks in the same multicore processors is not injected into the network and is internally managed by the *VEF trace framework*. For this reason, the amount of traffic injected into the network is slightly lower than the amount of traffic generated by the traces.

Table 5
Traffic generated by each application, measured in Gbytes, and percentage of traffic generated per each trace in each trace set.

App	Num. tasks	Traffic Gen. (Gbytes)	% of traffic generated per set		
			Low	*Mid*	*High*
Graph 500	128	69,32	9,69	6,09	4,28
	256	139,14	19,44	12,21	8,59
	512	302,08	42,21	26,52	18,65
Linpack	128	10,83	1,51	0,95	0,67
	256	16,47	2,30	1,45	1,02
	512	25,98	3,63	2,28	1,60
Mpi Random Access	128	6,58	0,92	0,58	0,41
	256	9,69	1,35	0,85	0,60
	512	18,36	2,57	1,61	1,13
Gromacs	128	7,09	0,99	0,62	
	256	9,54	1,33	0,84	
	512	13,49	1,89	1,18	
NAMD	128	20,63	2,88		
	256	26,44	3,69		
	512	40,03	5,59		
	Total Gbytes		715,67	1139,11	1619,53

Down Threshold, indicated with -P or -PH suffixes as described in Section 3.2.

Results on energy consumption at network level (E_{net} plots) indicate that the power saving mechanisms provide reductions for every configuration. All network topologies are highly sensitive to power saving solutions based on dynamically switching links on and off. Even though the network load may eventually require longer time to be delivered, energy saving is always achieved. The increase in time is compensated by power reduction.

Results on energy dissipation at system level ($E_{cluster}$ plots) indicate that energy savings at network level provide a positive impact on energy consumption by the entire system. The exception is Dragonfly topology when the default PDT is used, as indicated in Section 3.2. The poor performance achieved when links are placed into low-power mode and short PDT is used (10 μs, in our experiments, labeled with -P suffix) is due to the lack of adaptivity of the Dragonfly routing algorithm. Since the routing algorithm cannot choose a path based on the power status of the network links, the number of asleep ports in the chosen path increases, degrading the system performance and increasing the system energy consumption due to the higher execution times. That limitation is solved when using longer PDT for systems based on Dragonfly (100 μs, in our experiments, labeled with -PH suffix). By using a less aggressive value for PDT, the performance degradation is lower and therefore, the system energy consumption decreases, although the links are turned off less times than using a more aggressive PDT value.

Runtime results (*Runtime*) indicate, as expected, that energy saving solutions based on link powering-down increase the execution time. In our evaluation, all application mixes increase their runtime when network links are moved to a low-power state during idle periods. The time required to change link status, which prevents data from being sent, increases runtime. Nevertheless, the impact on energy consumption is always favorable. Although applications require longer time to complete, the overall system energy decreases.

To ease choosing the best configuration, Fig. 4 shows the runtime–energy relationship for the three analyzed scenarios for a 64-node system. This plot shows the trade-off among both figures of merit. The best configuration is the one that achieves the lowest execution time and consumes the lowest energy. In the Figure, it will be the downmost leftmost point of the plot. If there are several configurations that achieve a similar execution time (i.e. they are on the same value of the Y axis), the one with the lowest consumed energy (i.e., the lowest X value) will be the best choice. If there are several configurations that consumes a similar amount of energy (i.e. they are on the same value of the X axis), the one with the lowest execution time (i.e., the lowest Y value) will be the best choice.

Let us apply this analysis to the results shown in Fig. 4. For all the problem sizes analyzed (Set Low, Set Mid and Set High), the best configuration from the network energy point of view is the 80-node power-saving Dragonfly. It can be seen clearly that it has the lowest network energy consumption and an acceptable execution time. However, if we consider the system energy, the winner is the 3D-torus with power saving. It provides one of the lowest execution times, being the shortest execution time for sets Mid and High, and the best whole system energy figures.

The dots of the low-arity fat-tree ($k = 4$) are in the upper right area of the plots, and therefore to use this topology should be discarded. The fat-tree configuration with higher arity ($k = 8$) provides system energy results close to the torus despite longer runtime, due to its lower network energy consumption.

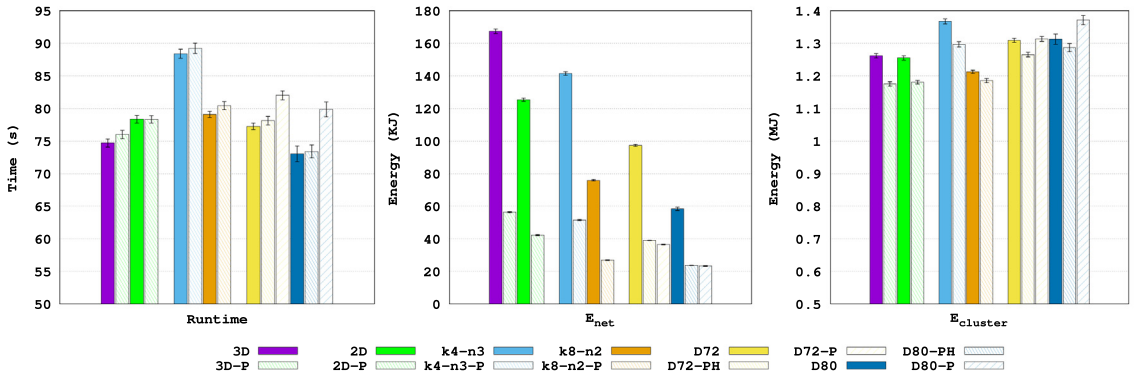
In general, the Dragonfly provides the worst energy consumption. As can be seen in Fig. 4, the Dragonfly points are the leftmost points of the plot, specially when the PDT value is 10 μs. This happens despite the runtime is acceptable, being the fastest system for Set Low, and having shorter runtime than the fat-tree in the remaining sets. The reason for those poor energy metrics is the excess in computing nodes required by the Dragonfly (72 or 80) versus torus and fat-tree. Despite the Dragonfly using more computing nodes, overall runtime is not reduced enough and energy consumption increases.

4.2. 256-node network evaluation

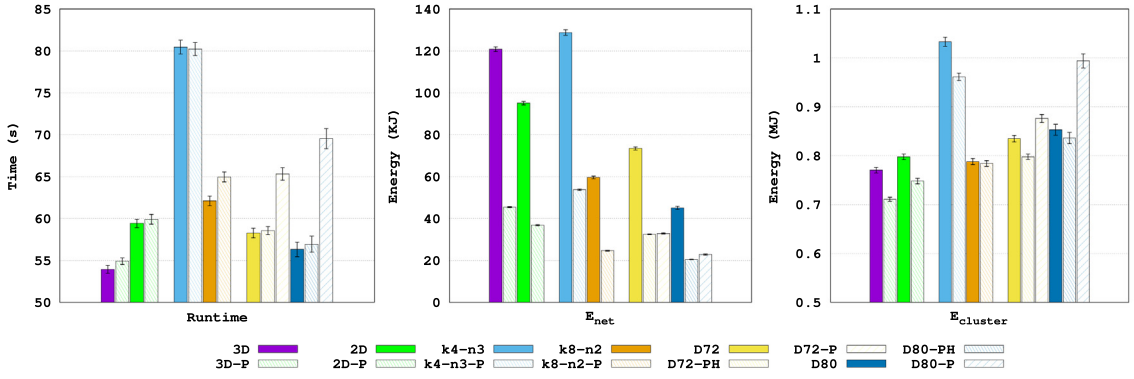
Fig. 5 shows *Runtime*, E_{net} and $E_{cluster}$ results for each topology, connecting at least 256 computing nodes. As in Fig. 3, every plot depicts three groups of contiguous bars, each one corresponding to the same topology (namely, torus, fat-tree and Dragonfly). Systems based on different network topologies have been tested with a power saving mechanism based on Low Power Idle and Power Down Threshold, indicated with -P or -PH suffixes as described in Section 3.2. For Dragonfly configuration only results for 100 μs PDT are shown (-PH suffix), since results for 10 μs PDT are significantly worse.

As for the experiments with 64 computing nodes, all network topologies benefit from using power saving mechanism since E_{net} is significantly reduced in all cases. The measured contribution of the interconnection network to the overall system energy consumption, across all tested configurations, ranges from 7 to 22%, when no power-saving mechanism is being utilized. This result matches estimations previously reported by Abts et al. [38].

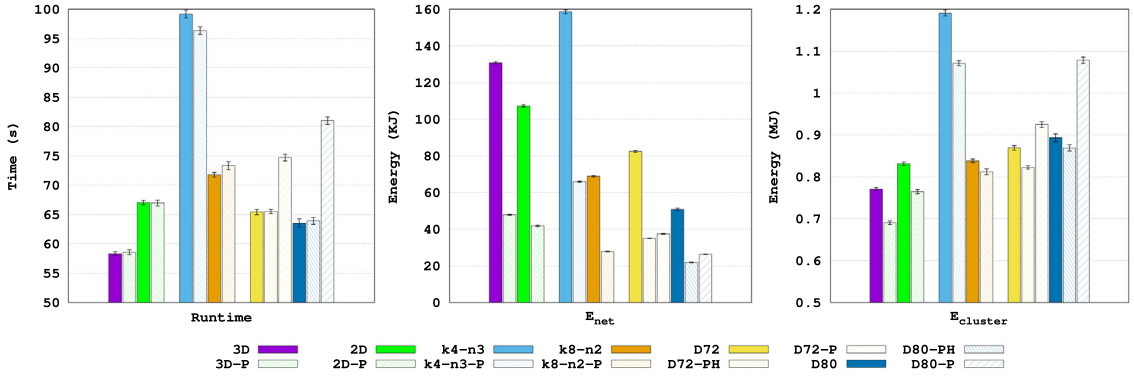
The impact of power saving mechanisms on runtime is low and valuable energy reductions are achieved at system level ($E_{cluster}$) for all our case studies. Indeed, for several cases, runtime marginally decreases when applying power saving mechanisms. For instance, 4D and 3D tori or 16-ary fat-tree with set Low. In these cases, for low loads, powering down several network links sometimes reduces runtime. The change in topology generated by disconnecting some links limits the adaptability of the routing algorithm, concentrating the traffic in fewer links and thus reducing the conflicts between packets, which reduces packet latency and in turn results in a reduction in execution time.



(a) Set Low



(b) Set Mid



(c) Set High

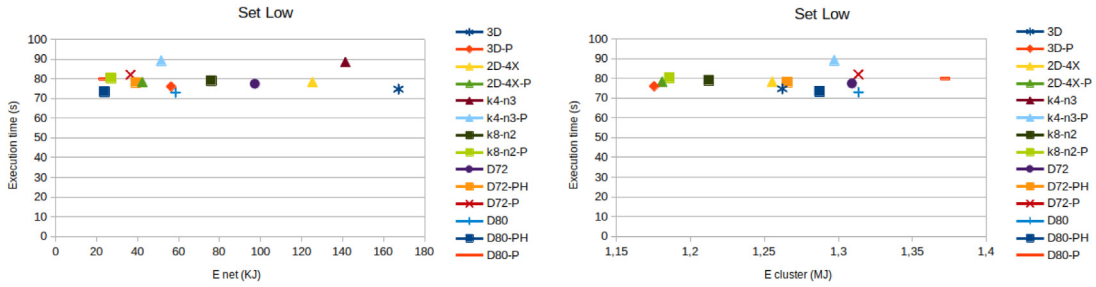
Fig. 3. Performance and energy evaluation for 64-node tori, fat-trees, and near-sized dragonflies.

At system level, again the torus topology with the higher number of dimensions provides the minimum energy consumption, for all the configurations tested. Only for set Low, fat-tree with $k = 16$ and Dragonfly with 272 nodes provide similar results. For the other sets, where network load is increased, the torus significantly outperforms fat-tree and Dragonfly in energy requirements.

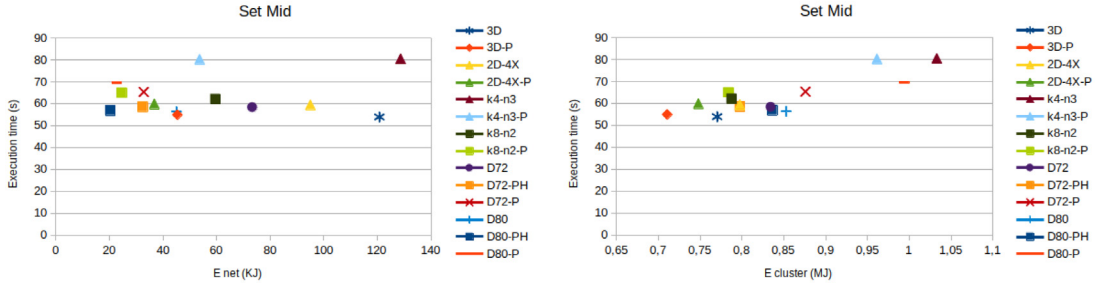
To ease choosing the best configuration, Fig. 6 shows the runtime–energy relationship for the three analyzed scenarios for a 256-node system. For the execution time–network energy relationship and the Set Mid and Set High problem sizes, the 342-node Dragonfly is the best choice. Although there are other faster networks, the increase of the energy consumption makes these options unworthy. This network also obtains good results for Set Low. The fat-tree with $k = 18$ also achieves a better trade-off between runtime and network energy.

Conversely, for the more interesting execution time–system energy relationship, the 256-node 4D torus with power saving is the absolute winner for Set Mid and Set High. Although the 342-node Dragonfly and the 320-node 4D torus are faster systems (around 5 $\sim$ 6% faster), the energy consumption reduction of the 256-node 4D torus is much more significant (around 10 $\sim$ 15%). However, 342-node Dragonfly and the 320-node 4D torus with power saving also offer a good trade-off between energy system consumption and execution time.

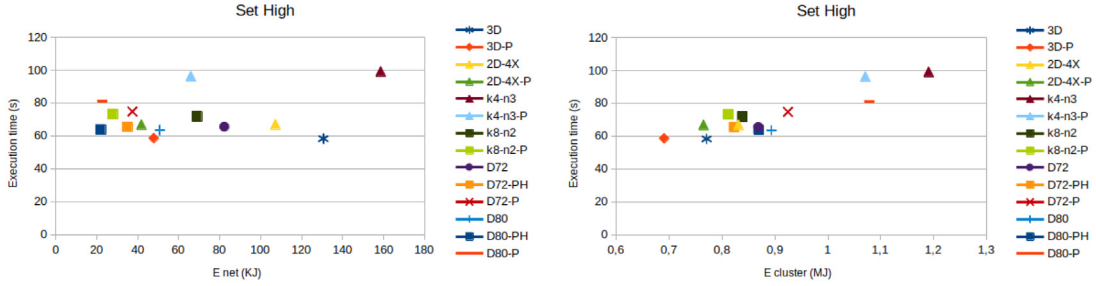
Interestingly, for low loads (Set Low), there is much more variability in the results. The best choice will depend on which parameter receives the most importance. To improve the performance, the best choices are the 18-ary fat-try and 342-node Dragonfly, both with power saving, meanwhile the most interesting option to improve the energy consumption are 272-node Dragonfly and the 256-node 4D torus, again with power saving.



(a) Set Low



(b) Set Mid



(c) Set High

Fig. 4. Execution time and energy trade-off for 64-node tori, fat-trees, and near-sized dragonflies.

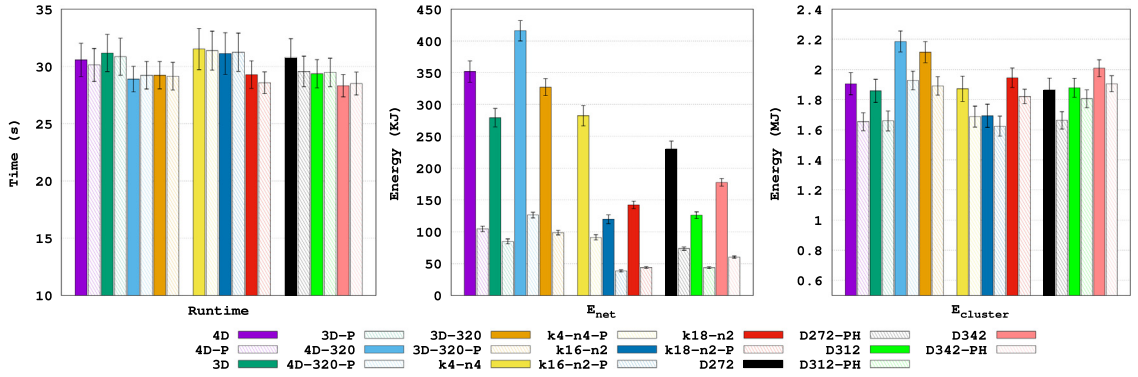
5. Related work

Various research works investigate energy consumption in high-speed interconnection networks, aiming at improving energy efficiency of HPC systems and data centers.

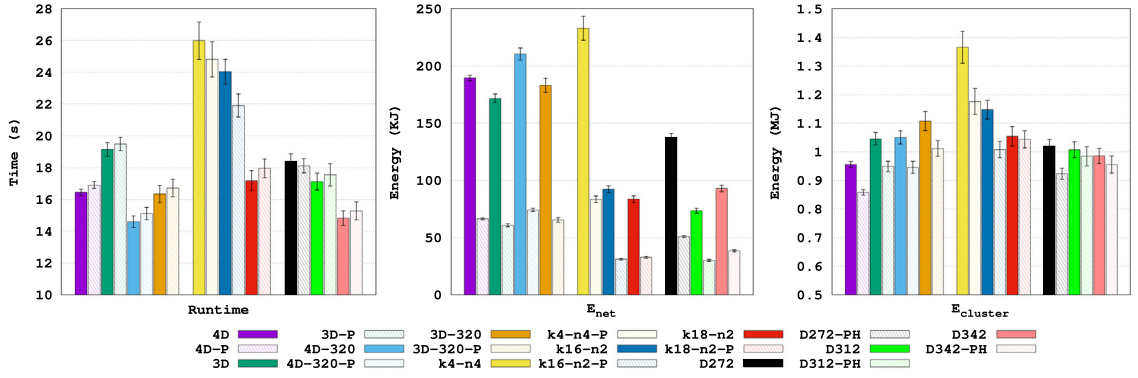
Kim et al. [39] evaluate dynamic voltage scaling and on/off based techniques on network link energy consumption. For a mesh topology, they show that link shutting down performs significantly better. Soteriou et al. [40] propose a dynamic power management (DPM) mechanism for mesh topologies. Their proposal relies on using a deadlock-free fully-adaptive routing algorithm that allows traffic redirection when links are turned off. These results show that link shutting down, supported by efficient network switch design, has potential to provide significant power saving with moderate impact on performance. Alonso et al. [41–43] target torus interconnect topologies based on aggregated links. Their works propose dynamically turning on and off network links as a function of traffic, while maintaining at least one active link per aggregated link. For very low loads on a single active link, link bandwidth is reduced to obtain additional power saving. An advantage to previous works is that network topology is not modified nor the routing algorithm. Similarly, additional work [44] extends the approach of shutting down redundant links to fat-tree systems. This proposal demonstrates that significant network power consumption reduction with limited impact on performance can be obtained.

Gunaratne et al. [45] estimate potential energy savings achievable by using adaptive link rate of Ethernet links (ALR). Their results, obtained by simulation with synthetic traffic patterns, show that an Ethernet link can operate at a low data rate most of the time yielding to significant energy savings with small impact on packet delay. ALR continues consuming power during idle periods, although at a reduced rate due to the slower link speed. Low Power Idle (LPI), proposed on the IEEE Energy-Efficient Ethernet (EEE) standard (IEEE 802.3az) [14], saves power by switching off transceiver components when a link is idle. LPI improves ALR because it requires less delay (microseconds versus milliseconds) and offers the maximum power savings: up to 90% less power consumption than fully active mode [46]. Although some studies have combined ALR with LPI, other works show how the combination of ALR and LPI introduces a significant performance degradation in terms of the QoS application receives (higher latency and jitter) which is not acceptable for end-users [47].

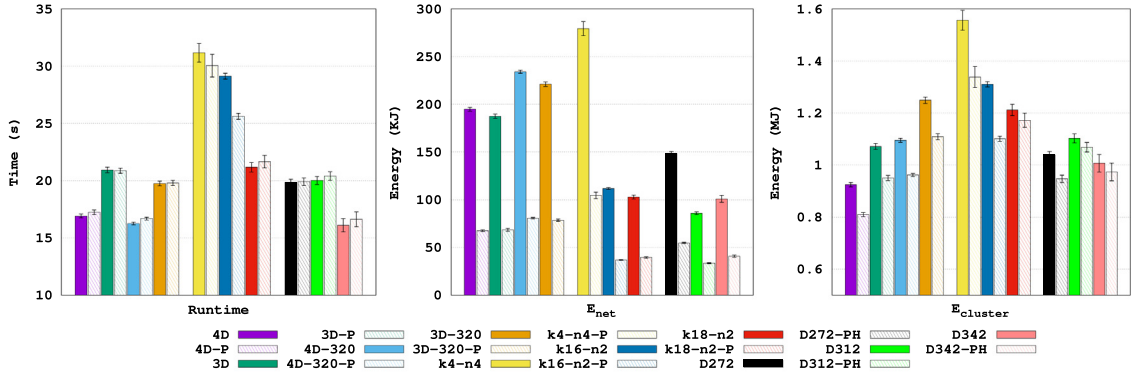
D. Abts et al. [38] show that high-performance interconnects consume a significant fraction of total system energy, in particular when servers operate at a fraction of their maximum utilization. Since servers typically run at low utilization levels and are increasingly becoming energy-proportional, this work highlights the need for more energy-proportional networks. Based on an analytical consumption model, they propose reducing link data



(a) Set Low



(b) Set Mid

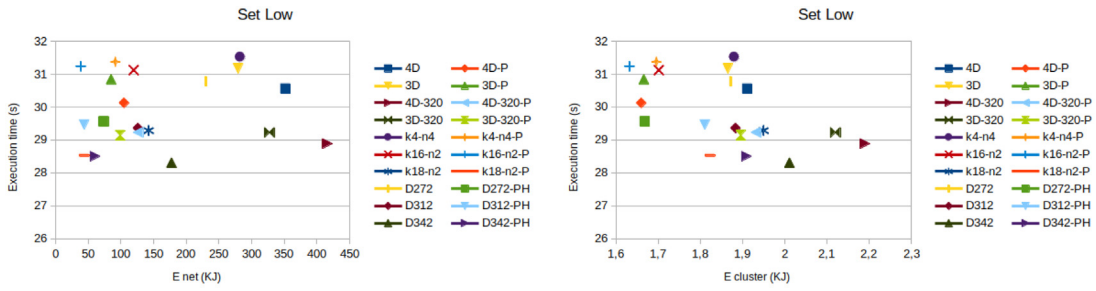


(c) Set High

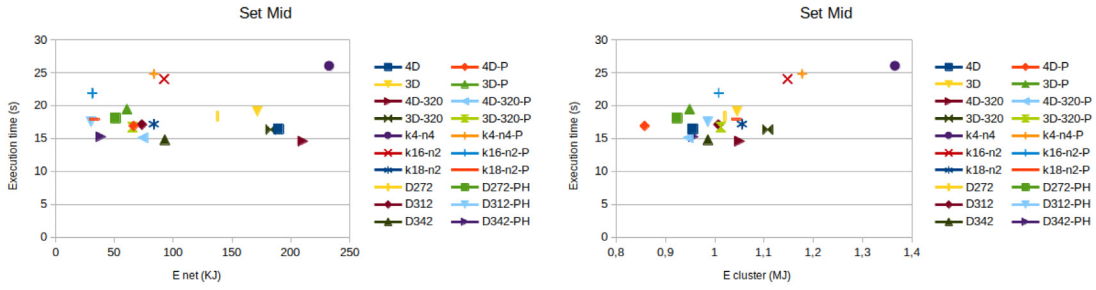
Fig. 5. Performance and energy evaluation for 256-node tori, fat-trees, and near-sized tori, fat-trees and dragonflies.

rates during periods of inactivity to save energy, but do not provide solutions. Saravanan et al. [48] propose a mechanism to reduce link energy consumption for Energy Efficient Ethernet. Their proposals rely on the implementation of variable length stall-timer, used by links to enter low-power states, that is dynamically set according to link statistics collected by NICs and switches and a performance overhead bound. Simulation results using traces for a hierarchical topology based on fat-tree show significant energy savings at link level with limited performance degradation.

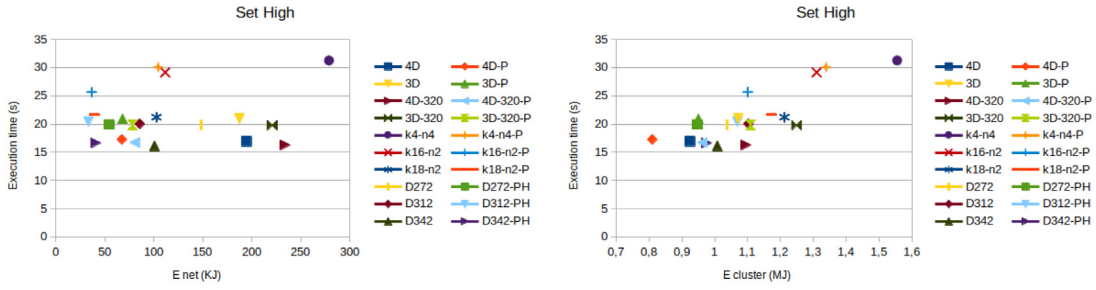
Zahn et al. [49] perform a comparative evaluation of different strategies for obtaining energy savings on torus, fat-tree and Dragonfly networks. They compare mechanisms that set link power states as a function of link utilization. However, this proposal only presents energy metrics at link level and does not discuss overall system performance. In a previous work [10], we presented a comparative performance and energy evaluation of different torus network configurations when link on/off power reduction mechanisms are used. Our simulations using real traces, evaluate total system energy consumption (including energy by the network fabric and the compute nodes) and application execution time. We show that using a power-saving mechanism always pay-off and that aggregated-link torus topologies provide the best trade-off between performance and energy consumption. In this paper, we extend this previous work by considering not only torus but also fat-trees and dragonflies thus allowing a comparison of the most widely used topologies. Indeed, the combined performance–energy plots presented in this paper allow an easy comparison of results and the selection of the best configuration according to either the criterion of performance-first or energy-first. On the other hand, this paper considers as power-saving mechanism the Low Power Idle proposed on the IEEE Energy Efficient Ethernet standard, thus allowing the utilization of the obtained results on commercial interconnection networks.



(a) Set Low



(b) Set Mid



(c) Set High

Fig. 6. Execution time and energy trade-off for 256-node tori, fat-trees, and near-sized tori, fat-trees and dragonflies.

6. Conclusion

We present an energy/performance study of HPC systems based on energy-efficient interconnects for realistic, multi-job trace-based workloads. To the best of our knowledge, this is the first research work where actual power consumption metrics measured on a real system are presented and used to configure the simulator. We conduct experiments on three widely used network topologies, namely torus, fat-tree and Dragonfly, applying low-power modes based on the Energy Efficient Ethernet standard.

Results show that all network topologies provide significant energy savings at network level when low-power modes are applied. At system level, overall energy consumption always decreases when the interconnection network implements Low Power Idle and Power Down Threshold, despite the increase in runtime. This demonstrates that implementing power saving strategies in interconnection networks is a good idea.

For the evaluated case studies, the torus topology with the highest number of dimensions provides lowest energy consumption at system level, and generally one of the shortest runtime, achieving in some cases the best performance (e.g. sets High and Low in the 64-node systems). Although torus topology consumes significantly higher energy at network level, when low-power mechanisms are applied the energy gets drastically reduced contributing to excellent overall system energy consumption. Fat-trees show good efficiency for low loads, but for medium and high loads provide the worst results both in terms of performance and energy. Dragonfly offers results close to torus. Overall, we find that torus topology provides the best energy-performance trade-off in all cases, followed by Dragonfly.

CRediT authorship contribution statement

Francisco J. Andújar: Methodology, Software Programming, Formal analysis, Investigation, Visualization. **Salvador Coll:** Conceptualization, Writing – original draft, Formal analysis. **Marina Alonso:** Conceptualization, Writing – original draft, Formal analysis. **Juan-Miguel Martínez:** Resources, Project administration, Writing – review & editing. **Pedro López:** Supervision, Project administration, Writing – review & editing, Visualization. **José L. Sánchez:** Methodology, Investigation, Writing – original draft. **Francisco J. Alfaro:** Resources, Project administration, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work has been supported by the *Spanish Ministerio de Ciencia e Innovación* (MICINN, formerly MINECO), and the European Commission (FEDER funds) under the projects PID2019-105903RB-100 and PID2021-123627OB-C5, and by *Junta de Comunidades de Castilla-La Mancha* under the project SBPLY/21/180501/000248.

References

[1] United Nations Sustainable Development Goals homepage, 2021, https://www.un.org/sustainabledevelopment, (Accessed December 20, 2021).
[2] TOP500 homepage, 2021, https://www.top500.org, (Accessed December 20, 2021).
[3] Y. Kodama, T. Odajima, E. Arima, M. Sato, Evaluation of power management control on the supercomputer fugaku, in: 2020 IEEE International Conference on Cluster Computing, CLUSTER, 2020, pp. 484–493.
[4] NVIDIA DGX SuperPOD webpage, 2021, https://www.nvidia.com/en-us/data-center/dgx-superpod/, (Accessed December 20, 2021).
[5] L.A. Barroso, U. Hölzle, The case for energy-proportional computing, Computer 40 (12) (2007) 33–37.
[6] A. Shehabi, S.J. Smith, D.A. Sartor, R.E. Brown, M. Herrlin, J.G. Koomey, E.R. Masanet, N. Horner, I.L. Azevedo, W. Lintner, United States Data Center Energy Usage Report, Tech. rep., 2016.
[7] C. Seitz, Concurrent VLSI architectures, IEEE Trans. Comput. C-33 (12) (1984) 1247–1265.
[8] F. Petrini, M. Vanneschi, K-ary N-trees: High performance networks for massively parallel architectures, in: Proceedings of 11th International of Parallel Processing Symposium, 1997, pp. 87–93.
[9] J. Kim, W.J. Dally, S. Scott, D. Abts, Technology-driven, highly-scalable dragonfly topology, in: ISCA '08: Proceedings of the 35th Annual International Symposium on Computer Architecture, IEEE Computer Society, Washington, DC, USA, 2008, pp. 77–88.
[10] F.J. Andújar, S. Coll, M. Alonso, J.M. Martínez, P. López, J.L. Sánchez, F.J. Alfaro, R. Martínez, Energy efficient torus networks with on/off links, J. Parallel Distrib. Comput. 130 (2019) 37–49.
[11] F.J. Andújar, S. Coll, M. Alonso, P. López, J.M. Martínez, POWAR: Power-aware routing in HPC networks with on/off links, ACM Trans. Archit. Code Optim. 15 (4) (2019).
[12] F. Guo, O. Ormond, M. Collier, X. Wang, Power measurement of NetFPGA based router, in: 2012 IEEE Online Conference on Green Communications (GreenCom), 2012, pp. 116–119.
[13] J.L. Sánchez, F.J. Alfaro, R. Galindo, F.J. Andújar, S. Coll, M. Alonso, J.M. Martínez, P. López, Power Consumption of HPC Applications, Tech. Rep. DIAB-21-02-1, Department of Computing Systems. Universidad de Castilla-La Mancha, 2021.
[14] K. Christensen, et al., IEEE 802.3az: the road to energy efficient ethernet, IEEE Commun. Mag. 48 (11) (2010) 50–56.
[15] P. Reviriego, J.A. Hernandez, D. Larrabeiti, J.A. Maestro, Performance evaluation of energy efficient ethernet, IEEE Commun. Lett. 13 (9) (2009) 697–699.
[16] K.P. Saravanan, P. Carpenter, A. Ramírez, Power/performance evaluation of energy efficient ethernet (EEE) for high performance computing, in: 2012 IEEE International Symposium on Performance Analysis of Systems & Software, Austin, TX, USA, 21-23 April, 2013, 2013, pp. 205–214.
[17] Hiperion repository homepage, 2021, https://gitraap.i3a.info/fandujar/hiperion, (Accessed December 20, 2021).
[18] VEF traces homepage, 2021, http://www.i3a.uclm.es/VEFtraces/, (Accessed December 20, 2021).
[19] F.J. Andújar, J.A. Villar, J.L. Sánchez, F.J. Alfaro, J. Escudero-Sahuquillo, VEF traces: A framework for modelling MPI traffic in interconnection network simulators, in: The 1st IEEE International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era, Chicago, IL, USA, 2015, pp. 841–848.
[20] H. Fröning, M. Nüssle, H. Litz, C. Leber, U. Brüning, On achieving high message rates, in: 2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, 2013, pp. 498–505.
[21] D. Chen, et al., The IBM Blue Gene/Q interconnection network and message unit, in: 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, 2011, pp. 1–10.
[22] S. Derradji, T. Palfer-Sollier, J.P. Panziera, A. Poudes, F.W. Atos, The BXI interconnect architecture, in: 2015 IEEE 23rd Annual Symposium on High-Performance Interconnects, 2015, pp. 18–25.
[23] M. Yokokawa, F. Shoji, A. Uno, M. Kurokawa, T. Watanabe, The K-Computer: Japanese next-generation supercomputer development project, in: Proceedings of the International Symposium on Low Power Electronics and Design, 2011, pp. 371–372.
[24] B. Alverson, E. Froese, L. Kaplan, D. Roweth, Cray XC Series Network, White Paper WP-Aries01-1112, Cray Inc., 2012.
[25] M. Karol, M. Hluchyj, Queuing in high-performance packet-switching, IEEE J. Sel. Areas 1 (1998) 1587–1597.
[26] T. Anderson, S. Owicki, J. Saxe, C. Thacker, High-speed switch scheduling for local-area networks, ACM Trans. Comput. Syst. 11 (1993) 319–352.
[27] N. Adiga, et al., Blue Gene/L torus interconnection network, IBM J. Res. Dev. 49 (2) (2005) 265–276.
[28] S.Q. Zheng, M. Yang, Algorithm-hardware codesign of fast parallel round-robin arbiters, IEEE Trans. Parallel Distrib. Syst. 18 (1) (2007) 84–95.
[29] J. Duato, S. Yalamanchili, L. Ni, Interconnection Networks. An Engineering Approach, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.
[30] W. Dally, C. Seitz, Deadlock-free message routing in multiprocessor interconnection networks, IEEE Trans. Comput. C-36 (5) (1987) 547–553.
[31] J. Duato, J. Flich, T. Nachiondo, A cost-effective technique to reduce HOL blocking in single-stage and multistage switch fabrics, in: 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing, 2004. Proceedings., 2004, pp. 48–53.
[32] E. Gabriel, et al., Open MPI: Goals, concept, and design of a next generation MPI implementation, in: Proceedings of the 11th European PVM/MPI Users' Group Meeting, 2004, pp. 97–104.
[33] GALGO - Supercomputer Center of Albacete Research Institute of Informatics homepage, 2021, https://www.i3a.uclm.es/i3a_t/galgo-supercomputing/, (Accessed December 20, 2021).
[34] HPC challenge benchmark homepage, 2021, http://icl.cs.utk.edu/hpcc/index.html, (Accessed December 20, 2021).
[35] J.C. Phillips, et al., Scalable molecular dynamics with NAMD, J. Comput. Chem. 26 (16) (2005) 1781–1802.
[36] S. Pronk, S. Páll, R. Schulz, P. Larsson, P. Bjelkmar, R. Apostolov, M.R. Shirts, J.C. Smith, P.M. Kasson, D. van der Spoel, B. Hess, E. Lindahl, GROMACS 4.5: a high-throughput and highly parallel open source molecular simulation toolkit, Bioinformatics 29 (7) (2013) 845–854.
[37] Graph500 homepage, 2021, https://graph500.org, (Accessed December 20, 2021).
[38] D. Abts, M.R. Marty, P.M. Wells, P. Klausler, H. Liu, Energy proportional datacenter networks, in: Proceedings of the 37th Annual International Symposium on Computer Architecture, ISCA '10, ACM, New York, NY, USA, 2010, pp. 338–347.
[39] E.J. Kim, K.H. Yum, G.M. Link, N. Vijaykrishnan, M. Kandemir, M.J. Irwin, M. Yousif, C.R. Das, Energy optimization techniques in cluster interconnects, in: Proceedings of the 2003 International Symposium on Low Power Electronics and Design, ISLPED '03, Association for Computing Machinery, New York, NY, USA, 2003, pp. 459–464.
[40] V. Soteriou, L.-S. Peh, Dynamic power management for power optimization of interconnection networks using on/off links, in: 11th Symposium on High Performance Interconnects, 2003, pp. 15–20.
[41] M. Alonso, J.M. Martínez, V. Santonja, P. López, Reducing power consumption in interconnection networks by dynamically adjusting link width, in: Lecture Notes in Computer Science, vol. 3149, Springer-Verlag, 2004, pp. 882–890.
[42] M. Alonso, J.M. Martinez, V. Santonja, P. Lopez, J. Duato, Power saving in regular interconnection networks built with high-degree switches, in: 19th IEEE International Parallel and Distributed Processing Symposium, 2005, p. 5b.
[43] M. Alonso, S. Coll, J.M. Martínez, V. Santonja, P. López, J. Duato, Power saving in regular interconnection networks, Parallel Comput. 36 (12) (2010) 696–712.
[44] M. Alonso, S. Coll, J.M. Martínez, V. Santonja, P. López, Power consumption management in fat-tree interconnection networks, Parallel Comput. 48 (C) (2015) 59–80.
[45] C. Gunaratne, K. Christensen, B. Nordman, S. Suen, Reducing the energy consumption of ethernet with adaptive link rate (ALR), IEEE Trans. Comput. 57 (4) (2008) 448–461.
[46] G. Georgakoudis, N. Jain, T. Ono, K. Inoue, S. Miwa, A. Bhatele, Evaluating the impact of energy efficient networks on HPC workloads, in: 26th IEEE International Conference on High Performance Computing, Data, and Analytics (HiPC), 2019.

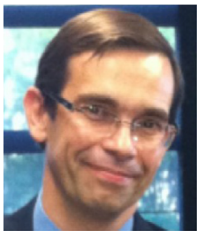
[47] T. Haudebourg, A. Orgerie, On the energy efficiency of sleeping and rate adaptation for network devices, in: Algorithms and Architectures for Parallel Processing - 17th International Conference, ICA3PP, in: Lecture Notes in Computer Science, vol. 10393, Springer, 2017, pp. 132–146.
[48] K.P. Saravanan, P. Carpenter, PerfBound: Conserving energy with bounded overheads in On/Off-based HPC Interconnects, IEEE Trans. Comput. (2018) 1.
[49] F. Zahn, A. Schoffer, H. Froning, Evaluating energy-saving strategies on torus, K-ary N-tree, and dragonfly, in: 2018 IEEE 4th International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB), 2018, pp. 16–23.

Francisco J. Andújar received the M.Sc. degree in Computer Science from the University of Castilla-La Mancha, Spain, in 2010, and the Ph.D. degree from the University of Castilla-La Mancha in 2015. He worked in the Universitat Politècnica de València under a post-doctoral contract Juan-de la Cierva, and currently works in the University of Valladolid as Associate Professor. His research interests include multicomputer systems, cluster computing, HPC interconnection networks, switch architecture and simulation tools.

Salvador Coll is Associate Professor of Electronic Technology at the Universitat Politècnica de València, Spain His research interests are in multicomputer systems and high-performance interconnection networks, with special interest in routing algorithms, collective communications and power efficiency. He received a B.Sc. degree in Electronic Engineering in 1992, a M.Sc. in Computer Science in 1996 and the Ph.D. degree from the Technical University of Valencia in 2005.

Marina Alonso is Associate Professor of Computer Architecture at the Universitat Politècnica de València, Spain. His research interests are in multicomputer systems and high-performance interconnection networks, with special interest in routing algorithms and power efficiency. He received a M.Sc. in Computer Science in 1993 and the Ph.D. degree from the Technical University of Valencia in 2012.

Juan-Miguel Martínez-Rubio received the B.Eng. degree in Electrical Engineering and the M.S. and Ph.D. degrees in Computer Engineering from the Universitat Politecnica de Valencia (UPV), Valencia, Spain, in 1986, 1993, and 1999, respectively. He has been with the Department of Computer Engineering, UPV, since 1986, where he is currently an Associate Professor. His research interests include Industrial Informatics, power saving, deadlock handling and congestion control mechanisms for computer interconnection networks. He has been involved in different projects in innovation in higher education since 1990.



Pedro López received the B.Eng. degree in Electrical Engineering and the M.S. and Ph.D. degrees in Computer Engineering from the Universitat Politécnica de Valéncia (UPV), Valencia, Spain, in 1984, 1990, and 1995, respectively. He is currently a Professor of Computer Architecture and Technology with the Department of Computer Engineering (DISCA), UPV. He has taught several courses on computer organization, computer architecture and computer clusters. His research interests mainly include high performance interconnection networks for multiprocessor systems and clusters. He has published over 130 refereed conference and journal papers.

José L. Sánchez received the Ph.D. degree from the Technical University of Valencia, Spain, in 1998. Since November 1986 he has been a member of the Computer Systems Department at the University of Castilla-La Mancha. He is currently Full Professor of Computer Architecture and Technology. His research interests include multicomputer systems, QoS in high-speed networks, interconnection networks, networks-on-chip, multicore architectures, parallel programming, and heterogeneous computing.

Francisco J. Alfaro received the M.Sc. degree in Computer Science from the University of Murcia in 1995 and the Ph.D. degree from the University of Castilla-La Mancha in 2003. He is currently Full Professor of Computer Architecture and Technology in the Computer Systems Department at the Castilla-La Mancha University. His research interests include high-performance networks, QoS, design of high-performance routers, and design of on-chip interconnection networks for multicore processors.