POWAR: Power-Aware Routing in HPC Networks with On/Off Links

FRANCISO J. ANDÚJAR, Universidad de Valladolid, Spain
SALVADOR COLL, MARINA ALONSO, PEDRO LÓPEZ, and JUAN-MIGUEL MARTÍNEZ, Universitat Politècnica de València, Spain

In order to save energy in HPC interconnection networks, one usual proposal is to switch idle links into a low-power mode after a certain time without any transmission, as IEEE Energy Efficient Ethernet standard proposes. Extending the low-power mode mechanism, we propose ***PO****W**er-**A**ware **R**outing* (*POWAR*), a simple power-aware routing and selection function for fat-tree and torus networks. *POWAR* adapts the amount of network links that can be used, taking into account the network load, and obtaining great energy savings in the network (55%–65%) and the entire system (9%–10%) with negligible performance overhead.

CCS Concepts: • **Networks** → **Routing protocols**; Network simulations; • **Hardware** → **Interconnect power issues**; *Networking hardware*; • **Computer systems organization** → *Interconnection architectures*;

Additional Key Words and Phrases: Switch architecture, high-performance computing, interconnection networks, power saving, routing algorithms

ACM Reference format:
Franciso J. Andújar, Salvador Coll, Marina Alonso, Pedro López, and Juan-Miguel Martínez. 2019. POWAR: Power-Aware Routing in HPC Networks with On/Off Links. *ACM Trans. Archit. Code Optim.* 15, 4, Article 61 (January 2019), 22 pages.
https://doi.org/10.1145/3293445

1 INTRODUCTION

The efficient use of natural resources is a growing concern that is being transferred to the different productive sectors. Accordingly, the design of a High Performance Computing (HPC) system and, more specifically, its interconnection networks do not ignore this issue. The interconnection network can contribute to around 10%~20% of the total HPC system energy consumption at full load (Abts et al. 2010; Greenberg et al. 2008). Therefore, all the actions that are carried out to achieve energy savings at a reasonable cost make the HPC more environmentally sustainable.

This work has been supported by the Spanish MINECO and European Commission (FEDER funds) under project TIN2015-66972-C5-1-R. Francisco J. Andújar has been partially funded by the Spanish MICINN and by the ERDF program of the European Union: PCAS Project (TIN2017-88614-R), CAPAP-H6 (TIN2016-81840-REDT), and Junta de Castilla y Leon - FEDER Grant VA082P17 (PROPHET Project).

Authors' addresses: F. J. Andújar, Departamento de Informática, Universidad de Valladolid, Edificio de Tecnologías de la Información, Paseo de Belén 15, Valladolid, 47011, Spain; email: fandujarm@infor.uva.es; S. Coll, Instituto de Instrumentación para Imagen Molecular (I3M), Universitat Politècnica de València, Camino de Vera s/n, 8B Building, Valencia, 46022, Spain; email: scoll@upv.es; M. Alonso, P. López, and J.-M. Martínez, Department of Computer Engineering, Universitat Politècnica de València, Camino de Vera s/n, 1G Building, Valencia, 46022, Spain; email: {malonso, plopez, jmmr}@upv.es.

1544-3566/2019/01-ART61
https://doi.org/10.1145/3293445

The European Commission has launched a plan to create an Energy Union in Europe and thus ensure that EU citizens and businesses have a secure, affordable, and climate-friendly energy supply. A more reasonable use of energy boosts growth and job creation while at the same time entailing an investment in the future of Europe.

Ethernet is one of the most widely used standards in interconnection networks. In September 2010, the Institute of Electrical and Electronics Engineers (IEEE) created the IEEE 802.3az standard that came to be called Green Ethernet (Energy Efficient Ethernet (EEE)) and whose objective was the reduction of energy consumption in Fast Ethernet, Gigabit Ethernet, and 10 Gigabit Ethernet links. The links are put in sleep mode (low-power idle (LPI)) when data is not transmitted. This leads to energy savings with some losses in performance. Refresh signals are transmitted periodically to maintain the synchronization of link endpoints. When there are data ready to be transmitted, a wake signal is sent to activate the links again. Some amount of time is required to transition between link states. For instance, for 10Gbps links, switching to LPI mode requires $2.88\mu s$ whereas for switching to normal mode $4.16\mu s$ are needed.

In the Top500 list of the most powerful supercomputers, Ethernet is the most popular interconnection in its different versions: 100G, 40G, 25G, 10G, or Gigabit. According to the June 2018 list, there are 247 systems of the 500 machines that use it. Three-quarters of these systems are located in China (Dongarra et al. 2018).

A direct implementation of the IEEE 802.3az standard shows poor performance as a result of the overheads by switching between wake-up and sleep states compared to actual communication time. For this reason, since the publication of the IEEE 802.3az standard, different research works have been developed with the aim of increasing energy savings, although the loss of performance is a determinant factor. The work of Saravanan et al. (2013) characterizes HPC applications over on/off links and proposes "Power-Down Threshold," a technique that reduces the performance overheads generated by transitions between link power modes. After transmitting a packet, the link remains in the active state until a threshold is exceeded before switching to low-power mode. Some previous works (Alonso et al. 2010, 2015; Soteriou and Peh 2003; Totoni et al. 2013) also proposed to switch links on and off to save power, mainly based on network utilization.

In this article, we propose to extend the standard operation of EEE with Power-Down Threshold by using a power-aware routing algorithm. However, note that our proposal is not only intended for the Ethernet technology, although it is strongly inspired in the EEE standard and the Power-Down Threshold. Our power-saving strategy could be applied in any HPC interconnection network, as long as the network technology supports on/off links and adaptive routing.

Specifically, our proposal is inspired by the mechanism proposed in Alonso et al. (2010, 2015) for torus and fat-tree topologies. While Alonso's proposal uses network utilization to switch links on and off, our novel routing algorithm uses network utilization to improve routing decisions, instead of directly changing link state, introducing power-awareness in the routing algorithm.

The rest of the document is organized as follows. In Section 2, we describe some background like the EEE standard, related work on power-saving strategies for HPC networks by using on/off links, and the assumed power consumption model. Section 3 presents a new routing algorithm to save power in interconnection networks, which will be referred to as ***POWer-Aware Routing*** (*POWAR*). After that, we analyze and discuss network performance and energy evaluation results in Section 4. Finally, in Section 5, we outline the conclusions.

2 BACKGROUND

In this section, we first discuss the motivation and operating principles of EEE. Then, we describe previously proposed power-saving algorithms based on the utilization of on/off links. Finally, we

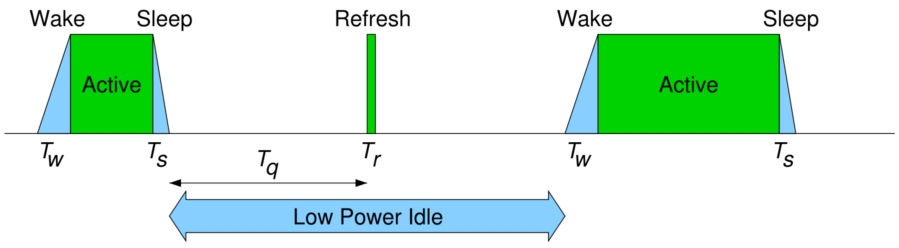


Fig. 1. Transitions between the active and low-power modes in EEE.

present the power model we devised to estimate the energy consumption of a computer system considering with special emphasis the contribution of the interconnection network.

2.1 Energy Efficient Ethernet: Low-Power Idle and Power-Down Threshold

Ethernet is the dominant wireline communication technology for LANs, and holds a prevalent position also in the high-performance computing field. For data rates higher that 100 Mb/s, Ethernet transmitters consume power continuously even when there are no data to transmit. An auxiliary signal called IDLE is used to keep transmitters and receivers properly aligned. As a consequence, a non-negligible amount of energy is wasted because most of the elements in the interfaces are consuming power at all times. The evolution of technology leading to higher data rates also implies that the energy waste will increase over time. A 1000BASE-T Ethernet physical layer transceiver (PHY) typically consumes over 0.5W while a 10GBASE-T PHY is usually over 5W (mains AC power consumption).

The huge amount of Ethernet devices and their widespread adoption mean that large energy savings will be obtained by reducing Ethernet devices energy consumption. For this purpose, in 2010, IEEE Std 802.3az-2010 EEE was approved. The standard defines a low-power mode to reduce power consumption when no data transmission occurs. LPI is used instead of the continuous IDLE signal to maintain links properly aligned during the low-power mode. EEE moves from the active to the low-power mode and vice versa as a function of link traffic. While in the active mode, when no further packets are available for transmission the link enters into the low-power (or "Sleep") mode. During the low-power mode, the energy consumption of the link is estimated to decrease to about 10% of its nominal value (Christensen et al. 2010). The key point is that moving links from active to sleep mode is relatively fast (in the order of microseconds), while no change is made on link speed. However, during transitions in and out of the low-power mode the energy consumption is significant, ranging from 50% to 100% of the active mode energy consumption.

Figure 1 illustrates EEE operating on a link that experiences a period of inactivity. When the transmission of a packet ends and no more packets are available for transmission, the link is switched to the low-power mode. During this low-power mode, the device sends periodic refresh signals to guarantee the link endpoints are aligned while the link stays quiet during large intervals. When packets need to be transmitted, the link is woken up to the active mode. During the transition times (T_s, T_w) and refresh time (T_r), the link consumes the same power as when it is active (as indicated above, this is a conservative assumption); and during the low-power mode (T_q), the link consumes about 10% of its nominal power. The basic limitation of the EEE standard is the overhead of switching links from the active to the low-power mode and back. For instance, considering a 10Gbps link, the transmission time of a 1500-byte frame is 1.2μs, while the sleep time is $T_s = 2.88\mu s$, and the wake-up time is $T_w = 4.16\mu s$, leading to a frame transmission efficiency of only 14.6%. As a consequence, EEE will typically work best for long transmission bursts alternating with long inactivity periods.

Preliminary EEE evaluation results (Christensen et al. 2010) showed that power saving in EEE links quickly decreases when link utilization increases. With link utilization at 20%, the power consumption is higher than 70%, exceeding 90% power for link utilization at 50%. This is mainly due to the overheads incurred in switching links from the active to the low-power modes and vice versa. The authors propose a packet coalescing technique to buffer multiple packets in the Ethernet interface to minimize the impact of link power state transitions. However, this technique increases packet delay and produces performance degradation of latency sensitive applications.

In Saravanan et al. (2013), the authors propose a "Power-Down Threshold" as an extension to EEE to cope with the overhead of link state transitions. After transmitting a packet, the link remains in the active state until a threshold time is elapsed before switching to low-power mode. This strategy tries to avoid unnecessary transitions between link power modes. Their experiments show that interconnect power saving of up to 70% with performance impact below 2% can be obtained on typical applications running on HPC platforms. In Saravanan and Carpenter (2018) PerfBound is proposed, a technique that also reduces link energy but imposing a bound on the application's performance degradation.

2.2 On/Off Power-Saving Strategies

Various research works propose strategies for switching links on and off to reduce the power consumed by interconnection networks. In Soteriou and Peh (2003), the authors propose a dynamic power management policy for mesh networks where network links are turned on and off depending on network utilization. Their proposal relies on three key mechanisms. First, a power-performance connectivity graph that indicates which links can be turned on/off and which links must remain always on. Second, a routing protocol that accounts for the state of the links and steers traffic accordingly. And third, input buffer utilization metrics drive the on/off decision policy using thresholds as link state transition deciders. Their results obtained by simulation using uniform random traffic indicate that on/off links can provide significant power saving.

Totoni et al. (2013) propose a runtime system-based approach to turn off unused links. The runtime system, at the network power management state, makes every node to calculate the route for each of its destinations. A notification is sent to the intermediate and destinations nodes to mark their used links. Finally every node turns off all its unused links. If the communication pattern between objects varies, the runtime system must detect it and the power management algorithm needs to be executed again. Their approach results in up to 20% savings of total system power for scientific applications with near neighbor communication.

In previous work, we propose dynamic power management for on/off links on torus with aggregated links (Alonso et al. 2010) and fat-tree networks (Alonso et al. 2015). Our mechanism does not require changes in the routing algorithm. We use link utilization as the metric to turn links on and off by defining two thresholds T_{on} and T_{off} (Figure 2) that fix the aggressiveness and the responsiveness of the power management policy. The thresholds dynamically change as a function of link state to avoid generating on/off oscillations and inducing network congestion. Our detailed evaluation of the thresholds impact using synthetic workloads shows great potential for power saving with limited impact on performance.

The work of Kim et al. (2018) proposes a traffic consolidation strategy for achieving energy-proportionality (TCEP) in high-radix networks, such as Flattened Butterfly (FBFLY), HyperX, or Dragonfly. Among other contributions, their mechanism defines a set of root links that are always connected by a star topology and concentrates active links to a small number of routers, creating hubs. Non-root links are power-gated considering the global impact on the network of re-routing minimal traffic vs. non-minimal traffic. Hence, TCEP prioritizes power-gating links with the least

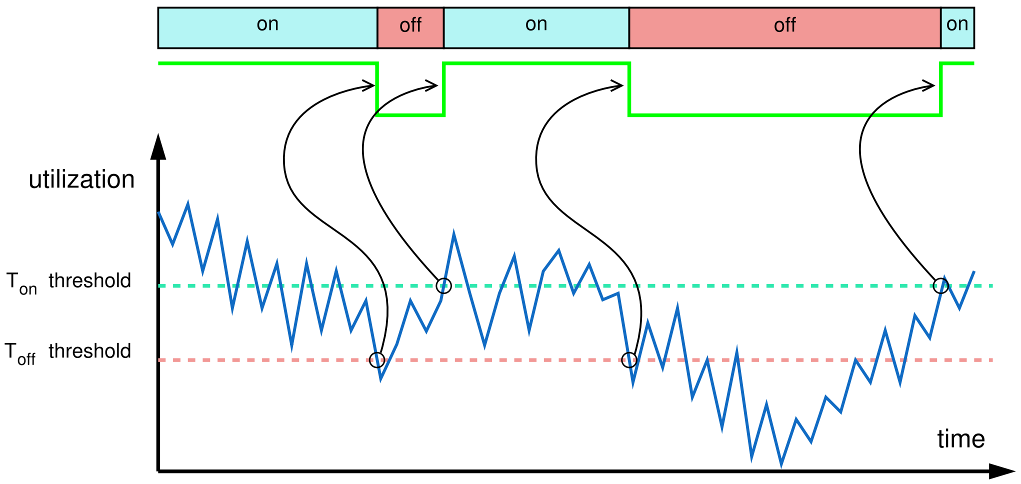


Fig. 2. Alonso's concept for dynamic on/off link switching.

amount of minimally routed traffic. TCEP link power-gating algorithm measures the utilization at every router and deactivates links whose traffic can be handled by other active links unused bandwidth (even using non-minimal routes). Although being close to *POWAR*, there are several significant differences between *POWAR* and TCEP: TCEP can not be applied to torus topologies with trunk links nor fat-tree topologies, where we tested *POWAR*; TCEP routing relies on non-minimal routing and can not be applied on minimally-routed networks, while *POWAR* can; TCEP limits the set of links that can be power gated, while *POWAR* does not set any restriction; TCEP requires additional virtual channels, and *POWAR* does not; TCEP authors only report network energy metrics and average flit latency while we provide whole cluster energy (network plus servers) metrics including application runtime to asses the impact of *POWAR* on performance.

Similar techniques exploiting the availability of high-degree switches have been used for networks-on-chip (NoC). Node-Router Decoupling (NoRD) (Chen and Pinkston 2012) decouples the ability of nodes to inject packets from the on/off status of routers by using bypass paths in the NoC. In multi-networks, traffic sources select to inject a packet into an active network or to activate a new network using oblivious (Camacho and Flich 2011) or adaptive (Das et al. 2013) metrics. An alternative technique is adaptive bandwidth networks (ABNs) (Michelogiannakis and Shalf 2014), which divide channels and switches into lanes for the network to provide the appropriate bandwidth at every hop. ABNs use power gated lanes, which consume near zero static power when inactive, take advantage of drowsy Static Random Access Memory (SRAM) cells that can be reactivated in a single cycle, and restrict packets to using a single lane per hop. Unlike multi-networks, in ABNs flits can select different lanes at each hop instead of committing to a given network at injection time. This restriction resembles the operation of the aggregated link proposal for HPC networks presented in Alonso et al. (2010), with the difference of much faster lane activation and deactivation times of NoC environments. MP3 (Chen et al. 2014) defines a strategy for power-gating for Clos topology on NoCs that resembles the proposal by Alonso et al. (2007) for fat-tree topology on HPC networks. Both approaches rely on defining a set of resources that are not power-gated to guarantee full network connectivity. The rest of the network elements are power-gated according to load conditions indicated by maximum buffer occupancy (Chen et al. 2014) or link utilization (Alonso et al. 2007).

2.3 Power Model

As mentioned in Section 1, the objective of this article is to demonstrate the power-saving ability of a novel power-aware routing algorithm by showing its impact on the energy consumption of an HPC platform (cluster or supercomputer) for different configurations of the interconnection network topology. In Andujar et al. (2018), we defined a power model to estimate system energy consumption. For the article to be self-contained, we include a minimal set of definitions and a brief explanation about the power model.[1]

Due to the relevance of the links in the performance and energy efficiency of the network, in order to determine the total power consumption of a switch, our model considers the power consumption of the links and the power consumption of the remaining switch logic. According to the state of the art, we consider the following general hypotheses:

- The switch power consumption increases linearly with the number of ports (Guo et al. 2012).
- As pointed in the previous section, a power-saving mechanism like LPI (Christensen et al. 2010) is assumed and, therefore, two states for the switch ports: *wake-up* (or *turned on*) and *sleep* (or *turned off*). Hence:
  - Since the transceiver is working regardless of the port is transmitting data or not, the port power consumption is 100% when it is turned on.
  - When the port is in sleep mode, it consumes a small part of the total power consumption.
  - During the transitions from one state to another, the port power consumption is 100%.

2.3.1 Definitions. We introduce the parameters we use 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, in Table 1.

2.3.2 Power Consumption Model. Here, we present the power consumption model. As we are going to carry out a comparative study, it is not relevant to use the absolute power consumption of each system component, but to determine the fraction of the total power consumed by each component.

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 only consumes ω^p_{Sport} of the total 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_{Sport} is the same for all ports, and the port power consumption is:

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

Next, we calculate the proportion of the switch power consumption with respect to its maximum power consumption. To simplify the model, we consider that the remaining switch logic consumes the maximum power consumption independently of its utilization, i.e., it consumes $(1 - \omega^s_{ports})$. Therefore:

$$W^s_{sw} = (1 - \omega^s_{ports}) + \omega^s_{ports} \cdot W^s_{ports} \tag{2}$$

[1]Note that the power model has been simplified regarding the one presented in Andujar et al. (2018). In previous work, we had compared network topologies with different number of switches and ports per switch. Therefore, we normalized all the results with respect to a "reference" network. In this article, the evaluated networks in each case study have the same network topology and normalizing the results is not necessary. For this reason, the mentions to the "reference" network have been removed in the power model presented in this article.

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)	
ω^{p}_{Sport}	Fraction of the power consumption that a port p consumes while in *sleep* mode
ω^{s}_{ports}	Contribution of the ports in a switch s to the total power consumption of that switch
ω_{net}	Contribution of the network to the total power consumption of the system.
ω_{Snodes}	Fraction of the power consumption that a node always consumes, regardless of its CPUs load
Variables obtained from simulation	
$RunTime$	Execution time of the HPC application
U^{p}_{port}	Fraction of $RunTime$ that a port p is turned on
U_{cpu}	Fraction of $RunTime$ that a CPU is running
Variables calculated by the power model	
W^{s}_{ports}	Power consumption of ports in switch s
W^{s}_{sw}	Power consumption of switch s
W_{net}	Power consumption of the network
W_{nodes}	Power consumption of the compute nodes
$W_{cluster}$	Power consumption of the cluster system
E_{net}	Energy consumption of the network
$E_{cluster}$	Energy consumption of the cluster system

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_{ports} is the same for all the switches. Therefore:

$$W_{net} = \frac{1}{N_{sw}} \sum_{i=1}^{N_{sw}} W^{i}_{sw} = \frac{1}{N_{sw}} \sum_{i=1}^{N_{sw}} ((1-\omega_{ports}) + \omega_{ports} \cdot W^{i}_{ports}) = (1-\omega_{ports}) + \frac{\omega_{ports}}{N_{sw}} \sum_{i=1}^{N_{sw}} W^{i}_{ports} \tag{3}$$

In order to obtain the total energy of the HPC platform, we need to consider the power consumption of the computing subsystem, mainly due to the computing nodes. Considering that all the nodes have the same characteristics, the fraction of the node power consumption with respect to their maximum power consumption, can be expressed as:

$$W_{nodes} = \omega_{Snodes} + (1-\omega_{Snodes}) \cdot \frac{1}{N_{nodes}} \sum_{i=1}^{N_{nodes}} U^{i}_{cpu}, \tag{4}$$

where we assume that the computing nodes are energy proportional, but also there is a fixed part of the total power consumption that is always consumed. The rest of the power consumption is proportional to the CPU utilization.

Then, we can calculate the HPC platform power consumption:

$$W_{cluster} = \omega_{net} \cdot W_{net} + (1-\omega_{net}) \cdot W_{nodes} \tag{5}$$

$W_{cluster}$ provides the fraction of the maximum power (both network and nodes) consumed during the application execution and normalized with respect to its maximum power consumption.

Table 2. Power Model Parametrization

Parameter	ω_{Sport}	ω_{ports}	ω_{net}	ω_{Snodes}
Value	0.1	0.65	0.15	0.5

Table 3. Simulation Values for Power Model Example

Network	$RunTime$	U^0_{port}	U^1_{port}	U_{cpu}
ref	650,000	1	1	0.800
pow	685,000	0.7	0.8	0.750

Finally, the energy consumed by an application is:

$$E_{net} = W_{net} \cdot RunTime \tag{6}$$

$$E_{cluster} = W_{cluster} \cdot RunTime \tag{7}$$

2.3.3 *Parameter Characterization.* Table 2 summarizes the selected values for the parameters that determine the fraction of power dissipated by the various network building blocks, as defined in our model.

According to the IEEE EEE standard (IEEE 802.3az), the power consumption of an idle link[2] is estimated to be 10% of the link power consumption (Christensen et al. 2010; Reviriego et al. 2009). Therefore, we set ω_{Sport} to 0.1.

The weight of the link power consumption with respect to the switch power consumption is 65% and 63% for the Dell PowerConnect 5324 (24-port switch) and the Dell PowerConnect 6248 (48-port switch) (Koibuchi et al. 2009), respectively; 64% for an IBM Infiniband 8-Port 12X switch (Li et al. 2011) and 68% for the EXTOLL Tourmalet switch (Zahn et al. 2016). According to that, we consider that 0.65 is a realistic estimation for ω_{ports}.

Finally, we set ω_{net} to 0.15, since the network power consumption is 10%~20% (Abts et al. 2010; Greenberg et al. 2008) of the full system and ω_{Snodes} to 0.5, since even energy-efficient servers still consume half of their power while idle (Barroso and Hölzle 2007).

2.3.4 *Power Model Example.* Finally, we include a example of the use of the power model. Let's suppose a network with two switches. We want to determine the performance/energy impact of a hypothetical power-saving strategy based on turning on/off the switch links. For this purpose, we execute by simulation a Message Passing Interface (MPI) application in: (i) the original system (*ref* configuration); and (ii) the same system after implementing the power-saving strategy (*pow* configuration). Table 3 shows $RunTime$, U_{port}, and U_{cpu} obtained by simulation for both configurations. Note that the table directly shows the average value of U_{port} per switch and the average value of U_{cpu} for each system.

Since there is no power saving in the *ref* network, its network power consumption is the maximum, i.e., $W^{ref}_{net} = 1$. Then, we only need to calculate W^{ref}_{nodes} and $W^{ref}_{cluster}$, using Equations (4) and (5), respectively:

$$\begin{aligned} W^{ref}_{nodes} &= \omega_{Snodes} + (1 - \omega_{Snodes}) \cdot U^{ref}_{cpu} = 0.5 + (1 - 0.5) \cdot 0.8 = 0.9 \\ W^{ref}_{cluster} &= \omega_{net} \cdot W^{ref}_{net} + (1 - \omega_{net}) \cdot W^{ref}_{nodes} = 0.15 \cdot 1 + (1 - 0.15) \cdot 0.9 = 0.915, \end{aligned} \tag{8}$$

that is, the power consumption of the reference system is 0.915 of the maximum power consumption of the system. Now we must calculate W_{net} and $W_{cluster}$ for *pow* configuration. Then, we

[2]Note that, in our terminology, an idle link is equivalent to an off link, while an active link is equivalent to an on link.

calculate W_{net}^{pow} using Equation (3):

$$W_{net}^{pow} = (1 - \omega_{ports}) + \frac{\omega_{ports}}{N_{sw}} \sum_{i=1}^{N_{sw}} W_{ports}^{i} = (1 - 0.65) + 0.65 \cdot \frac{0.7 + 0.8}{2} = 0.8375 \tag{9}$$

After that, we calculate W_{nodes}^{pow} and $W_{cluster}^{pow}$ using Equations (4) and (5):

$$W_{nodes}^{pow} = \omega_{Snodes} + (1 - \omega_{Snodes}) \cdot U_{cpu}^{pow} = 0.5 + (1 - 0.5) \cdot 0.75 = 0.875$$

$$W_{cluster}^{pow} = \omega_{net} \cdot W_{net}^{pow} + (1 - \omega_{net}) \cdot W_{nodes}^{pow} = 0.15 \cdot 0.8375 + (1 - 0.15) \cdot 0.875 = 0.869375 \tag{10}$$

Now we have calculated all the power consumption variables for *ref* and *pow* configurations, we can calculate the energy results using the *RunTime*. Since we want to calculate the impact of the power-saving strategy, we normalized the results with respect to the *ref* network. First, we calculate the normalized *RunTime* of the *pow* network:

$$Normalized \quad RunTime = \frac{RunTime^{pow}}{RunTime^{ref}} = \frac{685000}{650000} = 1.054$$

Therefore, the power-saving strategy approximately increases 5% the *RunTime*. Now we calculate the normalized E_{net} of the *pow* network using the results obtained in Equation (9):

$$Normalized \quad E_{net}^{pow} = \frac{E_{net}^{pow}}{E_{net}^{ref}} = \frac{W_{net}^{pow} \cdot RunTime^{pow}}{W_{net}^{ref} \cdot RunTime^{ref}} = \frac{0.8375 \cdot 685000}{1 \cdot 650000} = 0.8826,$$

and we calculate the normalized $E_{cluster}^{pow}$ using the results obtained in Equations (8) and (10):

$$Normalized \quad E_{cluster}^{pow} = \frac{E_{cluster}^{pow}}{E_{cluster}^{ref}} = \frac{W_{cluster}^{pow} \cdot RunTime^{pow}}{W_{cluster}^{ref} \cdot RunTime^{ref}} = \frac{0.869375 \cdot 685000}{0.915 \cdot 650000} = 1.0013$$

Therefore, this power-saving strategy does not fulfill its purpose. Although the network energy consumption is approximately reduced 12%, the cluster energy consumption is practically the same (0.1% greater than *ref* network) due to the performance degradation (≃5%). Then, this hypothetical power-saving strategy is not worth it.

3 POWER-AWARE ROUTING

The main goal of this work is to minimize network energy consumption in interconnection network technologies that support on/off links, like EEE with LPI (Christensen et al. 2010). Moreover, we also want to minimize the performance overheads generated by turning links on and off, and to achieve it through a simple routing mechanism, which will be referred to as *POWAR*. We present *POWAR* for two of the most popular topologies used in HPC: the k-ary n-cube (a.k.a. torus topology) and the k-ary n-tree (a.k.a. fat-tree topology). *POWAR* is based on popular adaptive routing algorithms for this topologies.[3]

An adaptive routing algorithm can be decomposed into two functions: the routing function and the selection function (Duato et al. 2003). The routing function supplies a set of feasible output channels based on the source and the destination node. After that, the selection function chooses one channel from this set based on the network status and the routing strategy.

In the fat-tree topology, the routing function consists of two phases: an ascending phase to a nearest common ancestor switch; and a descending phase to the destination. During the ascending phase, all the ports in the ascending direction can be chosen by the selection function, while the

[3]Deterministic routing algorithms drastically minimize the opportunities to reduce energy consumption by turning off unused links since the path between every source-destination pair is fixed.

message is self-routed to the destination following a deterministic path during the descending phase (Duato et al. 2003).

Regarding the torus topology, we have implemented a minimal-path fully-adaptive routing algorithm using Duato's Protocol (Duato 1993), since this methodology is widely used in HPC interconnection networks (Chen et al. 2011; Fröning et al. 2013). Briefly, this methodology splits each physical link into several virtual channels (VCs). A set of them (i.e., adaptive VCs) are used by the packets to traverse the network switches in any order. The remaining VCs are used taking into account the restrictions imposed by a deadlock-free routing algorithm. If a packet can not go through the adaptive VCs, the deterministic VCs provide an "escape path" that guarantees deadlock freedom.

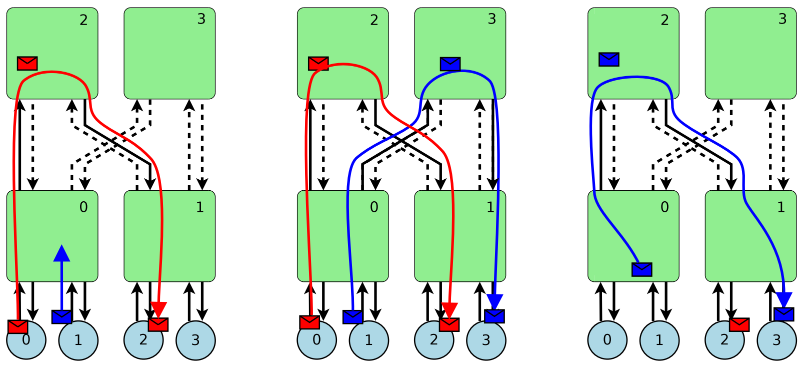
In particular, we use the adaptive VCs to implement a minimal adaptive routing; i.e., all the VCs that bring the packet closer to its destination can be selected. Regarding the deterministic subnetwork, we use dimension-order routing (DOR) with two VCs to avoid deadlocks (Dally and Seitz 1987). Although any number of VCs could be used, we will use four VCs per physical channel: 2 deterministic VCs + 2 adaptive VCs. As a consequence, the routing function returns a set of adaptive ports/VCs and one escape link/VC.

As stated above, the selection function must also be defined. This is the point where *POWAR* works in order to take a good choice that minimizes both energy consumption and performance penalty. Note that *POWAR* does not turn on/off switch ports: the decision of turning on/off a port is the sole responsibility of the link on/off mechanism. In our case, we have used the mechanism proposed by EEE and described in Section 2.1 (LPI with Power-Down Threshold).

Algorithm 1 shows the *POWAR* selection function. Notice that the ports are identified with an integer from 0 to ($Num\_Ports - 1$). If the selection function returns the keyword *NULL*, it means that there is no available port for transmitting the corresponding packet. We have chosen round-robin arbitration as the selection function, modifying it to be power-aware. Round-robin arbitration is a straightforward function that tries to balance network traffic. However, round-robin arbitration presents some problems in a network with on/off links. In the first place, if there are wake-up and sleep ports in the set returned by the routing function, the selection function could choose a sleep port despite having free wake-up ports available. That would add extra latency to the packet that could be avoided. For this reason, *POWAR* prioritizes wake-up ports: a sleep port will be chosen only when there are no wake-up ports in the port set or all the wake-up ports are busy.

Secondly, although there were no free wake-up ports, selecting a sleep port could have a negative impact on performance because waking it up might not compensate. Let's consider the 2-ary 2-tree shown in Figure 3. Note that sleep links are represented with dotted lines, while wake-up links are represented with solid lines. As seen in Figure 3(a), node 0 is transmitting several packets to node 2 through switch 2, when a message from node 1 to node 3 arrives at switch 0. In switch 0, since the port to switch 2 is busy, the switch wakes up its second port and sends the message through switch 3 (Figure 3(b)). However, the ports of switch 3 are also in sleep state, and therefore, another link will be woken up, increasing power consumption. On the other hand, waking up a port may require much more time than transmitting a single packet. Time and power consumption could be saved if the switch waits for finishing the previous transmission and sends the new message through switch 2 after that (Figure 3(c)). However, the message can not indefinitely wait. If node 0 is transmitting a large burst of messages to node 2, the new message could suffer a greater latency penalty due to starvation than the delay for reaching its destination through switch 3 after waking up sleep ports.

As a consequence, we must: (i) avoid waking up unnecessary resources in a situation with low network traffic (Figure 3(b)); and (ii) wake up free resources under high network loads. For this reason, we define a "selectable" state for switch ports. The selection function can only choose a port if it is marked as "selectable." To determine which ports are selectable or not, the switch



(a) A message arrives while node 0 is sending messages to node 2

(b) The message wakes up a new port to switch 3

(c) The message waits until the first message frees the port to switch 2

Fig. 3. Routing examples.

takes into account the switch load, adapting the mechanisms proposed by Alonso et al. for fat-tree topologies (Alonso et al. 2015) and torus topologies with aggregated links (Alonso et al. 2010).

The operation of the selectable port setting is based on a pair of thresholds that determine when a port is eligible for routing (selectable) or not (not selectable). The original concept, as highlighted in Figure 2, was devised to control link power-gating. Link utilization higher than the on threshold (T_{on}) indicates a new port (if available) must be powered on, while utilization lower than the off threshold (T_{off}) indicates a port must be powered down (if full network connectivity is preserved). In *POWAR*, the dual threshold concept does not generate power-gating actions; in contrast, it is used to indicate if a port is selectable or not and, later, the selection function uses this information to perform power-aware decisions in coordination with the routing function, as described below. Eventually, idle links resulting by the power-aware routing will be switched off by means of a power-down mechanism similar to that defined by the EEE standard.

The following restrictions hold in the selection of T_{on} and T_{off}:

— $T_{\text{off}} > 0$
— $T_{\text{on}} < U_{\text{MAX}}$, being U_{MAX} the maximum link utilization for a given topology under uniform traffic. This limitation avoids generating network congestion since new links are set to selectable, and hence elegible for routing, before exceeding available network capacity.
— $T_{\text{on}} \geq 2T_{\text{off}}$, to avoid selectable setting cycles for any given link. When a port is set as not selectable the traffic through alternative routes will increase and that could trigger a decision to set the port as selectable (if the average utilization increases over the on threshold), repeating again the sequence in a loop. Conversely, when a port is set as selectable, the traffic through alternative routes will decrease, inducing a similar situation if the average utilization decreases below the off threshold.

Apart from that, a peculiarity of our utilization thresholds implementation is that they can be tuned to provide different aggressiveness and sensitivity settings:

— Mechanism aggressiveness: this is controlled by the average value of the thresholds, $T_{\text{avg}} = (T_{\text{on}} + T_{\text{off}})/2$. High average thresholds provide an aggressive policy, since the power-aware routing algorithm includes new alternative routes when current ones reach high utilization. On the other hand, if T_{avg} is low, a conservative policy is applied since additional routes are considered even with low loads. As a result, the power-gating mechanism pursues higher or lower energy saving, respectively.

— Mechanism responsiveness: the hysteresis band, defined by the difference of the thresholds, T_{on}-T_{off} controls the routing algorithm responsiveness against traffic variations. Higher hysteresis bands will require higher traffic variations for the routing algorithm to modify the set of possible routes, while lower traffic variations will not modify the available routes, and vice versa.

As seen in Algorithm 1, the selection function first tries to find an adaptive output for forwarding the packet. If no adaptive output is found, then the selection function checks the deterministic output and it also checks if the deterministic output is "selectable." This can be counterintuitive since the selection function is shared for fat-tree and torus networks with aggregated links, which have disparate routing functions, but it is easy to explain:

— **Torus** topology: The selection function first searches for an adaptive channel; if no adaptive channels are found, it checks the deterministic output. But the deterministic output returned by the routing function is an aggregated link that comprises several ports. In the same way as in the adaptive channels, the algorithm must check which ports of the deterministic aggregated link are set as selectable or not. Note that there is always at least one selectable port per aggregated link, as will be seen in Section 3.2. Indeed, to choose among the ports of an aggregated link, another arbitration must be performed by the selection algorithm, but we have omitted this latter arbitration for the sake of simplicity.

— **Fat-tree** topology: We must distinguish between the ascending and the descending phase:

  — *Ascending* phase: All the ascending ports can be selected, but there is no deterministic output. Therefore, the routing function returns $\mathcal{P} \neq \emptyset$ and $\mathcal{D} = NULL$, and the selection function only executes the adaptive part of the pseudo-code.

  — *Descending* phase: The routing is deterministic (self-routing), and there are no adaptive channels. Therefore, the routing function returns $\mathcal{P} = \emptyset$ and $\mathcal{D} \neq NULL$, and the selection function only executes the deterministic part of the pseudo-code.

3.1 Selectable Port Management on Fat-trees

Algorithm 2 shows the management of the selectable ports on the fat-tree. Notice that the descending links are always selectable (as the descending phase is deterministic), and there is always one ascending link in selectable status. Regarding port identification, the descending ports are numbered from 0 to $(k-1)$, while the ascending ports are numbered from k to $(2k-1)$. Each switch requires a counter that stores the number of flits forwarded to the upper stage. Switches periodically check their counter and calculate the utilization of the current selectable ports. If the utilization is greater than the on threshold, called T_{on}, the first non-selectable port is set as selectable. If the utilization is lower than the off threshold, called T_{off}, and there are two or more selectable ports, the last selectable port is set as non-selectable. After that, the flit counter is reset.

3.2 Selectable Port Management on Torus with Aggregated Links

In the torus network with aggregated links, the selectable port management algorithm is similar to the fat-tree algorithm, but the switches require one flit counter for each aggregated link. Using the aggregated link flit counter, the switch calculates the utilization of each aggregated link. Then, if the utilization is greater than T_{on}, the first non-selectable port of the aggregated link is set as selectable, while if the utilization is lower than T_{off}, and there are at least two selectable ports, the last selectable port of the aggregated link is marked as non-selectable. After that, all the flit counters are reset. Algorithm 3 shows the selectable port management on torus networks. Note that, in the torus case, the ports of an aggregated link i are identified from $(Ports\_per\_agg\_link * i)$ to

ALGORITHM 1: *POWAR* selection function

```
Input: A set of adaptive ports $\mathcal{P} = (\{p_1, p_2, \ldots, p_r\}$ xor $\emptyset)$ given by the routing function,
       being $0 \le r, p_r < Num\_Ports$
       Deterministic port $\mathcal{D}$, being $0 \le \mathcal{D} < Num\_Ports$
       the round-robin pointer $rrp$, being $0 \le rrp < Num\_Ports$
Output: The selected port, the round-robin pointer $rrp$
/* Are there adaptive ports in the set given by the routing function?              */
if $\mathcal{P} \ne \emptyset$ then
    $p_{sleep} := NULL$
    for $i := 0$ to $Num\_Ports - 1$ do
        /* Check all the ports using $rrp$ to go through the ports in the correct order  */
        $p_{eval} := (rrp + i)\%Num\_Ports$
        /* Check if $p_{eval}$: (i) belongs to the adaptive port set;                  */
        /* (ii) is not busy (is not currently transmitting a packet); (iii) is a
           selectable port                                                            */
        if $p_{eval} \in \mathcal{P}$ and $p_{eval}$ is not busy and $p_{eval}$ is a selectable port then
            if $p_{eval}$ is wake-up then
                /* If $p_{eval}$ is wake-up, update the round-robin pointer and return $p_{eval}$  */
                $rrp := (p_{eval} + 1)\%Num\_Ports$
                return $p_{eval}$
            else if $p_{sleep} = NULL$ then
                /* If $p_{eval}$ is sleep and if the first sleep port checked, set $p_{sleep}$    */
                $p_{sleep} := p_{eval}$
            end
    end
    /* No wake-up adaptive ports found. If $p_{sleep}$ is set, update the round-robin pointer
       and return $p_{sleep}$                                                          */
    if $p_{sleep} \ne NULL$ then
        $rrp := (p_{sleep} + 1)\%Num\_Ports$
        return $p_{sleep}$
    end
end
/* No adaptive ports found. Check the deterministic output (if there is a deterministic
   output)                                                                             */
if $\mathcal{D} \ne NULL$ and $\mathcal{D}$ is not busy and $\mathcal{D}$ is a selectable port then
    return $\mathcal{D}$
end
/* Return $NULL$ value to indicate that valid ports are not found                      */
return $NULL$
```

$(Ports\_per\_agg\_link * (i + 1) - 1)$, $Ports\_per\_agg\_link$ being the number of ports per aggregated link. For example, if $Ports\_per\_agg\_link = 4$, the ports of the aggregated link 0 are identified from 0 to 3; the ports of the aggregated link 1 are identified from 4 to 7, and so on.

4 PERFORMANCE EVALUATION

In this section, we show the performance/energy evaluation of *POWAR*. The evaluation has been performed using an interconnection network simulator, fed by VEF traces (Andújar et al. 2015; VEF 2017), an open-source framework to simulate MPI applications. Using the simulator, we

ALGORITHM 2: Fat-tree selectable port management

```
Input: Set of switch ascending ports $\mathcal{P} = \{p_k, p_{k+1}, \ldots, p_{2k-1}\}$
       Number of selectable ports $Num\_Sel\_Ports$
       Flit counter $FC$
       Maximum number of flits transmitted per port in a checking cycle $max$
       Threshold for adding selectable ports $T_{on}$
       Threshold for removing selectable ports $T_{off}$.
/* Obtain the up-port utilization taking into account the number of the current
   selectable ports                                                          */
$utilization := FC/(Num\_Sel\_Ports * max)$
/* Is the utilization greater than $T_{on}$? Is adding a new selectable port possible?   */
if $utilization > T_{on}$ and $Num\_Sel\_Ports < k$ then
    /* Set the first non-selectable up-port as selectable                    */
    $i := k + Num\_Sel\_Ports$
    $SetAsSelectable(p_i)$
    $Num\_Sel\_Ports := Num\_Sel\_Ports + 1$
    /* Is the utilization lower than $T_{off}$? Is removing a selectable port possible?  */
else if $utilization < T_{off}$ and $Num\_Sel\_Ports > 1$ then
    /* Set the last selectable up-port as non-selectable                     */
    $i := k + Num\_Sel\_Ports - 1$
    $SetAsNonSelectable(p_i)$
    $Num\_Sel\_Ports := Num\_Sel\_Ports - 1$
end
/* Reset the flit counter                                                    */
$FC := 0$
```

obtain the $RunTime$, U^p_{port} (the fraction of $RunTime$ that the ports are turned on) and U_{cpu} (the fraction of $RunTime$ that the CPUs are running). These values are later used in the power model (Section 2.3) for obtaining the normalized values of $RunTime$, E_{net}, and $E_{cluster}$, as seen in the example in Section 2.3.4. These normalized values will be shown in the plots of Section 4.3.

Section 4.1 outlines the switch architecture model employed in the simulations, while Section 4.2 briefly describes the evaluated topologies, the power-saving strategies evaluated, and the VEF traces used to feed the simulations. Finally, Section 4.3 shows and comments on the results of the experiments.

4.1 Switch Model

The modeled architecture is not based on a single specific system, but it is realistic and representative of current state-of-the-art HPC platforms since the design parameters have been chosen based on several commercial networks (Alverson et al. 2012; Chen et al. 2011; Derradji et al. 2015; Fröning et al. 2013; Yokokawa et al. 2011).

The main specifications of the switch architecture are the following: *IQ* (*Input Queued*) switches (Karol and Hluchyj 1998), *virtual cut-through* switching (Anderson et al. 1993), credit-based flow control, and the three-stage allocation algorithm implemented in the IBM Blue Gene L (Adiga et al. 2005), with the only difference that our algorithm employs round-robin arbiters at all the allocator stages.

Data are transmitted in flits of 16 bytes, grouped in 8-flit packets (or 128-byte packets). The switch logic frequency is 625MHz (i.e., the switch clock resolution is 1.6ns). Since the switch

ALGORITHM 3: Torus selectable port management

```
Input: Set of switch ports $\mathcal{P} = \{p_0, p_1, \ldots, p_{Num\_Ports-1}\}$
       Number of aggregated links $Num\_Agg\_Links$
       Set of flit counters $\mathcal{FC} = \{fc_0, fc_1, \ldots, fc_{Num\_Agg\_Links-1}\}$
       Maximum number of flits transmitted per port in a checking cycle $max$
       Threshold for adding selectable ports $T_{on}$
       Threshold for removing selectable ports $T_{off}$
/* Obtain the number of ports per aggregated link                                  */
$Ports\_per\_agg\_link := Num\_Ports/Num\_Agg\_Links$
/* Check all the aggregated links                                                  */
for $agg := 0$ to $Num\_Agg\_Links - 1$ do
    /* Get the number of selectable ports of the current aggregated link           */
    $Num\_Sel\_Ports = getNumberOfSelectablePorts(agg)$
    /* Obtain the aggregated link utilization                                      */
    $utilization := fc_{agg}/(Num\_Sel\_Ports * max)$
    $First\_Port\_in\_agg\_link := Ports\_per\_agg\_link * agg$
    /* Is the utilization greater than $T_{on}$? Is removing a selectable port possible?  */
    if $utilization > T_{on}$ and $Num\_Sel\_Ports < Ports\_per\_agg\_link$ then
        /* Set the first non-selectable port in the aggregated link as selectable  */
        $i := First\_Port\_in\_agg\_link + Num\_Sel\_Ports$
        $SetAsSelectable(p_i)$
        /* Is the utilization lower than $T_{off}$? Is removing a selectable port possible? */
    else if $utilization < T_{off}$ and $Num\_Sel\_Ports > 1$ then
        /* Set the last selectable port in the aggregated link as non-selectable   */
        $i := First\_Port\_in\_agg\_link + Num\_Sel\_Ports - 1$
        $SetAsNonSelectable(p_i)$
    end
    /* Reset the flit counter of the current aggregated link                      */
    $fc_{agg} := 0$
end
```

crossbar can deliver one flit per cycle, each switch port offers a peak bandwidth of 10Gbytes/s. The latency per hop is approximately 50ns. Each input port has an input buffer of 1024 flits, or 16Kbytes, statically split between the four VCs.

Finally, although our switch is not referred to EEE or any specific technology, we have implemented the LPI mechanism (Christensen et al. 2010) and the Power-Down Threshold (Saravanan et al. 2013) for saving energy on the links. We have used the time values specified in IEEE EEE standard (Reviriego et al. 2009) to configure the delays for turning on ($4.16\mu s$) and turning off ($2.88\mu s$) a link, while we have evaluated several time values for configuring the Power-Down Threshold, as we will explain in the next section.

We have used a system with 64 nodes and 8 cores per node (i.e., 512 cores) for evaluating *POWAR*. Since we want to test *POWAR* in torus and fat-tree networks, we have evaluated the following topologies, also shown in Table 4:

— **4 × 4 × 4 3D torus with four ports per aggregated link**: The network comprises 64 switches with 7 aggregated links composed of 4 single links (6 links for building a 3D torus

Table 4. Evaluated Topologies

	Torus	Fat-tree
Topology	$4 \times 4 \times 4$	8-ary 2-tree
Number of switches	64	16
Number of ports	28	16
Link aggregation	4 ports	No
Number of aggregated links	7	–
Routing algorithm	2 deterministic VCs (DOR) 2 fully-adaptive VCs	Fully-adaptive in ascending phase Self-routing in descending phase

and 1 link to the compute node), i.e., 28 ports per switch. The switches use the fully-adaptive routing algorithm described in Section 3.

— **8-ary 2-tree**: The network comprises 16 switches with 16 ports per switch. The switches use fully-adaptive routing in the ascending phase, and self-routing in the descending phase, as described in Section 3.

We have performed experiments varying the value of the Power-Down Threshold. Specifically, we selected the values of 0, 1, 10, and 100μs, and 1ms. Note that we could choose lower values, in the order of ns, but according to Saravanan et al. (2013), those values offer poor performance results. For this reason, we have chosen values in the order of μs.

4.2 Case Studies

Regarding the network load, we have used the VEF trace framework (Andújar et al. 2015; VEF 2017), an open access trace-driven traffic model. VEF traces are generated by capturing the MPI traffic injected by parallel applications. VEF traces preserve message communication dependences and, using the library provided by the VEF framework, the traffic can be replayed in the network simulator. VEF traces model both MPI point-to-point and MPI collective communication primitives, using the collective communication algorithms implemented in OpenMPI (Gabriel et al. 2004).

Specifically, we have performed the evaluation using the VEF traces generated by parallel applications run in the *GALGO* supercomputer (GALGO 2017). For our tests, we have selected the following applications, trying to consider different realistic scenarios:[4] Nanoscale Molecular Dynamics (*NAMD*, formerly Not Another Molecular Dynamics program) (Phillips et al. 2005), a parallel application for simulating large biomolecular systems; *Gromacs* (Pronk et al. 2013), a scientific application to perform molecular dynamics; and *Linpack* and *MPI Random Access* application from High-Performance Computer Challenge (HPCC) benchmark (HPCC n.d.).

In order to analyze the behavior of *POWAR*, each case study has been tested using 5 different configurations:

— **Reference:** LPI is not implemented and, therefore, no power-saving strategies are applied. The network implements the adaptive routing algorithms described in Section 3, performing the selection function with a round-robin arbitration. Note that the results of the remaining configurations are normalized with respect to this network. For this reason, this configuration is not included in the plots, since its normalized runtime or energy values would always be 100%.

In addition, to evaluate the impact of the power-saving mechanisms, we have also measured the "ideal" energy consumption of the reference network. We consider the "ideal"

[4]All the VEF traces described in this work and the software needed to run the VEF traces are freely available at the VEF website (VEF 2017).

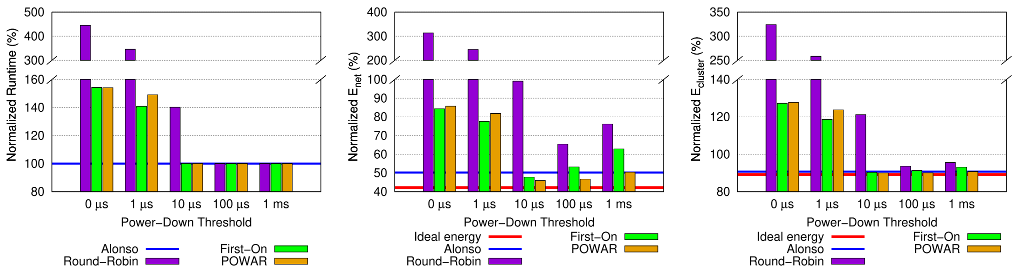
energy consumption as the energy consumed if the network has an ideal mechanism that instantaneously wakes up a port when a packet requires it, and instantaneously powers off the port when the packet transmission finishes. We have added the "ideal" energy consumption to the E_{net} and $E_{cluster}$ plots.

- **Alonso:** Same configuration as reference network but implementing LPI and the power-saving mechanisms by Alonso et al. for fat-tree networks (Alonso et al. 2015) and torus with aggregated links (Alonso et al. 2010). Note that the ports are turned on/off by Alonso's mechanism, not by the Power-Down Threshold, as the remaining configurations. For these reason, **Alonso** appears in the plots as a single line instead of a bar for each Power-Down Threshold value. We set T_{on} to 0.5 and T_{off} to 0.25, for the same reasons explained in *POWAR* configuration below.
- **Round-Robin:** Same configuration as reference network, but implementing LPI with the Power-Down Threshold (i.e., the ports are switched to low-power mode after a certain time without packet transmissions). The selection function is also a round-robin arbiter and it does not take into account the port status or the utilization of the links.
- **First-On:** Implements LPI and Power-Down Threshold, and the round-robin algorithm gives higher priority to wake-up ports. This selection function only chooses a sleep port if there are no available wake-up ports. This selection algorithm is basically the same as Algorithm 1, but the network ports are always selectable.
- ***POWAR:*** Implements LPI, Power-Down Threshold, and the power-saving strategy proposed in Section 3; that is, the round-robin algorithm gives higher priority to the wake-up ports and it also manages which ports can be selectable taking into account the link utilization (Section 3.1 for fat-trees topologies and Section 3.2 for torus topologies). To test *POWAR*, we have selected a set of on/off thresholds that provide both high aggressiveness and moderate responsiveness. In particular, T_{on} is set to 0.5 and T_{off} to 0.25. It must be noted that any pair of thresholds fulfilling the limitations defined in Section 3 is logically correct and provides power reduction. When considering real HPC applications, the eventual impact on energy consumption ranges in a 3% window for different threshold configurations for the applications we tested, always providing benefits with respect to the non-power-aware system. In general, more aggressive threshold values provide greater power consumption reductions and greater performance penalties, while more conservative thresholds generate the opposite effect, thus reducing the performance penalty and the power savings. However, unless we choose very conservative/aggressive threshold values, the observed differences are not very significant, although the best pair of threshold values is different for each application. To sum up, *POWAR* is resilient to the specific values of the configuration parameters (inside the limits fixed in Section 3), provides energy savings with low performance impact, while finding an optimal mechanism for the energy-performance binomial remains an open problem. For these reasons, and for the sake of clarity, we have only shown a pair of threshold values that obtains excellent results (although not the best results) in all the cases.

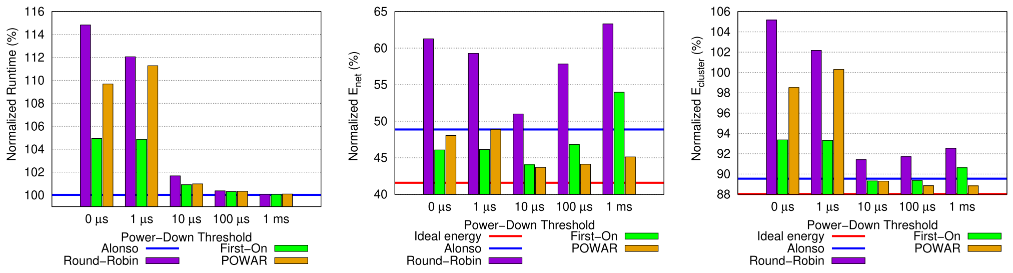
4.3 Results

Fat-tree Topology. Figure 4 shows normalized *RunTime*, normalized E_{net}, and normalized $E_{cluster}$ obtained after running the four applications on the fat-tree. Note that the normalized values have been calculated as explained in Section 2.3.4.

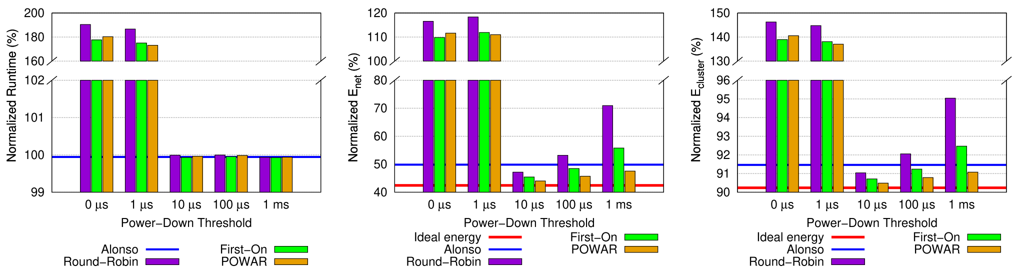
As expected, according to the state of the art, small values for the Power-Down Threshold (0 and 1μs) have negative impact on application performance. For these very low Power-Down Thresholds, in the worst case (Linpack application), the runtime is more than 400% higher than the



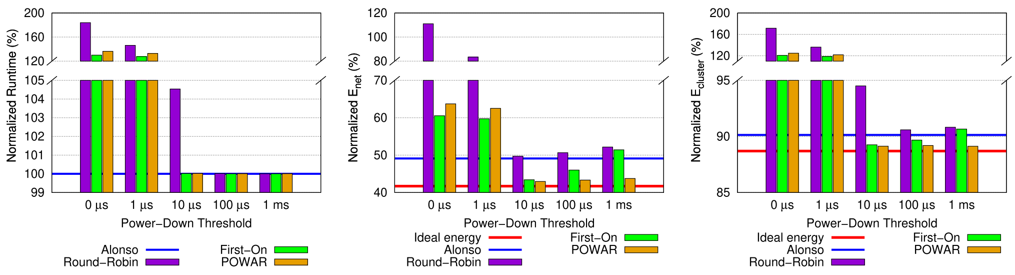
(a) Linpack



(b) Gromacs



(c) MPI Random Access



(d) NAMD

Fig. 4. $RunTime$, E_{net}, and $E_{cluster}$ for fat-tree topology.

runtime of the network without LPI. Applying "First-On" and *POWAR* selection functions reduces performance penalty in all cases, although the performance improvement is not high enough to save energy when the whole system is considered. Gromacs is the only application that gets power saving, while all the other applications increase their energy consumption due to longer runtime.

For the Power-Down Threshold of 10μs, the Round-Robin selection function has moderate performance penalties, except for Linpack, for which runtime increases to 140%. For the remaining thresholds (100μs and 1ms), this selection function has no significant performance penalty. Again, these results agree with the results obtained in the state of the art.

Considering the three higher values for the Power-Down Threshold, *POWAR* also has negligible impact on performance (only 1% in the worst case), but comparing it with the Round-Robin selection function, *POWAR* always reduces the network and system power consumption. Moreover, energy consumption provided by *POWAR* is very close to ideal.

The First-On selection function also has negligible impact on performance and reduces the power consumption with respect to the Round-Robin function. Specially with the Power-Down Threshold of 10μs, the energy savings are very close to the results obtained by *POWAR*. However, the higher the values of the Power-Down Threshold, the lower the energy savings achieved by First-On function.

In contrast, *POWAR* is more resilient to increasing values of the Power-Down Threshold. This is an extra advantage, since the choice of the best Power-Down Threshold depends on the application traffic. As shown in the results, for Linpack, MPI Random Access, and NAMD, the greatest system energy savings are achieved by setting the threshold to 10μs, but the best threshold for Gromacs is 100μs. In this case, although E_{net} is lower for 10μs, the increased application runtime makes the entire system consume more energy than using 100μs as a threshold.

Regarding *Alonso* configuration, it has a negligible impact on performance and obtains good energy savings. Alonso's power-saving strategy always maintains a turned-on subset of ports to ensure that the network is fully-connected. For this reason, the performance degradation is negligible, but the power saving is more limited. Due to this limitation, *POWAR* obtains greater power savings, also obtaining non-significant performance degradation.

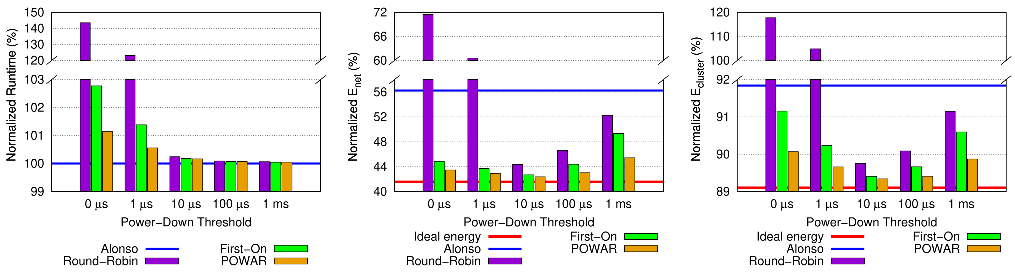
Summarizing, for Power-Down Threshold lower than 10μs, the negative impact on execution time increases the whole system energy consumption for all the analyzed selection functions. For Power-Down Threshold greater than or equal to 10μs, *POWAR* has negligible performance penalties, obtains good energy savings, and it is resilient to the changes of the Power-Down Threshold.

Torus Topology. Figure 5 shows normalized $RunTime$, normalized E_{net}, and normalized $E_{cluster}$ obtained after running the four applications in the 3D torus network. The results are similar to those obtained with the fat-tree, although the performance penalty is lower for lower values of Power-Down thresholds (0 and 1μs). In the torus, the greatest penalty is approximately 45% for MPI Random Access and Linpack applications.

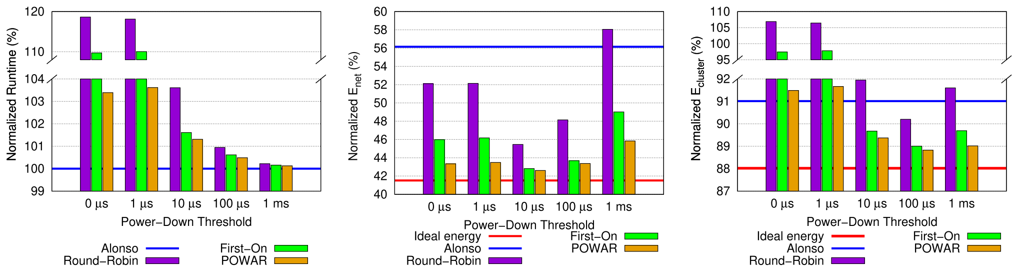
In general, *POWAR* achieves good energy savings with a small performance penalty for the torus network regardless of the Power-Down threshold value. Only for MPI Random Access and the Power-Down Threshold of 0, *POWAR* incurs in a great performance penalty, increasing the execution time up to 120% of the runtime on the reference network. But in the remaining cases, the performance penalty does not exceed 4%, including the values of 0 and 1μs of the Power-Down threshold. For the remaining thresholds, the performance penalty is negligible in most of the cases. The greatest system energy savings are achieved using the Power-Down Threshold of 10μs, except for the Gromacs application, which again achieves the best results with the threshold of 100μs for the same reason as in the fat-tree analysis.

Comparing *POWAR* with the *Alonso* configuration, we can reach the same conclusions as in the fat-tree: although the performance impact of *Alonso*'s mechanism is negligible, it achieves lower power savings than *POWAR* due to *Alonso* always maintaining a turned-on subset of ports.

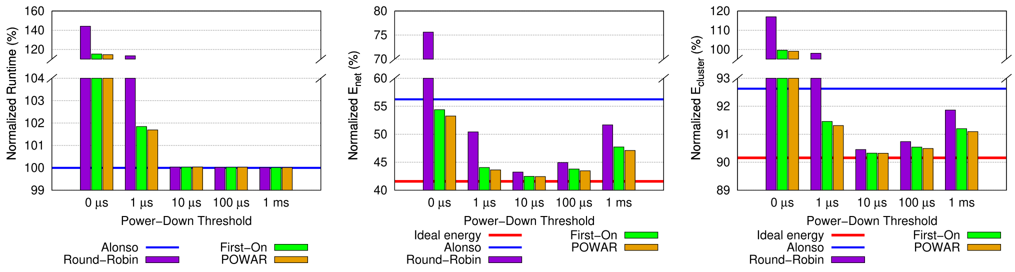
Summarizing, in the torus network, *POWAR* achieves great results, having small or negligible performance penalties independently of the selected Power-Down Threshold, while achieving good energy savings very close to the ideal energy consumption of the system.



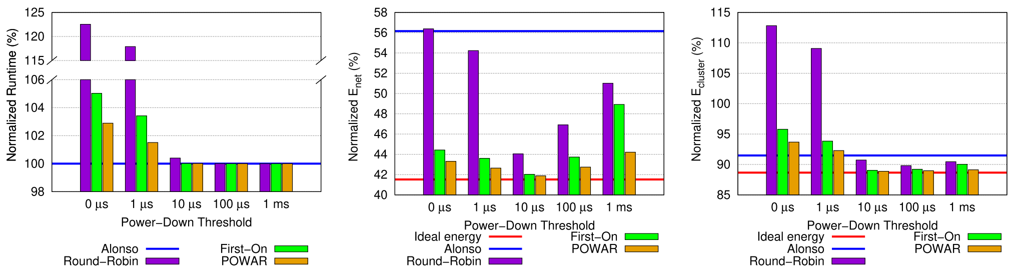
(a) Linpack



(b) Gromacs



(c) MPI Random Access



(d) NAMD

Fig. 5. $RunTime$, E_{net}, and $E_{cluster}$ for torus topology.

5 CONCLUSIONS

In this article, we propose the use of *POWAR*selection function to improve power savings of HPC network whose technology supports on/off links, such as EEE. While the LPI and the Power-Down Threshold mechanism manages when the ports can be switched "on" or "off," *POWAR* determines which network ports can be used for forwarding the packets, as a function of the network utilization, thus being aware of the energy consumption. Evaluation experiments have been performed by trace-driven simulation using typical HPC workloads on the most popular topologies (i.e., fat-tree and torus). Based on our results, we conclude that our proposed *POWAR* selection function maximizes energy savings both at network and at system levels, with little impact on performance, extending in all cases the benefits of using Power-Down Threshold alone. Energy savings in the order of 55%–65% have been obtained at network level, generating savings at system level in the order of 9%–10%, with no runtime penalties in most cases (less than 2% in the worst case). Most important, the energy consumption of the system when applying our proposal is very close to the ideal situation where network links only consume power while sending data. In addition, *POWAR* increases the resilience of the energy saving mechanism to suboptimal values of the Power-Down Threshold.

REFERENCES

Dennis Abts, Michael R. Marty, Philip M. Wells, Peter Klausler, and Hong Liu. 2010. Energy proportional datacenter networks. In *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA'10)*. ACM, New York, NY, 338–347.

N. R. Adiga et al. 2005. Blue Gene/L torus interconnection network. *IBM Journal of Research and Development* 49, 2 (March 2005), 265–276.

M. Alonso, S. Coll, J. M. Martínez, V. Santonja, and P. López. 2015. Power consumption management in fat-tree interconnection networks. *Parallel Comput.* 48, C (Oct. 2015), 59–80.

Marina Alonso, Salvador Coll, Juan-Miguel Martínez, Vicente Santonja, Pedro López, and José Duato. 2010. Power saving in regular interconnection networks. *Parallel Comput.* 36, 12 (2010), 696–712.

Marina Alonso, Salvador Coll, Vicente Santonja, Juan-Miguel Martínez, Pedro López, and José Duato. 2007. Power-aware fat-tree networks using on/off links. In *High Performance Computing and Communications*. Springer Berlin, Germany, 472–483.

Bob Alverson, Edwin Froese, Larry Kaplan, and Duncan Roweth. 2012. Cray XC series network. *Cray Inc., White Paper WP-Aries01-1112* (2012).

T. Anderson, S. Owicki, J. Saxe, and C. Thacker. 1993. High-speed switch scheduling for local-area networks. *ACM Transactions on Computer Systems* 11 (1993), 319–352.

F. J. Andujar, S. Coll, M. Alonso, J. M. Martinez, P. Lopez, F. J. Alfaro, J. L. Sanchez, and R. Martinez. 2018. Analyzing topology parameters for achieving energy-efficient k-ary n-cubes. In *2018 IEEE 4th International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)*. 24–31.

Francisco J. Andújar, Juan A. Villar, Jose L. Sánchez, Francisco J. Alfaro, and Jesús Escudero-Sahuquillo. 2015. 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, 841–848.

L. A. Barroso and U. Hölzle. 2007. The case for energy-proportional computing. *Computer* 40, 12 (Dec. 2007), 33–37.

Jesus Camacho and Jose Flich. 2011. HPC-mesh: A homogeneous parallel concentrated mesh for fault-tolerance and energy savings. In *Proceedings of the 2011 ACM/IEEE Seventh Symposium on Architectures for Networking and Communications Systems (ANCS'11)*. IEEE Computer Society, Washington, D.C., 69–80. DOI: https://doi.org/10.1109/ANCS.2011.17

Dong Chen et al. 2011. The IBM Blue Gene/Q interconnection network and message unit. In *2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–10.

Lizhong Chen and Timothy M. Pinkston. 2012. NoRD: Node-router decoupling for effective power-gating of on-chip routers. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-45)*. IEEE Computer Society, Washington, D.C., 270–281. DOI: https://doi.org/10.1109/MICRO.2012.33

L. Chen, L. Zhao, R. Wang, and T. M. Pinkston. 2014. MP3: Minimizing performance penalty for power-gating of Clos network-on-chip. In *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*. 296–307. DOI: https://doi.org/10.1109/HPCA.2014.6835940

K. Christensen et al. 2010. IEEE 802.3az: The road to Energy Efficient Ethernet. *IEEE Communications Magazine* 48, 11 (November 2010), 50–56.

W. J. Dally and C. L. Seitz. 1987. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers* C-36, 5 (1987), 547–553.

Reetuparna Das, Satish Narayanasamy, Sudhir K. Satpathy, and Ronald G. Dreslinski. 2013. Catnap: Energy proportional multiple network-on-chip. In *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA'13)*. ACM, New York, NY, 320–331. DOI: https://doi.org/10.1145/2485922.2485950

S. Derradji, T. Palfer-Sollier, J. P. Panziera, A. Poudes, and F. W. Atos. 2015. The BXI interconnect architecture. In *2015 IEEE 23rd Annual Symposium on High-Performance Interconnects*. 18–25.

Jack Dongarra, Hans W. Meuer, and Erich Strohmaier. 2018. TOP500 Supercomputer Sites. Retrieved from https://www.top500.org.

José Duato. 1993. A new theory of deadlock–free adaptive routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems* 4, 12 (Dec. 1993), 1320–1331.

José Duato, Sudhakar Yalamanchili, and Lionel Ni. 2003. *Interconnection Networks. An Engineering Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA.

H. Fröning, M. Nüssle, H. Litz, C. Leber, and U. Brüning. 2013. On achieving high message rates. In *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*. 498–505.

Edgar Gabriel et al. 2004. Open MPI: Goals, concept, and design of a next generation MPI implementation. In *Proceedings of the 11th European PVM/MPI Users' Group Meeting*. 97–104.

GALGO 2017. GALGO—Albacete Research Institute of Informatics Supercomputer Center homepage. Retrieved from http://www.i3a.uclm.es/galgo.

Albert Greenberg, James Hamilton, David A. Maltz, and Parveen Patel. 2008. The cost of a cloud: Research problems in data center networks. *SIGCOMM Comput. Commun. Rev.* 39, 1 (Dec. 2008), 68–73.

F. Guo, O. Ormond, M. Collier, and X. Wang. 2012. Power measurement of NetFPGA based router. In *2012 IEEE Online Conference on Green Communications (GreenCom)*. 116–119.

HPCC [n.d.]. HPC Challenge Benchmark. Retrieved from http://icl.cs.utk.edu/hpcc/index.html.

M. Karol and M. Hluchyj. 1998. Queuing in high-performance packet-switching. *IEEE Journal on Selected Areas* 1 (1998), 1587–1597.

G. Kim, H. Choi, and J. Kim. 2018. TCEP: Traffic consolidation for energy-proportional high-radix networks. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. 712–725. DOI:https://doi.org/10.1109/ISCA.2018.00065

M. Koibuchi et al. 2009. An on/off link activation method for low-power ethernet in PC clusters. In *2009 IEEE International Symposium on Parallel Distributed Processing*. 1–11.

J. Li, W. Huang, C. Lefurgy, L. Zhang, W. E. Denzel, R. R. Treumann, and K. Wang. 2011. Power shifting in thrifty interconnection network. In *2011 IEEE 17th International Symposium on High Performance Computer Architecture*. 156–167.

G. Michelogiannakis and J. Shalf. 2014. Variable-width datapath for on-chip network static power reduction. In *2014 8th IEEE/ACM International Symposium on Networks-on-Chip (NoCS)*. 96–103. DOI:https://doi.org/10.1109/NOCS.2014.7008767

James C. Phillips et al. 2005. Scalable molecular dynamics with NAMD. *Journal of Computational Chemistry* 26, 16 (2005), 1781–1802.

Sander Pronk, Szilárd Páll, Roland Schulz, Per Larsson, Pär Bjelkmar, Rossen Apostolov, Michael R. Shirts, Jeremy C. Smith, Peter M. Kasson, David van der Spoel, Berk Hess, and Erik Lindahl. 2013. GROMACS 4.5: A high-throughput and highly parallel open source molecular simulation toolkit. *Bioinformatics* 29, 7 (2013), 845–854.

P. Reviriego, J. A. Hernandez, D. Larrabeiti, and J. A. Maestro. 2009. Performance evaluation of Energy Efficient Ethernet. *IEEE Communications Letters* 13, 9 (Sept. 2009), 697–699.

K. P. Saravanan and P. Carpenter. 2018. PerfBound: Conserving energy with bounded overheads in on/off-based HPC interconnects. *IEEE Trans. Comput.* (2018), 1–1.

Karthikeyan P. Saravanan, Paul Carpenter, and Alex Ramírez. 2013. 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, April 21-23, 2013*. 205–214.

V. Soteriou and Li-Shiuan Peh. 2003. Dynamic power management for power optimization of interconnection networks using on/off links. In *11th Symposium on High Performance Interconnects*. 15–20.

E. Totoni, N. Jain, and L. V. Kale. 2013. Toward runtime power management of exascale networks by on/off control of links. In *2013 IEEE International Symposium on Parallel Distributed Processing, Workshops and PhD Forum*. 915–922.

VEF 2017. VEF traces homepage. Retrieved from http://www.i3a.info/VEFtraces.

M. Yokokawa, F. Shoji, A. Uno, M. Kurokawa, and T. Watanabe. 2011. The K-computer: Japanese next-generation supercomputer development project. In *Proceedings of the International Symposium on Low Power Electronics and Design*. 371–372.

F. Zahn, P. Yebenes, S. Lammel, P. J. Garcia, and H. Fröning. 2016. Analyzing the energy (dis-)proportionality of scalable interconnection networks. In *2nd IEEE International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era*. 25–32.

Received May 2018; revised November 2018; accepted November 2018