Document downloaded from:

http://hdl.handle.net/10251/102265

This paper must be cited as:

The final publication is available at

http://doi.org/10.1002/cpe.4361

Copyright John Wiley & Sons

Additional Information

CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2017; **00**:1–25
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe

Constructing virtual 5-dimensional tori out of lower-dimensional network cards

Francisco J. Andújar[2], Juan A. Villar[1*] José L. Sánchez[1], Francisco J. Alfaro[1], José Duato[2] and Holger Fröning[3]

[1]*Department of Computing Systems, University of Castilla-La Mancha, Albacete, Spain*
Email: {juanan, jsanchez, falfaro}@dsi.uclm.es
[2]*Department of Systems Data Processing and Computers, Polytechnic University of Valencia, Valencia, Spain*
Email: fandujarm@gap.upv.es, jduato@disca.upv.es
[3]*Institute of Computer Engineering, Ruprecht-Karls University of Heidelberg, Mannheim, Germany*
Email: holger.froening@ziti.uni-heidelberg.de

SUMMARY

In the Top500 and Graph500 lists of the last years some of the most powerful systems implement a torus topology to interconnect the millions of computing nodes they include. Some of these torus networks are of five or six dimensions, which implies an additional difficulty as the node degree increases. In previous works we proposed and evaluated the nD Twin (nDT) torus topology to virtually increase the dimensions a torus is able to implement. We showed this new topology reduces the distances between nodes, increasing, therefore, global network performance. In this work, we present how to build a 5DT torus network using a specific commercial 6-port network card (EXTOLL card) to interconnect those nodes. We show, using the same number of cards, the performance of the 5DT torus network we are able to implement using our proposal, is higher than the performance of the 3D torus network for the same number of compute nodes.
Copyright © 2017 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Torus Topology; HPC; Deadlock-Free Routing; Performance Evaluation.

1. INTRODUCTION

Today's supercomputers performance depends in large part on the interconnection network they use to interconnect the millions of computing nodes they are composed. Among other aspects, the chosen network topology plays a major role in the overall system performance. Fat-tree [1] and torus

*Correspondence to: Juan Antonio Villar. E-mail: juanan@dsi.uclm.es

Contract/grant sponsor: This work has been supported by the Spanish MINECO and European Commission (FEDER funds) under the projects TIN2015-66972-C5-1-R and TIN2015-66972-C5-2-R; and by the JCCM under the project PEII-2014-028-P. Francisco J. Andujar is also funded by the Spanish MICINN under a Juan de la Cierva grant FJCI-2015-26080.

Copyright © 2017 John Wiley & Sons, Ltd.

topologies [2] are widely used for implementing indirect and direct networks, respectively. In June 2017, there are four supercomputers using the torus topology in the top ten of the Top500 list [3] and eight in the top ten of the Graph500 list [4] (e.g., K-Computer [5] and several supercomputers of Blue Gene/Q family [6]), while the remaining supercomputers use some kind of indirect network topologies (e.g., fat-tree or Dragonfly).

Regarding the network characteristics of a large supercomputer, besides global network performance, there are other important features to take into account like economical cost, power consumption, reliability or scalability. Although the fat-tree topology provides equal access bandwidth to every node of the system and is appropriate for running parallel applications that generate a lot of communication among the nodes, the torus topology provides a reduced hardware and an excellent scalability, allowing an easier implementation for large systems. Moreover, the torus topology supports several routing algorithms that increase the path diversity so that the fault tolerance and load balance become more feasible. For all these reasons, the torus topology is a common topology used in the greatest HPC systems, according to the Top500 and Graph500 lists.

In torus topologies, the network performance is very dependent on the number of dimensions. When the number of dimensions increases, the average number of hops between any pair of nodes is reduced and therefore, the network latency is also reduced. Then, it is well known that, for a large number of nodes to be interconnected, the higher the number of dimensions, the higher the network performance obtained. However, to build torus networks with more dimensions requires to increase the number of ports in the communication hardware, which is not always easy and for sure increases its complexity. Thus, the number of dimensions of the torus networks is clearly limited by the switch radix.

In previous works, we have proposed a new topology, called n-dimensional twin torus, or just nDT torus [7], which allows us to increase the number of dimensions of the torus. For example, if we have 4-port communication cards to build a torus network, we can build a 2-dimensional (2D) torus* or we can build a 3DT torus [8]. In the 3DT torus, each node in the network comprises two 4-port cards: one port of each card is used to interconnect the cards and the six remaining ports are used to connect the node to its neighbours in the three dimensions of the 3D torus.

In general, an nDT torus can be built using $(n+1)$-port cards: one port from each card interconnects both cards and the $2n$ remaining ports compose the nDT torus node. In [7], we show how the network diameter and average distance decrease when building an nDT torus instead of a $\frac{n+1}{2}D$ torus. As a consequence, the performance of the network is increased without extra economic investment.

In those previous works, we evaluated the nDT torus topology by simulation, assuming a simple card architecture to model the network without taking into account a specific interconnection technology. This Special Issue paper extends the study presented in [9] that used a more accurate model based on a specific commercial communication hardware. Specifically, we have developed a model based on the EXTOLL technology [10, 11]. EXTOLL permits to construct direct networks with a node degree of six, and was designed specifically for the use in high-performance computing. It comes with dedicated support for fine-grained communication, as well as bulk data transfers. It features state-of-the-art techniques like virtual output queuing, virtual channels and link-level

*Note that two ports are required for each dimension in the n-dimensional torus.

Copyright © 2017 John Wiley & Sons, Ltd.

Prepared using cpeauth.cls
DOI: 10.1002/cpe

retransmission. Since EXTOLL cards have six communication ports, the common approach is to build a 3D torus network, but using our topology, we propose in [9] to build a 5DT torus network in order to increase the global network performance using the same number of interconnection cards.

Due to the specific characteristics of EXTOLL cards, we have modified in our previous model the crossbar architecture, including a new flow control mechanism, a new routing algorithm, among other resources that will be outlined in Section 3. All these novelties, and a new and more accurate evaluation process, are the main contributions of this new work. The rest of the paper is organized as follows: In Section 2 we present an overview of the EXTOLL architecture and the nDT torus topology. Section 3 explains the implementation of the EXTOLL model and the implementation of the 5DT torus using the EXTOLL model. In Section 4, we evaluate by simulation the performance of several 3D torus and 5DT torus networks with the same number of nodes. Finally, Section 5 outlines the conclusions and proposes some future work.

2. BACKGROUND

In this section we review the previous work. Section 2.1 shows a brief description of the EXTOLL card and a more detailed description of the EXTOLL switch, on which this paper is focused. The terms card and switch refer to the same element because the card is utilized as switch when necessary. Section 2.2 formally defines the nDT torus and shows a brief explanation about the nDT torus port configuration and the nDT torus routing algorithm.

2.1. The EXTOLL technology

EXTOLL is an interconnection network technology designed for high-performance computing and puts special attention on achieving a very low latency, a high bandwidth, a high sustained message rate and a high availability in large-scale networks (up to 64k nodes). EXTOLL cards have six ports, allowing to build any direct topology with a maximum node degree of six. Among these candidates, a 3D torus is the preferred topology for large networks.

Figure 1 shows the top-level block model of the EXTOLL architecture. Three main logical blocks can be distinguished: the host interface, the network interface controller or NIC and the network switch, drawn in green, blue and red, respectively. The host interface connects EXTOLL to the host system using PCIe interface. Previous versions also supported host interfaces based on HyperTransport.

The second block, the EXTOLL NIC, implements different modules to transform the packets sent through the host interface to network packets and vice versa. There are three main functional units (FUs) integrated into the EXTOLL NIC: The Virtualized Engine for Low Overhead (VELO) [12] unit is dedicated to low latency transfer of small messages, while the Remote Memory Access (RMA) [13] unit employs Direct Memory Access to handle large messages. Additional units include an Address Translation unit (ATU) that is used for a fast translation from virtual address to physical address for the RMA. The Shared Memory Functional Unit (SMFU) [14] allows load/store forwarding between the different address spaces of multiple nodes. Finally, the Control & Status Register unit is used to configure the EXTOLL card, to acquire status information and for debugging purposes.

Copyright © 2017 John Wiley & Sons, Ltd.

Prepared using cpeauth.cls
DOI: 10.1002/cpe

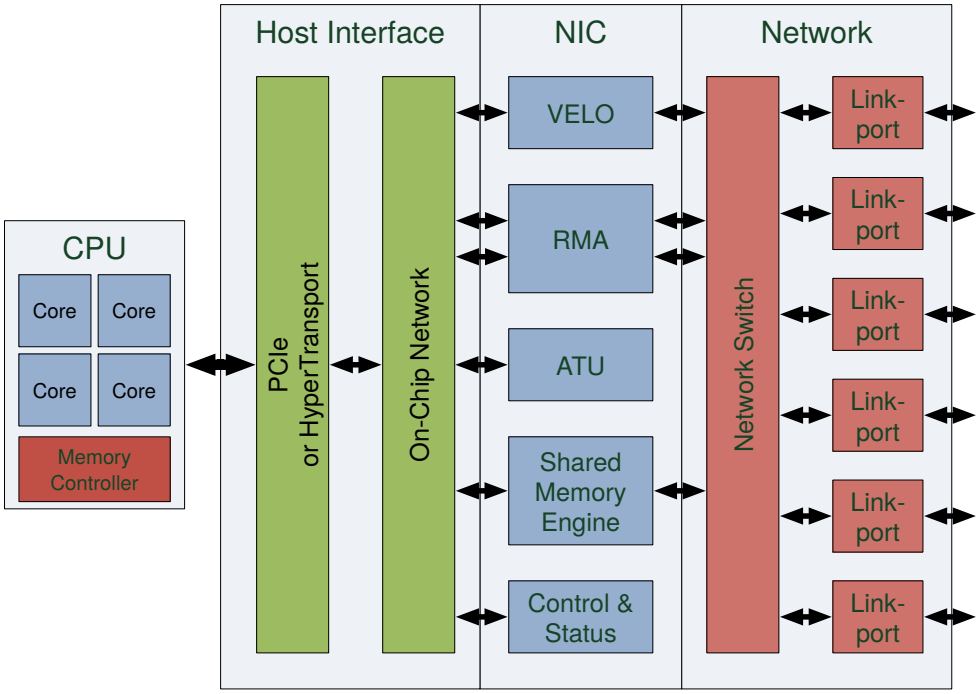


Figure 1. Top-level block diagram of EXTOLL architecture.

The third block implements a crossbar-based switch, which is therefore integrated in each add-in card. The main EXTOLL switch features are described below. A couple of other characteristics are out of scope of this work, like for instance multicast and barrier hardware support.

The EXTOLL switch itself has ten ports, with four of them connecting the switch with the NIC FUs, and the remaining six ports for network-side connectivity. The switch is based on virtual cut-through [15] as switching technique, the iSLIP scheduling algorithm [16] and a credit-based flow control.

The EXTOLL switch is an IQ (Input Queued) switch [17, 18]. A typical problem of IQ switches is head-of-line (HOL) blocking, as, when the first packet in the buffer is blocked, other packets stored in the same buffer but destined to other output ports which could be forwarded, are also blocked. This HOL blocking is circumvented by virtual output queuing (VOQ) [19, 15] at switch level; i.e, each packet is stored in a different queue depending on its output channel, and therefore, the negative effect of the HOL blocking is reduced.

To provide a quality-of-service mechanism, the EXTOLL switch offers four different Traffic Classes (TCs). The software decides on which TC the packets are injected, and the packets cannot change their TC during their transmission. Each TC has two deterministic virtual channels (VCs) to enable the implementation of deadlock-free routing algorithms [20]. Routing functions can differ for each TC to optimize for different traffic characteristics, and of course adaptivity. TC with support for adaptivity includes a third VC to allow the implementation of adaptive routing algorithms [21]. In order to support multiple TCs, VCs and the VOQ-switch technique, the EXTOLL switch is based on multi-queue FIFO buffers [22].

Routing for this switch is based on tables. Each input port has a dedicated routing table, thus allowing that multiple input ports perform the routing simultaneously. It is not necessary to arbitrate

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

among multiple input ports trying to read the same routing table. Moreover, the table-based routing allows to build arbitrary topologies and to recover the network when there are failures regarding add-in-cards, cables or other components. The main drawback of the table-based routing is that large tables are required at each input port. Thus, hierarchical routing tables are used to reduce the size of these tables.

EXTOLL packets have a variable size, with a minimum data granularity being a cell, a data chunk of 128 bits, and packet size varying from 1 cell to 32 cells. The main reason is to perform large data transfers efficiently, and avoid wasting buffer resources for small data transfers.

Fine-grain credit-based flow control is implemented to improve buffer usage. Each packet is logically split in multiple smaller parts (cells), consuming one credit for each part. As a result, a packet consumes one or several credits. If each packet would instead consume a single credit, each credit consumption would have to block the maximum packet size in the receiving buffer. This would be very inefficient since if there are a lot of small packets travelling in the network, a huge amount of buffer space could be reserved unnecessarily and, therefore, be wasted. As buffer space is very scarce, a fine-grain control flow allows for a more efficient use of resources.

2.2. The nDT torus

A couple of definitions related to the nDT torus are included for this paper to be self-contained. More details can be found in [8, 7]. First, we introduce the notation used and formally define the nD twin torus topology [7].

Notation

- n: number of dimensions of the torus, $n \geq 2$.
- d_i: i-th dimension of the nD torus (or the nDT torus), $0 \leq i < n$.
- d_i^+, d_i^-: ports of the dimension d_i.
- PE0, PE1: processing elements of a nDT torus node.

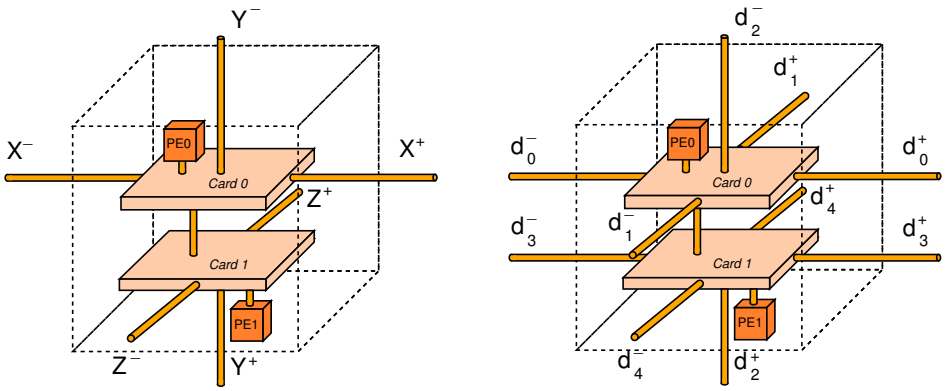
Definition 2.1
An nD Twin torus, or just nDT torus, is an n-cube k-ary (nD torus) topology, with $k \in \mathbb{N}^*$, $k \geq 2$ and $n \geq 3$, where each node is basically composed of the following main components:

- Communication hardware: it consists of two $(n+1)$-port cards, offering a total of $2n+2$ ports. Two of these ports (one in each card) are used to interconnect the cards, and the $2n$ remaining ports are used to connect the node to the other dimensions, building a torus topology with n dimensions.
- Computing hardware: each internal $(n+1)$-port card is connected to a processing element (PE), and so there are two PEs in each node. Therefore, there is a total of $2k^n$ PEs interconnected by the network.

Figure 2 shows two examples of nDT torus nodes, 3DT and 5DT torus nodes, comprise two 4-port cards and two 6-port cards, respectively. Both nodes have two PEs, but from the topological point

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe



(a) 3DT torus node (4-port cards). (b) 5DT torus node (6-port cards).

Figure 2. Examples of nDT torus nodes.

of view, we only need to consider the fact that packets are sourced/destined from/to the PEs and the particular internal layout of PEs is not relevant.

Communication between ports belonging to different cards is established by means the internal link*. In order to reduce the latency introduced by this link, it is important to avoid, as much as possible, the paths passing through it.

This can be achieved with an optimal configuration of the nDT nodes. In [7], we obtained this optimal node configuration (Definition 2.2) under specific conditions. We considered the DOR (Dimension Order Routing) routing algorithm [2]. This deterministic routing algorithm is commonly used in k-ary n-cubes because it is a very simple routing algorithm. Moreover, DOR is deadlock-free using two virtual channels [20] or combining it with the bubble flow control mechanism [23]. Figure 2 shows the optimal node configuration for the 3DT torus and the 5DT torus.

Definition 2.2
Given two communication cards, each one with $(n+1)$ ports, $n \geq 3$, odd†, the port configuration that minimizes the number of paths crossing the internal link in an nDT torus node is defined as follows:

- The ports belonging to dimensions from d_0 to $d_{\frac{n-1}{2}-1}$ are connected to Card0.
- The ports belonging to dimensions from $d_{\frac{n-1}{2}+1}$ to d_{n-1} are connected to Card1.
- The two ports of the dimension $d_{\frac{n-1}{2}}$ are distributed between the two cards. Since k is odd, the number of paths that cross the internal link is the same, regardless of the card in which each port is connected. From hereon in, we assume that the port $d^-_{\frac{n-1}{2}}$ is connected to Card0 and the port $d^+_{\frac{n-1}{2}}$ is connected to Card1.

*Henceforth, we will use "internal link" to refer to the connection between the two cards in an nDT torus node, and "external links" to refer to the remaining ports.

†The nDT torus node for even values of n is also defined in [7], but we have omitted such as definition here because the study presented in this current paper mainly focuses on 5DT tori.

Copyright © 2017 John Wiley & Sons, Ltd.

Prepared using cpeauth.cls
DOI: 10.1002/cpe

A few modifications to the DOR algorithm are required in order to route the messages in the nDT torus. We call this modified version DORT routing algorithm. As DOR, DORT routing algorithm sends a message by the n dimensions following a strictly ascending (or descending) order. Since the nDT torus node comprises two internal cards, once the output port is selected, DORT checks whether the output port is connected to the current card. Depending on the result, the message is routed to the output port or to the internal link. Finally, when the message reaches the destination node, the message is routed to the NIC if the destination is the PE attached to the current card, or it is routed to the internal link if the destination is the other PE in the node.

DORT is not deadlock-free due to the use of the internal link. However, this problem can be solved by adding a few virtual channels to the internal link (0, 1 or at most 2 extra virtual channels, depending on the number of dimensions and the mechanism chosen to avoid deadlocks) [7].

In the case of 5DT torus, we can distinguish three cases where internal link is used, depending on the destination of the message after using the internal link: i) The message destination is the PE connected to the other card in the 5DT torus node*; ii) the message uses the internal link to be injected into the network in a dimension connected to the other card (e.g., a message is routed in Card0 to d_4); iii) the message is travelling through dimension d_2.

As can be deduced, a message can use the internal link regardless of the dimension where it is travelling, and as a consequence, deadlocks could appear. For example, let us consider a message routed in a 5DT torus, such that the message needs to use the five dimensions following an ascending order. First, the message is routed in d_0, after that the message is routed in d_1 and so on. When the message has already been routed in d_1, the message cannot be routed in d_0 again, avoiding the cycles between messages travelling in d_0 and d_1. However, the use of an internal link does not follow any order. The message is routed to the internal link before travelling in d_0, after travelling in d_4, travelling through d_2, etc.

To avoid deadlocks, the internal link requires four VCs instead of the two VCs required by the external links. The first and second VCs are used for sending the messages of the case i) and ii), respectively. The third and the fourth VCs are used for sending the messages of case iii). Since the message is routed in d_2, two VCs are required for case iii), in the same way that the external links require two VCs.

3. MODELLING EXTOLL CARDS

As above-mentioned, the purpose of this work is to address a study of the 5DT torus topology built using the EXTOLL technology. Obviously, the most accurate study would be obtained using a real cluster whose interconnection network is composed of hundreds of EXTOLL cards. However, this option is unfeasible because we do not have access to a cluster of these characteristics. Moreover, if available, a lot of time is needed to reconfigure the hardware to perform the experiments, and most important, the system reconfiguration could interfere with the work of other cluster users. For these reasons, as usual, we have performed this study by simulation.

*Remember that the ports of dimensions d_0, d_1 and the port d_2^- are connected to Card0, while the ports of dimensions d_3, d_4 and the port d_2^+ are connected to Card1, as can be observed in Figure 2b.

Copyright © 2017 John Wiley & Sons, Ltd.

Prepared using cpeauth.cls
DOI: 10.1002/cpe

At the moment, the markets do not offer to customers the option of purchasing EXTOLL switches. Equipment based on such technology is only available in the form of network cards. Nevertheless, these cards can be considered as a solution due to their switching capability similar to those of switches. In fact, when we use the term EXTOLL switch, we are actually referring to the EXTOLL card.

In order to run simulations, we have developed EXTOLLsim, which models the EXTOLL card, that is accurate enough. Since we want to evaluate network performance from the topological point of view, we have focused on the development of a model for simulating the EXTOLL crossbar. The host interface is not modelled and the model of the NIC FUs has been simplified, thereby reducing the development efforts without losing accuracy of simulations. If the host interface would have been included in EXTOLLsim, there would be a latency in the communication introduced by the host interface model. However, this latency would be independent of the network topology and would have no effect for the purposes of this study. Therefore, the resulting effect would be to make simulations more inefficient in terms of execution time and memory consumption.

Regarding the real EXTOLL NIC, there are four different FUs (e.g., VELO, RMA) injecting traffic into the network. The lack of a detailed traffic model for each FU, EXTOLLsim does not require to implement all the FUs that exist in a real EXTOLL NIC, but only one generic FU that is not based on an specific FU of the real EXTOLL NIC. Its unique purpose is to generate, inject and receive traffic into the network using synthetic traffic patterns or trace-based traffic. Therefore, the EXTOLLsim model has only seven ports instead of ten ports: one port is connected to the NIC, and six ports are connected to other cards.

Regarding the real EXTOLL network, EXTOLLsim implements a detailed model of the real EXTOLL crossbar. The EXTOLLsim model comprises four logical units (Figure 3): the routing units, the buffer units, the arbitration unit, and the crossbar unit. Each logical unit supports different features of the EXTOLL crossbar, and specifies its own latency.

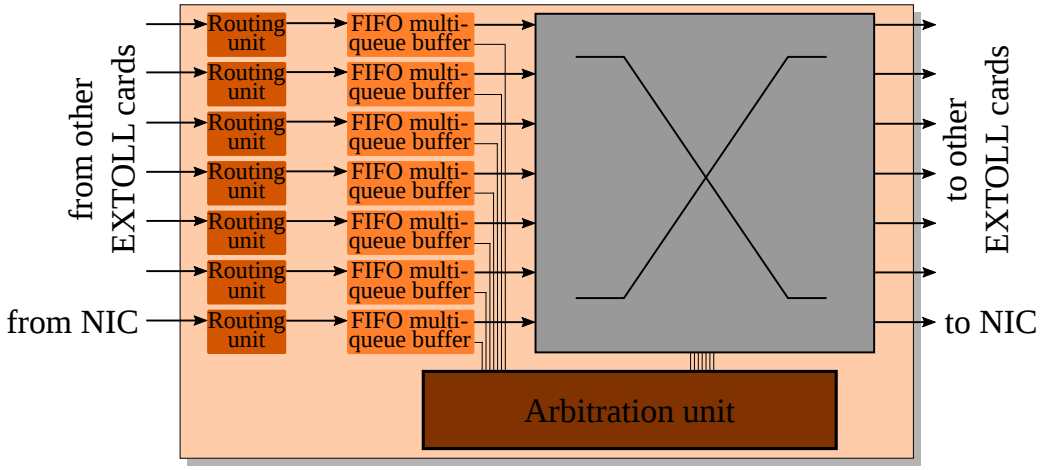


Figure 3. Logical scheme of EXTOLLsim.

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

When a packet arrives at an EXTOLLsim card from a given port, it is processed by the routing unit of that port*. Once the routing unit finishes, such a packet is stored in a buffer unit that implements a multi-queue FIFO buffer with virtual output queuing at switch level, thereby mitigating the negative effect of head-of-line blocking.

The arbitration unit implements the iSLIP scheduling algorithm [16]. Such a unit identifies the packets stored in the buffer units and go through the crossbar unit. Note that the crossbar unit is not only responsible for forwarding the packets to the next card, the proper operation of the virtual cut-through switching, and the fine-grain credit-based flow control. Similar to the real EXTOLL card, EXTOLLsim also implements variable-size packets, four traffic classes (TCs) with multiple VCs, and it supports various network topologies and deterministic/adaptive routing algorithms.

3.1. Deadlock-free routing for 5DT tori using the EXTOLL cards.

Once the development of EXTOLLsim is finally completed, the next step consists in implementing the nD and nDT tori in addition to defining the routing algorithms for both topologies. Since EXTOLL cards have six communication ports, a 3D torus and a 5DT torus can be built.

Regarding the routing algorithms, DOR has been chosen for the 3D torus. As commented above, the DOR algorithm avoids deadlocks using two VCs [20]. The EXTOLL crossbar provides two deterministic VCs per each TC in order to avoid deadlocks. Then, when a packet is travelling inside the dimension d_i, the d_i-coordinate of the current node and the d_i-coordinate of the destination node are compared. If d_i-coordinate of the destination node is greater than the d_i-coordinate of the current node, the packet will be routed to the first VC, called upper-VC. Otherwise, the packet will be routed to the second VC, called lower-VC.

A third VC must be used in the adaptive TCs, thereby allowing the implementation of a fully-adaptive routing algorithm for the 3D torus using the Duato's protocol [21]. A round-robin arbiter may be used to choose among the eligible adaptive channels.

On the other hand, the implementation of the DORT routing algorithm in the 5DT torus is not feasible using the current EXTOLL cards. As Section 2.2 highlighted, DORT requires four VCs in the internal link† to avoid deadlocks [7]. However, EXTOLL cards have insufficient deterministic VCs to implement this algorithm. Fortunately, an EXTOLL card still has the TCs, where each one can use different paths to route packets destined to the same node. As a consequence, we have designed a new routing algorithm for the nDT torus, based on the DOR routing algorithm, that avoids deadlocks using the TCs. We call this algorithm TS-DOR (Twin Source Dimension Order Routing).

Many cycles introduced by the internal link are consumed by the packets injected from PE1. Using DORT, in most of the cases the packets generated by PE1 must go across the internal link to be routed in d_0. However, this internal-link hop can be avoided if the packets are routed first in the dimensions connected to Card1. There exist approaches of routing in the field of networks-on-chip for balancing the network traffic. They use two VCs for routing the packets that follow different dimension orders [24]. For instance, for 2D mesh topologies, in the first virtual network the packets

*Although the EXTOLL switch implements a table-based routing, EXTOLLsim implements the routing function as hardware routing units. The reproduced behaviour is similar, but our implementation saves a big amount of RAM memory during simulation because the routing tables are not kept.

†Note the external links require no modifications.

Copyright © 2017 John Wiley & Sons, Ltd.

Prepared using cpeauth.cls
DOI: 10.1002/cpe

are routed following $X - Y$ order, whereas in the second virtual network the packets are routed following $Y - X$ order.

TS-DOR considers a similar approach to avoid the unnecessary usage of the internal link by the PE1-sourced packets: the PE0-sourced packets are routed from d_0 to d_{n-1}, whereas the PE1-sourced packets are routed from d_{n-1} to d_0. In EXTOLLsim, the TCs can be used to avoid deadlocks between PE0-sourced packets (routed from d_0 to d_4), and PE1-sourced packets (routed from d_4 to d_0). Since four TCs are available, two of them (one adaptive TC and one deterministic TC) are used to inject PE0-sourced packets, and the remaining two are used to inject PE1-sourced packets. Nevertheless, this approach only avoids deadlocks between PE0-sourced packets and PE1-sourced packets. Fortunately, the two deterministic VCs of each TC are enough to avoid deadlocks, considering the same approach used for the external links.

Therefore, when a packet is routed in the dimension $d_{\frac{n-1}{2}}$ and it uses the internal link, TS-DOR takes the same decision that would be chosen in the external links. If the $(d_{\frac{n-1}{2}})$-coordinate of the destination node is greater than the $(d_{\frac{n-1}{2}})$-coordinate of the current node, the message is routed to the upper-VC of the internal link. Otherwise, it is routed to the internal link lower-VC. If a packet is routed to the internal link for another reason, the upper-VC will be chosen if the packet goes from Card0 to Card1. On the contrary, the lower-VC will be chosen if the packet goes from Card1 to Card0.

Finally, when the number of nodes in dimension $d_{\frac{n-1}{2}}$ is even, there are nodes equidistant from the positive and the negative ports of the $(d_{\frac{n-1}{2}})$-dimension. Taking a wrong decision while choosing the port causes the unnecessary usage of the internal link. Let us consider a PE0-sourced packet in a 3DT torus with two nodes in the Y dimension. Commonly, after travelling through dimension Y, the packet travels through dimension Z. Choosing the port Y^+, the packet needs three hops to arrive at Card1 of the other node, whereas choosing the port Y^-, the packet only needs one hop. Figure 4 shows the correct and the wrong decisions in this scenario. Then, to avoid the unnecessary usage of the internal link in case of tie, the port $d^-_{\frac{n-1}{2}}$ is chosen for PE0-sourced packets, whereas the port $d^+_{\frac{n-1}{2}}$ is chosen for PE1-sourced packets.

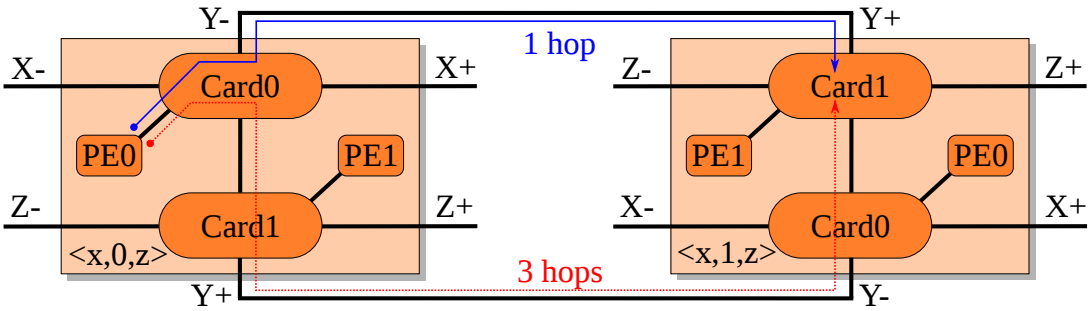


Figure 4. Wrong (red dashed line) and correct (blue solid line) selections of Y link in case of tie routing in the Y dimension.

3.2. TS-DOR: deadlock avoidance discussion

As commented, the *TS-DOR* algorithm checks the destination node $(d_{\frac{n-1}{2}})$-coordinate or the destination card to choose the output VC when the output port is the internal link. From now on,

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

the explanation is focused on the PE0-sourced TCs. From the topological point of view, the PE1-sourced TCs are identical to the PE0-sourced TCs and the following conclusions are the same for the PE1-sourced TCs. Note that in the PE0-sourced TCs, no packets are generated by PE1, but there are messages destined to PE1. Moreover, in order to simplify the explanation, we consider a 3DT torus (Figure 2a), but the conclusions are the same as considering a 5DT torus or an nDT torus.

Let us consider a 4-node Y-ring of a 3D and a 3DT torus, considering only the negative channels. Figure 5 shows the channel dependency graph of both rings. The number of the vertex label shows the Y-coordinate, the second character indicates the link (dimension Y or Internal link) and the third character indicates the VC (Upper-VC or Lower-VC). The filled vertices and the dotted edges represent the Y-dimension channels and their dependencies in a 3D torus, while the remaining vertices and edges represent the channels and dependencies generated by the 3DT torus.

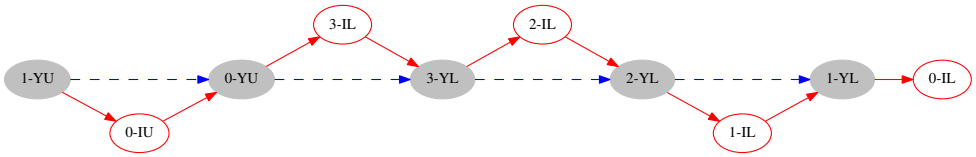


Figure 5. Channel dependency graph considering the negative channels of Y dimension (Y=4).

Let us consider the node 1. This node can receive packets destined to node 0 from the port Y^- of the node 2. In the 3D torus, these packets are routed to the port Y^- and the lower-VC, due to the destination node is lower than the current node, generating a dependency between the lower Y^- channels of nodes 2 and 1. However, in the 3DT torus, these packets are routed to the internal link before using the port Y^-. In both routing operations the destination node is the node 0 and therefore, the lower-VC is chosen. This replaces the dependency $(2-YL) \longrightarrow (1-YL)$ by the dependencies $(2-YL) \longrightarrow (2-IL) \longrightarrow (1-YL)$. In every case, although new dependencies are added, no cycles are generated in the channel dependency graph, thus the ring Y is deadlock-free [20].

However, the internal link is also used for other reasons. For example, in a 3DT torus, a packet can go through the internal link from Card0 to Card1 to travel from an X-dimension port to a Z-dimension port or to arrive at PE1. In this case, it is the first time that the packet uses the internal link and this hop can be seen as if the packet was travelling in the dimension Y. No new dependencies to the channel dependency graph are added in this case.

However, when a packet goes through the internal link from Card1 to Card0, new dependencies are added. In this case, the only reason for using the internal link is that, after travelling through dimension Z, the packet destination is PE0 of the current node. But the internal link is also used for packets travelling through dimension Y which presumably will use the Z-dimension channels in the future. Therefore, new dependencies between Z-dimension channels and Y-dimension channels appear in the channel dependency graph.

Figure 6 shows the channel dependency graph of a 3DT torus with four and two nodes in the dimensions Y and Z, respectively. The X-dimension channels have been omitted because there are no new dependencies that involve the dimension X. Only the Y-negative ring is shown in order to

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

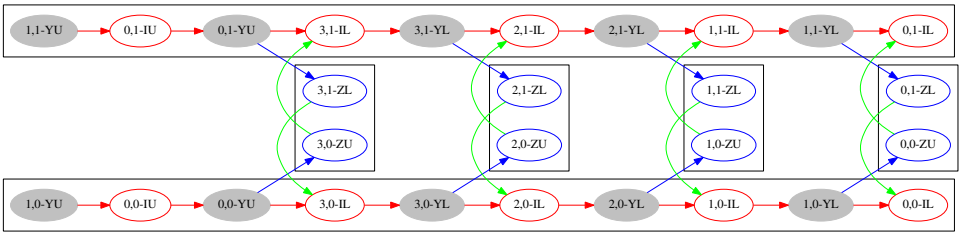


Figure 6. Negative Y dimension and Z dimension channel dependency graph ($Y = 4$ and $Z = 2$).

simplify the graph. The vertices are labelled in the same way that Figure 5, but in this case there are two numbers to indicate the coordinates Y and Z.

As seen, the new dependencies do not generate cycles in the channel dependency graph. Let us consider a packet arrives at Card1 of node $\langle 2, 0\rangle$ through the Z-dimension channel (channel $\langle 2, 1\rangle - ZL$). After that, the packet uses the lower-VC of the internal link (channel $\langle 2, 0\rangle - IL$) to arrive at PE0. This generates an indirect dependency between the Z-dimension channels and the Y-dimension lower channels. But all the packets in the lower channels are destined to a lower Y-coordinate than the current Y-coordinate. Therefore, it is not possible that there are packets using this channel whose destination node Y-coordinate is 3 or 2, generating cycles in the graph. The same happens when the Y-coordinate of the node is 1. Finally, when the node Y-coordinate is 0, a packet can only use the lower channel if the packet is destined to PE0. Then, in this case, the internal link lower channels of nodes $\langle 0, 0\rangle$ and $\langle 0, 1\rangle$ are the graph sinks. Although there are a lot of new dependencies in the graph, no cycles are added, and then, TS-DOR is deadlock free.

4. PERFORMANCE EVALUATION

In this section, we compare by simulation the 3D torus and the 5DT torus performance using the EXTOLLsim model. The main objective of this evaluation is to show that, using the same number of EXTOLL cards to build the 3D torus, we can improve the network performance building a 5DT torus. Note that, unlike previous studies, we do not evaluate the 5D torus. Since EXTOLL cards have 6 ports, building a 5D torus is not possible and performing this evaluation makes no sense.

Specifically, we evaluate 3D and 5DT tori with 256, 512, 1024 and 2048 PEs. Table I shows the topologies for each network size*. Remember that we consider PEs, not cores. Each PE can have several cores, but only one NIC. From this study viewpoint, if the 256-PE network has 256 cores (using single-core nodes) or 4096 cores (using 16-core nodes) is not relevant, since the number of NICs and switches are the same in both cases. Although the evaluated networks are small compared with the networks of the greatest Top500-list supercomputers, these network sizes are reasonable for building supercomputers in research centres with limited economic resources.

*Note that 5DT and 3D networks have the same number of PEs since the 5DT torus nodes comprise two PEs. In addition, since the dimension d_2 in the 5DT torus has two cards per dimension node, the nodes in this dimension are increased in the last place, in order to minimize the network diameter

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

Table I. 3D and 5DT tori evaluated.

Number of PEs	Topology	
	3D torus	5DT torus
256	8×8×4	4×4×2×2×2
512	8×8×8	4×4×2×4×2
1024	16×8×8	4×4×2×4×4
2048	16×16×8	4×4×4×4×4

Regarding the network workload, we consider two different scenarios: a synthetic traffic model (Section 4.1) and a trace-driven model (Section 4.2). First, we describe the different case studies used in the simulations. After that, we provide the simulation results and analyze them.

4.1. Performance evaluation using synthetic traffic patterns

The network performance is evaluated generating the workload synthetically. We consider two traffic patterns for modelling the destination distribution: the uniform traffic pattern, commonly used in network performance evaluations [2], and a destination distribution pattern based on the Zipf's law, which models more realistic traffic [25]. Regarding the packet size, its values are uniformly distributed from the minimum packet size (1 cell) to the maximum packet size (32 cells).

We have performed a set of tests, varying the topology and the traffic pattern in each case. Each test consists of 30 different experiments, obtaining the normalized average throughput and the average cell latency of each test case*. Note that the normalized injection rate and the normalized average throughput are the percentage of the host bandwidth injected and received per each NIC, respectively†.

Figures 7, 8, 9 and 10 show the results obtained for the 256-PE tori, the 512-PE tori, the 1024-PE tori and the 2048-PE tori, respectively. As seen, the 5DT torus obtains a better performance than the corresponding 3D torus. When the network is not saturated, the 5DT torus reduces the average network latency 10% and 15% for the 256-PE and the 512-PE tori, regardless the traffic pattern used, whereas the network latency reduction achieved by the 5DT torus is 30% in the two largest networks.

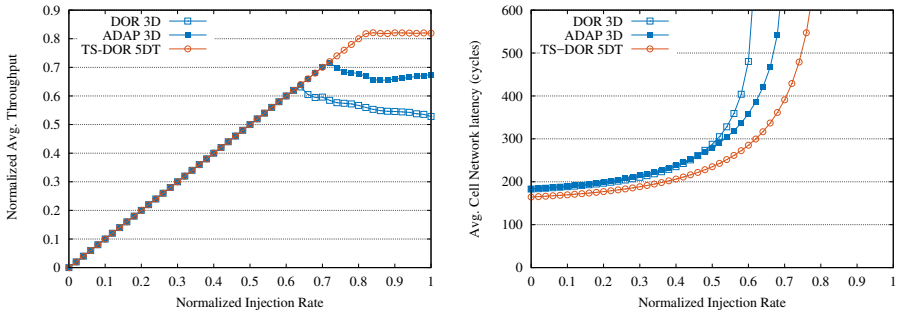
Furthermore, the 5DT torus saturates later, increasing the accepted traffic. For 256-PE and 512-PE tori, the 5DT torus achieves 20% and 12% more throughput than the corresponding 3D torus, using the uniform and the Zipf traffic patterns. But the increment of performance is more dramatic in the largest networks. For example, the 1024-PE 5DT torus achieves 80% of the maximum throughput, whereas the corresponding 3D torus only achieves 35%. This performance degradation is common in large torus networks that employ virtual channels for avoiding deadlock. The virtual channels are a successful scheme to avoid deadlock, but it generates the underutilization of several buffers in the network [26]. In turn, these underutilized buffers generate an unbalanced traffic load. The consequences of the unbalanced traffic load are negligible in small torus networks, but when the

*We have also computed the confidence interval at 95%, but these intervals are imperceptible in the charts at first glance and we have preferred to omit them.

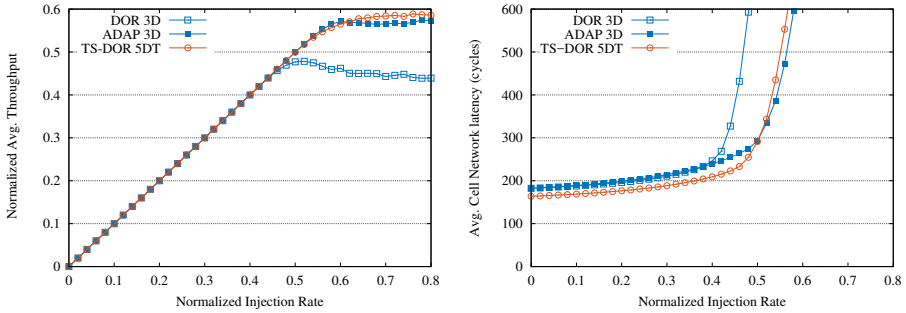
†Since the host bandwidth is 10.4 GBytes/s, each 10% of the normalized injection rate is approximately 1 GByte/s.

Copyright © 2017 John Wiley & Sons, Ltd.

Prepared using cpeauth.cls
DOI: 10.1002/cpe

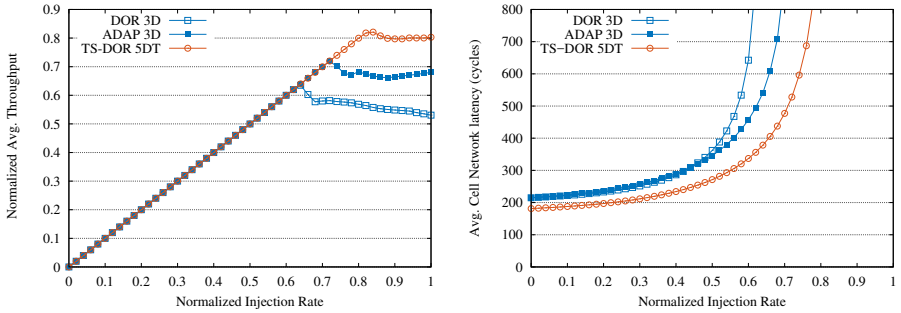


(a) Uniform traffic pattern.

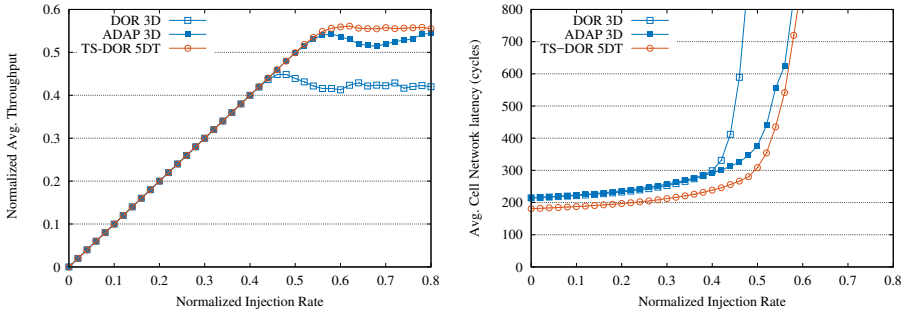


(b) Zipf traffic pattern.

Figure 7. Network performance using synthetic traffic patterns for 256-PE tori.



(a) Uniform traffic pattern.

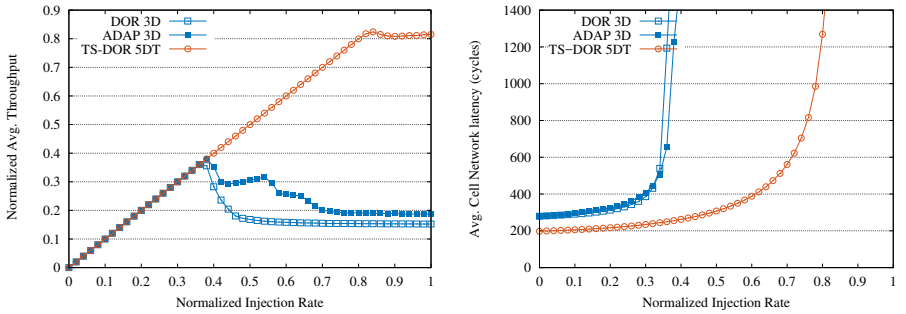


(b) Zipf traffic pattern.

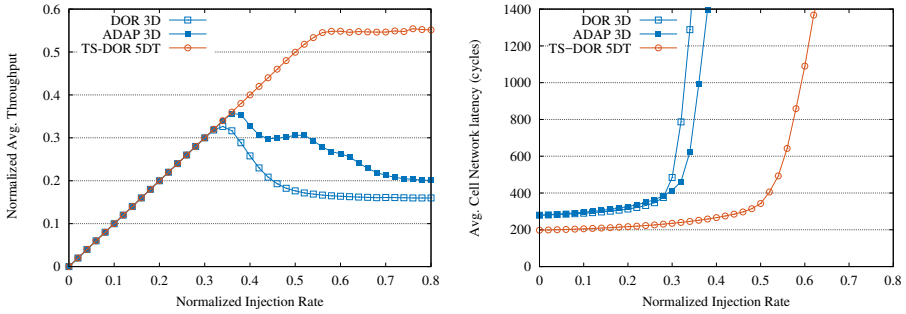
Figure 8. Network performance using synthetic traffic patterns for 512-PE tori.

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

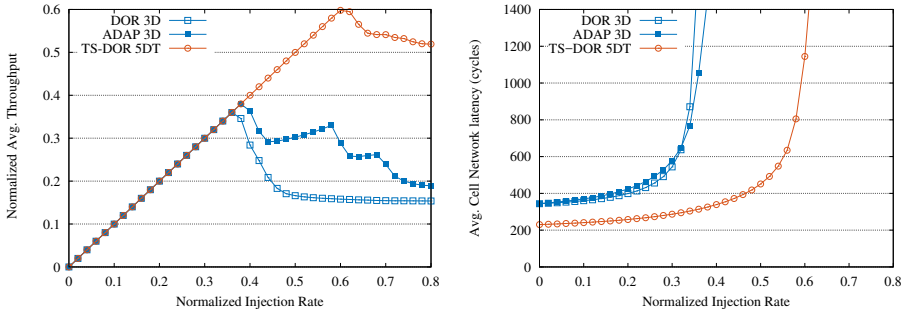


(a) Uniform traffic pattern.

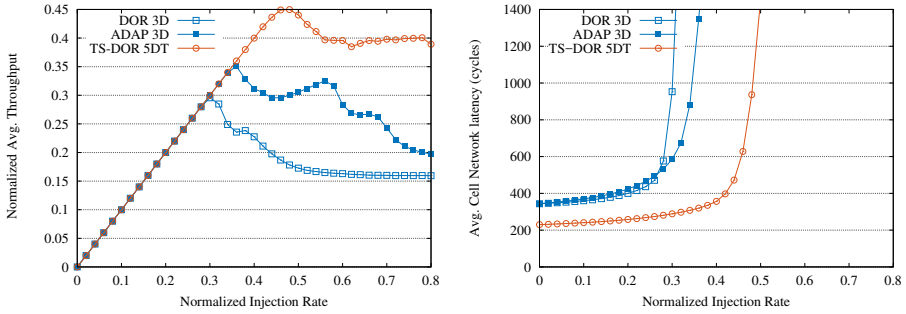


(b) Zipf traffic pattern.

Figure 9. Network performance using synthetic traffic patterns for 1024-PE tori.



(a) Uniform traffic pattern.



(b) Zipf traffic pattern.

Figure 10. Network performance using synthetic traffic patterns for 2048-PE tori.

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

torus network has a large number of nodes per dimension, the accepted traffic decreases significantly after the network saturates [27]. Since the number of nodes per dimension in the 1024-PE 5DT torus is low (a maximum of 4 nodes per dimension against the maximum of 16 nodes in the corresponding 3D torus), the 5DT torus is not affected by this degradation.

Note that the 3D torus performance is improved using an adaptive routing, but the 3D torus only obtains a similar throughput than the 5DT torus under Zipf traffic pattern. In any case, the 3D torus network latency is still higher when the network is not saturated.

4.2. Performance evaluation using the VEF trace-driven traffic model

In this section, we use an open access trace-driven traffic model, called VEF* trace model [28, 29], to evaluate the 5DT torus performance. Using modified MPI libraries, we capture the MPI traffic injected by parallel applications in a trace file, which will be used later for generating the network workload. The VEF traces model both MPI point-to-point and MPI collective communications. The latter are modelled using the collective communication algorithms implemented in OpenMPI [30]. Thus, using the traffic generated by a real application we can evaluate the networks using a more realistic environment and so obtain more significant results.

We consider two different scenarios: Section 4.2.1 shows a performance evaluation of several collective communications, while Section 4.2.2 shows a performance evaluation using VEF traces obtained from MPI applications†.

4.2.1. Performance evaluation of single collective communications. In order to emulate the traffic of a single MPI collective communication, we have created synthetic traces of *Broadcast*, *Reduce*, *AllGather* and *All2All* operations. The first two operations have been chosen for their simplicity, whereas the last two have been chosen because they generate more traffic than any other collective communication. We compare the performance of these operations in 3D and 5DT tori with 256, 512 and 1024 PEs. We also evaluate the network performance using different message sizes, from the minimum and the maximum packet size: 16 bytes (1-cell packet) and 512 bytes (32-cell packet).

Tables II and III show the average cell network latency and the execution time‡ obtained for the evaluated topologies and multicast operations. The reduction of both metrics obtained for the 5DT are also shown. Excluding the Broadcast communication in the smaller networks, all the collective communications are performed faster in the 5DT torus. Note that *AllGather* and *All2All* operations have a high execution time and network latency. For both operations the network is saturated. Even in this circumstance, the 5DT torus increases the performance of these collective communications, obtaining greater performance improvements than the collective communications that do not saturate the network.

4.2.2. Performance evaluation using MPI application traces. Finally, we perform an evaluation using the VEF traces generated by parallel applications run in the supercomputer GALGO [31].

*VEF is the acronym of the nick name of the original programmers: Villar, Escudero and Fran.

†All the VEF traces described in this work and the software needed to run the VEF traces are available free at the VEF website [28].

‡We consider as the execution time the simulation cycle when all the messages in the trace have been received at their destination and there is no message in the network.

Copyright © 2017 John Wiley & Sons, Ltd.

Prepared using cpeauth.cls
DOI: 10.1002/cpe

Table II. Average cell network latency for different multicast operations for 3D and 5DT tori.

Multicast operation	Message Size (bytes)	Average cell network latency (cycles)						Latency Reduction (%)		
		3D torus			5DT torus					
		256 PEs	512 PEs	1K PEs	256 PEs	512 PEs	1K PEs	256 PEs	512 PEs	1K PEs
Broadcast	16	89.718	121.912	123.206	92.227	99.125	110.573	-2.797	18.691	10.254
	512	90.322	125.184	127.392	91.953	101.528	115.877	-1.806	18.897	9.039
Reduce	16	76.710	76.855	89.440	60.271	60.511	60.599	21.430	21.266	32.246
	512	75.216	75.358	87.941	58.776	59.014	59.101	21.857	21.688	32.795
AllGather	16	105.614	171.022	252.850	110.047	149.148	260.962	-4.197	12.790	-3.208
	512	1189.600	2004.073	2123.151	566.583	965.176	1517.570	52.372	51.839	28.523
All2All	16	666.683	1276.833	2937.544	535.052	704.989	1238.380	19.744	44.786	57.843
	512	2743.504	4483.436	5546.037	1420.453	2516.929	3206.925	48.225	43.862	42.176

Table III. Execution time for different multicast operations for 3D and 5DT tori.

Multicast operation	Message Size (bytes)	Execution time (cycles)						Execution Time Reduction (%)		
		3D torus			5DT torus					
		256 PEs	512 PEs	1K PEs	256 PEs	512 PEs	1K PEs	256 PEs	512 PEs	1K PEs
Broadcast	16	754	914	1194	850	938	1026	-12.732	-2.626	14.070
	512	989	1189	1525	1025	1185	1321	-3.640	0.336	13.377
Reduce	16	734	886	1166	670	822	974	8.719	7.223	16.467
	512	961	1141	1449	897	1077	1257	6.660	5.609	13.251
AllGather	16	2621	5261	11761	2505	4917	9497	4.426	6.539	19.250
	512	160033	393377	1037137	120517	294009	697957	24.692	25.260	32.703
All2All	16	2126	5834	26318	1778	3334	9522	16.369	42.852	63.819
	512	25829	73517	231909	11285	33309	97845	56.309	54.692	57.809

We generate the traces using the *MPI Random Access* application of the *HPCC* benchmark [32] and the Graph500* benchmark [4], which are commonly used to evaluate the performance of HPC interconnection networks. Moreover, we generate traces of NAMD [34], a parallel application for simulating large biomolecular systems, using the available input benchmarks (specifically, the *STMV* benchmark) in the NAMD website [35].

We generate traces of these three applications with 256, 512 and 1024 tasks. In order to simulate a more realistic environment, we simulate multicore-PEs whenever possible. For example, the traces with 256, 512 and 1024 tasks are used for testing the 256-PE torus, simulating single-core PEs, two-core PEs and 4-core PEs, respectively. Unfortunately, the trace generation is limited by the size of supercomputer GALGO (up to 1168 cores), and therefore we can only simulate nodes with a small number of cores.

Traffic load characterization of the applications

Before showing the performance evaluation results, and in order to help to understand the obtained results, we discuss briefly the traffic characterization of the three applications. In order to facilitate the image viewing, Figure 11 only shows the message distribution of the applications executed with 128 tasks, but the main characteristics of the message distributions are the same for higher number of tasks.

*Specifically, the Graph500 benchmark is executed using the *replicated-csr* implementation[33].

Copyright © 2017 John Wiley & Sons, Ltd.

Prepared using cpeauth.cls
DOI: 10.1002/cpe

- *Graph500 benchmark*: All the communications are generated by MPI collective communications. *AllReduce()*[*] and *AllGather()*[†] functions are the most frequently performed. These communications generate the diagonal lines appreciated in Figure 11a. For example, when *AllReduce()* performs the *broadcast* communication, the task 0 sends the broadcast message to the tasks 1, 2, 4, 8, 16, 32 and 64. When the tasks receive the broadcast message, they retransmit the message: the task 1 sends the message to the tasks 3, 5, 9, 17...; the task 2 send the message to the tasks 6, 10, 18...; and this process is repeated until all the tasks have received the broadcast message, generating the observed diagonal lines. Moreover, all the tasks send a big amount of messages to task 0 due to the performing of *gather* communication during the *AllGather()*.

- *HPCC MPI Random Access*: Most of the communications are performed by MPI point-to-point communications. The messages are distributed uniformly among all the tasks, i.e., the message distribution of the *HPCC MPI Random Access* application is very close to an uniform traffic pattern.

- *NAMD STMV benchmark*: Most of the communications are performed by MPI point-to-point communications. The application maps logically the tasks in a 3D grid and the tasks communicate mainly with the neighbour tasks in the grid. For this reason most of the sent messages by each source task are concentrated in a few destination tasks, meanwhile there are many tasks that do not communicate during all the execution.

Performance results

Figures 12, 13 and 14 show the results obtained using the Graph500, the HPCC MPI Random Access and the NAMD traces, respectively. We present the average results of the network cell latency and the end-to-end cell latency[‡]. In this case, the execution time is not included. The time spent by the applications using the network is a small part of the total time. Then, the improvement obtained by the network is not significant respect to the total execution time, making this measure an useless metric. Instead of the execution time, we show the average router utilization. This measure indicates the average number of cycles in which the EXTOLL cards have been delivering messages, i.e., it shows how much time the resources are used to perform the same workload. Therefore, a lower value of the switch utilization indicates a more efficient use of the resources, since the network requires a lower time for performing the same work.

As seen, the 5DT torus reduces the latency and the router utilization. The most significant improvements are obtained using the Graph500 benchmark, where all the measures are reduced from 20% to 50%, depending on the case. This application is designed to optimize the use of the network. For this reason, in several parts of its execution the network is not saturated, and

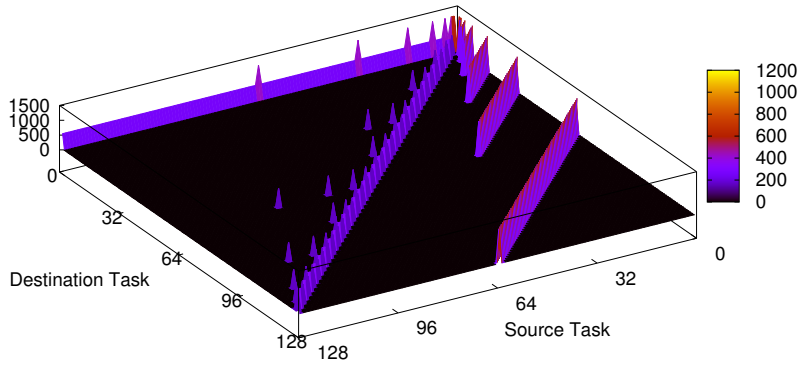
[*]Note that performing *AllReduce()* is equivalent to perform a *reduce* communication followed by a *broadcast* communication; i.e., multiple values on all tasks are reduced to a single value (e.g, obtaining the maximum or the minimum value of these multiple data), after that the reduced value is sent to all the tasks.

[†]The *AllGather()* is equivalent to perform a *gather* communication followed by a *broadcast* communication; i.e., the values of multiple tasks are gathered together in a single task; after that the gathered data is sent to the remaining tasks.

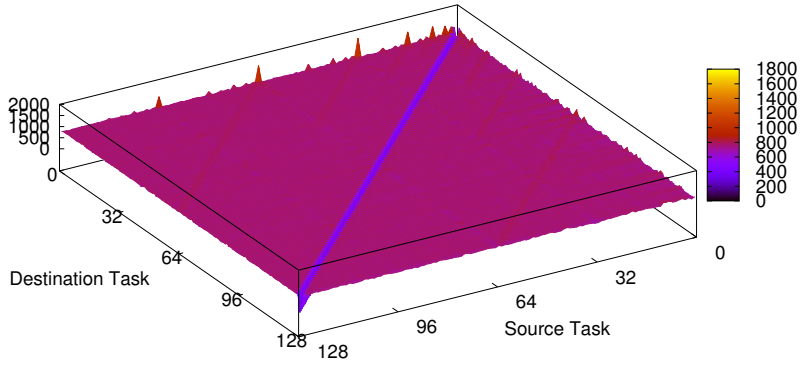
[‡]The network cell latency is the number of cycles spent by one cell travelling trough several EXTOLL crossbars. The end-to-end cell latency is the network latency plus the number of cycles from the moment the cell is generated in the EXTOLL NIC until the cell arrives at the first EXTOLL crossbar associated to this NIC.

Copyright © 2017 John Wiley & Sons, Ltd.

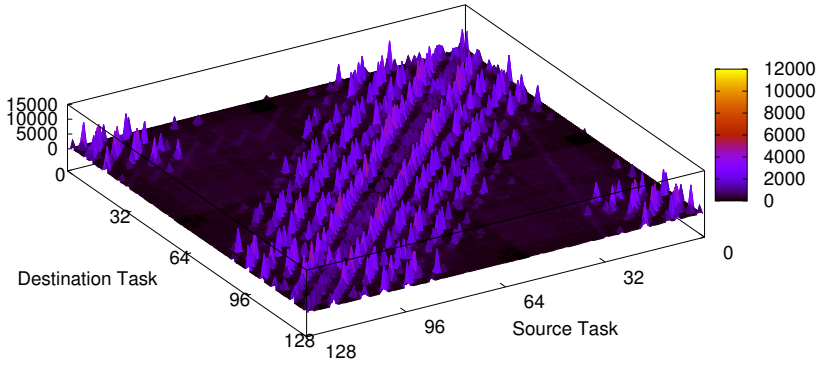
Prepared using cpeauth.cls
DOI: 10.1002/cpe



(a) Graph500 Benchmark.



(b) HPCC MPI Random Access.



(c) NAMD: STMV Benchmark.

Figure 11. Number of messages per source/destination running the evaluated applications with 128 tasks.

because of several messages are sent to distant tasks, the 5DT torus obtains a great improvement on performance with respect to the 3D torus. The adaptive routing also increases the 3D torus performance for the same reason: since the network is not saturated, to find alternative paths is easy, avoiding the possible conflicts between messages in the deterministic 3D torus. Nevertheless, the deterministic 5DT torus obtains a greater performance than the adaptive 3D torus.

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

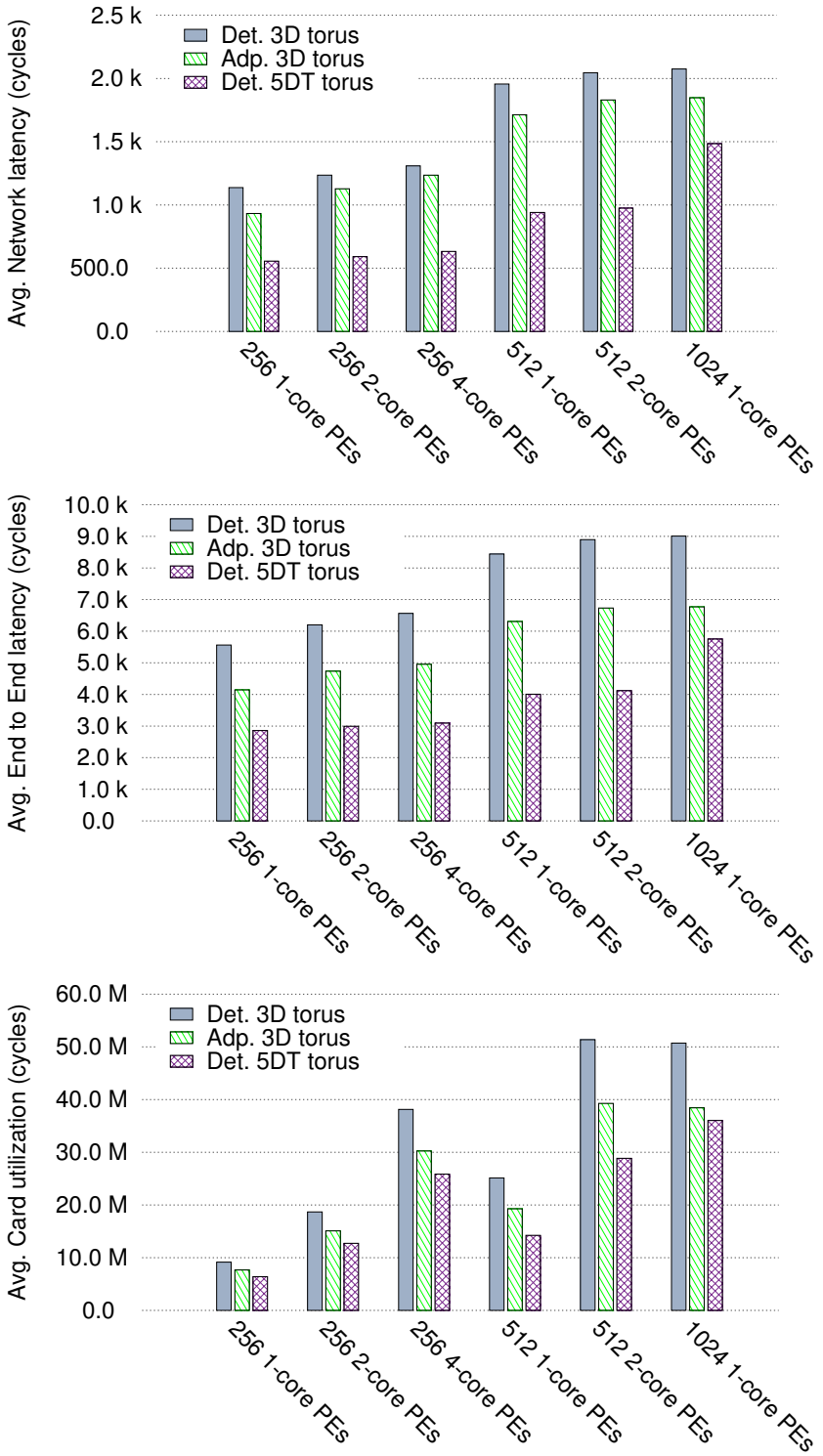


Figure 12. Results of Graph500 benchmark.

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

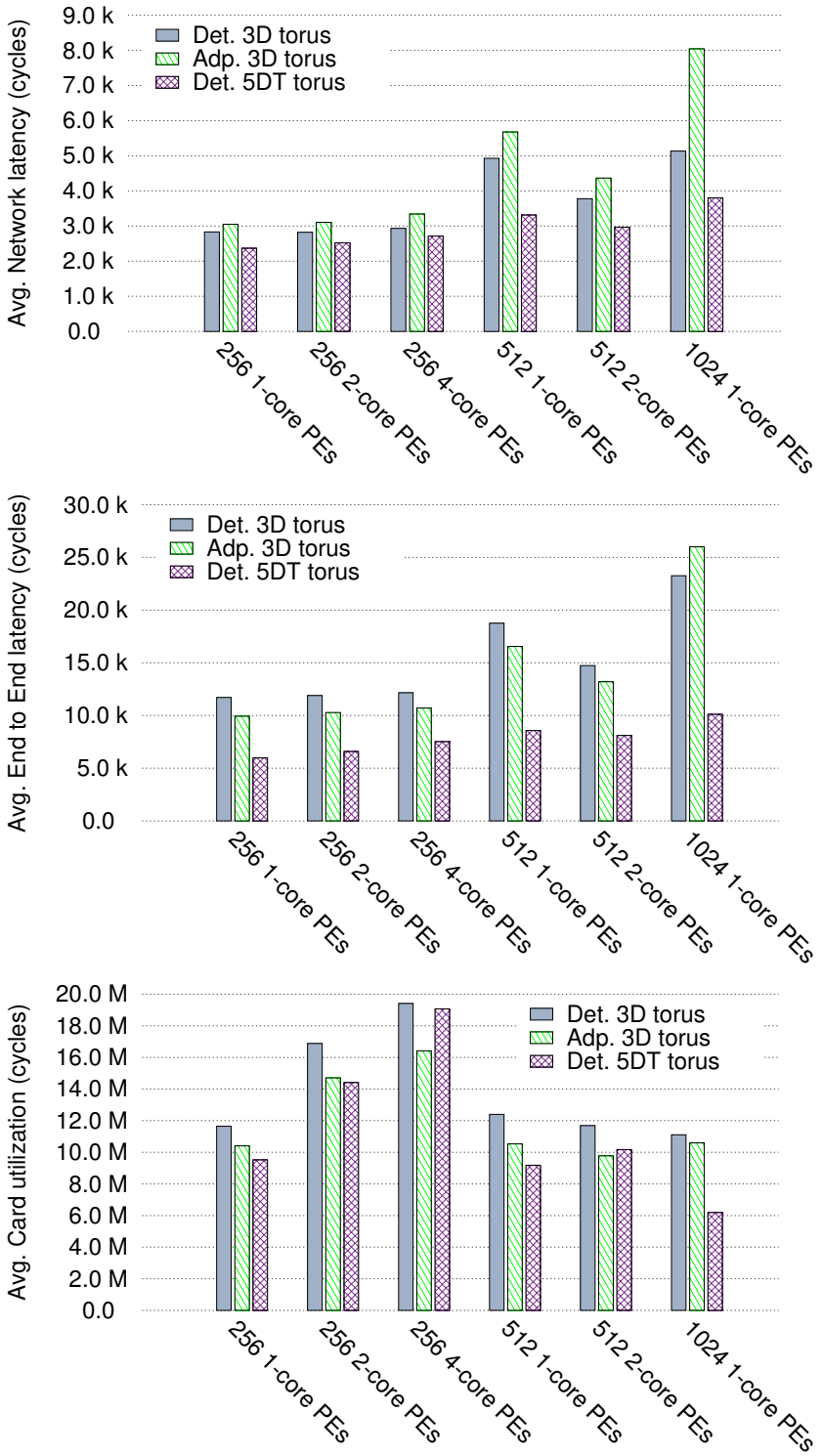


Figure 13. Results of HPCC Random Access benchmark.

Copyright © 2017 John Wiley & Sons, Ltd.

Prepared using cpeauth.cls
DOI: 10.1002/cpe

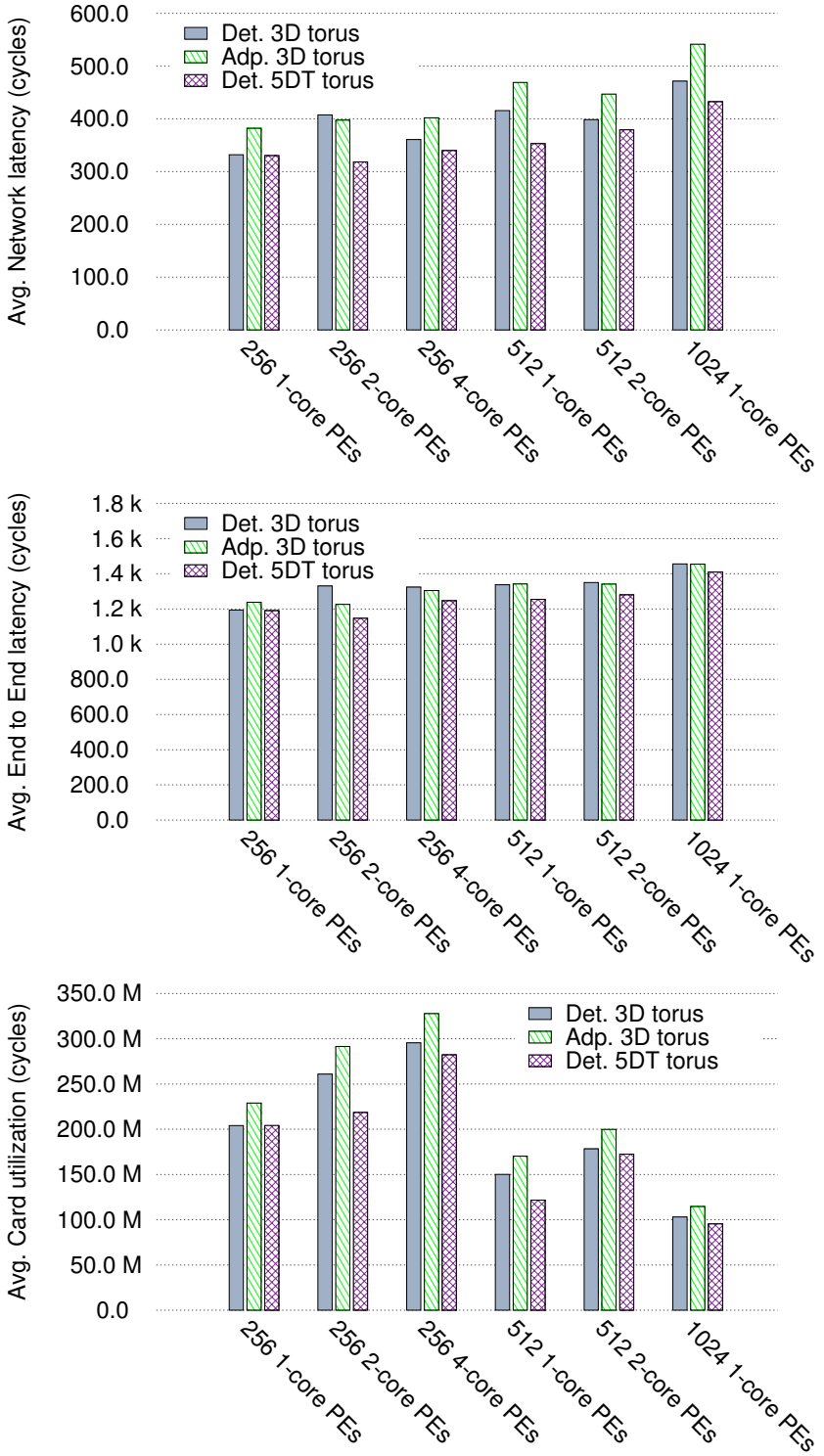


Figure 14. Results of NAMD: STMV benchmark.

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

The 5DT torus obtains similar performance improvement with respect to the 3D torus using the HPCC MPI Random Access traces. Regarding the adaptive 3D torus, remember that the message distribution of this application is very close to an uniform traffic pattern. For this reason, the network throughput is increased (both average end-to-end latency and router utilization metrics are reduced), although the network latency is increased, as usually occurs when adaptive routing algorithms are evaluated under uniform traffic patterns. However, the deterministic 5DT torus improves again the performance of the adaptive 3D torus.

Using the NAMD traces, the performance is also improved, although the differences are less significant than using the other application traces. The 5DT torus improves the network performance because it reduces the distances between the PEs. However, the messages generated by the NAMD application present a high spatial locality; i.e., the PEs usually communicate with the closest PEs. Therefore, the average distance reduction obtained for the messages generated by the NAMD application is lower than the average distance reduction obtained for the messages generated by the Graph500 and MPI Random Access applications. For this reason, although the performance of the NAMD application is improved, the differences obtained are less significant than using the other applications.

5. CONCLUSIONS

Torus topology is a well-known option in today's supercomputers. In order to increase the number of dimensions the topology is able to implement, we proposed in previous works how to build an nDT torus combining two cards of $(n + 1)$ ports and showed how the network performance is increased building the nDT torus instead of the mD ($m = \frac{n+1}{2}$) torus. Building an nDT torus the distances between network nodes are reduced and therefore the performance is increased. However, the nDT torus sets out various technical problems that must be resolved in advance.

In this paper, we apply those previous works to implement an nDT torus using commercial low-profile network cards. These EXTOLL cards have six ports allowing to build a 3D torus and also support arbitrary topologies, allowing to build an nDT torus topology, or more specifically, a 5DT torus topology. In this paper the implementation of the EXTOLL model is discussed with special interest in explaining the implementation of DORT, a routing algorithm for nDT tori. Moreover, we describe the TS-DOR algorithm, which is a custom version of DORT for EXTOLL cards, and discuss how TS-DOR is able to avoid the appearance of deadlock due to the internal link.

For the nDT torus we have proposed the DORT routing algorithm to minimize the use of the internal link. Unfortunately, DORT cannot be applied to nDT tori since EXTOLL cards do not have enough deterministic virtual channels. For this reason, we develop a new deterministic deadlock-free routing algorithm called TS-DOR (Twin Source DOR) that combines the use of EXTOLL VCs and EXTOLL TCs to ensure deadlock-freedom in a 5DT torus.

Finally, in this paper we compare the performance of the 3D torus and the 5DT torus using the EXTOLL simulation model. Both tori are evaluated under several traffic loads, obtained from uniform and Zipf synthetic traffic patterns and real-application traffic traces of the Graph 500 benchmark, the HPCC MPI Random Access application and the NAMD application. Moreover, a characterization of the three application traces is also provided to help understanding the results.

Copyright © 2017 John Wiley & Sons, Ltd.

Prepared using cpeauth.cls
DOI: 10.1002/cpe

As expected, the 5DT torus increases the network performance without changing the switch architecture, only modifying the topology and the routing algorithm. For instance, such throughput improvements can achieve values of up to 45% and 50% under synthetic and trace-based traffic patterns, respectively.

REFERENCES

1. Leiserson CE. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE Transactions on Computers* 1985; **34**(10):892–901.
2. Duato J, Yalamanchili S, Ni L. *Interconnection networks. An engineering approach*. Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2003.
3. Dongarra J, Meuer HW, Strohmaier E. TOP500 Supercomputer Sites June 2017. www.top500.org.
4. The Graph500 list June 2017. http://www.graph500.org/.
5. Yokokawa M, Shoji F, Uno A, Kurokawa M, Watanabe T. The K-Computer: Japanese next-generation supercomputer development project. *2011 International Symposium on Low Power Electronics and Design (ISLPED)*, 2011; 371–372.
6. Chen D, *et al.*. The IBM Blue Gene/Q interconnection network and message unit. *2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011; 1–10.
7. Andújar FJ, Villar JA, Sánchez JL, Alfaro FJ, Duato J. N-dimensional twin torus topology. *IEEE Transactions on Computers* Oct 2015; **64**(10):2847–2861.
8. Andújar FJ, Villar JA, Alfaro FJ, Sánchez JL, Duato J. Building 3D torus using low-profile expansion cards. *IEEE Transactions on Computers* Nov 2014; **63**(11):2701–2715, doi:10.1109/TC.2013.155.
9. Andújar FJ, Villar JA, Sánchez JL, Alfaro FJ, Duato J, Fröning H. A case study on implementing virtual 5d torus networks using network components of lower dimensionality. *2017 IEEE 3rd International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)*, 2017; 9–16, doi:10.1109/HiPINEB.2017.7.
10. Fröning H, Nüssle M, Litz H, Leber C, Brüning U. On achieving high message rates. *13th IEEE/ACM International Symposium on Cluster Cloud and Grid Computing (CCGrid)*, 2013; 498–505.
11. EXTOLL homepage. http://www.extoll.de.
12. Litz H, Fröning H, Nüssle M, Brüning U. VELO: A novel communication engine for ultra-low latency message transfers. *37th International Conference on Parallel Processing (ICPP-08)*, 2008; 238–245, doi:10.1109/ICPP.2008.85.
13. Nüssle M, Scherer M, Brüning U. A resource optimized remote-memory-access architecture for low-latency communication. *38th International Conference on Parallel Processing (ICPP-09)*, 2009; 220–227, doi:10.1109/ICPP.2009.62.
14. Fröning H, Litz H. Efficient hardware support for the partitioned global address space. *10th Workshop on Communication Architecture for Clusters (CAC2010), co-located with 24th International Parallel and Distributed Processing Symposium (IPDPS 2010)*, 2010; 1–6, doi:10.1109/IPDPSW.2010.5470851.
15. Anderson T, Owicki S, Saxe J, Thacker C. High-speed switch scheduling for local-area networks. *ACM Transactions on Computer Systems* 1993; **11**:319–352.
16. McKeown N. The iSLIP scheduling algorithm for input-queued switches. *IEEE/ACM Transactions on Networking* April 1999; **7**(2):188–201, doi:10.1109/90.769767.
17. Karol M, Hluchyj M. Queuing in high-performance packet-switching. *IEEE Journal on Selected Areas* 1998; **1**:1587–1597.
18. McKeown N, Izzard M, Mekkittikul A, Ellersick W, Horowitz M. The Tiny Tera: A packet switch core. *IEEE Micro* 1997; **17**:27–33.
19. Tamir Y, Frazier G. High–performance multi–queue buffers for VLSI communications switches. *SIGARCH Comput. Archit. News* 1988; **16**(2):343–354.
20. Dally W, Seitz C. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers* 1987; **C-36**(5):547–553, doi:10.1109/TC.1987.1676939.
21. Duato J. A new theory of deadlock–free adaptive routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems* dec 1993; **4**(12):1320–1331.
22. Tamir Y, Frazier GL. Dynamically-allocated multi-queue buffers for vlsi communication switches. *IEEE Trans. Comput.* Jun 1992; **41**(6):725–737.

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

23. Carrion C, Beivide R, Gregorio J, Vallejo F. A flow control mechanism to avoid message deadlock in k-ary n-cube networks. *Proceedings of The Fourth International Conference on High-Performance Computing*, 1997; 322–329, doi:10.1109/HIPC.1997.634510.
24. Atagoziyev M. *Networks on Chip: Topology, Switching, Routing*. VDM Verlag: Saarbrücken, Germany, 2009.
25. Elhanany I, Chiou D, Tabatabaee V, Noro R, Poursepanj A. The network processing forum switch fabric benchmark specifications: An overview. *IEEE Network* 2005; **19**(2):5–9.
26. Bolding K. Non-uniformities introduced by virtual channel deadlock prevention. *Technical Report* 1992.
27. Izu C. Throughput fairness in k-ary n-cube networks. *Proceedings of the 29th Australasian Computer Science Conference - Volume 48*, ACSC '06, Australian Computer Society, Inc.: Darlinghurst, Australia, Australia, 2006; 137–145.
28. VEF traces homepage. http://www.i3a.info/VEFtraces 2017.
29. Andújar FJ, Villar JA, Sánchez JL, Alfaro FJ, Escudero-Sahuquillo J. VEF Traces: A Framework for Modelling MPI Traffic in Interconnection Network Simulators. *The 1st IEEE International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB 2015), co-located with 2015 IEEE International Conference on Cluster Computing (CLUSTER 2015)*, Chicago, IL, USA, 2015; 841–848.
30. Gabriel E, *et al.*. Open MPI: Goals, concept, and design of a next generation MPI implementation. *Proceedings of the 11th European PVM/MPI Users' Group Meeting*, Budapest, Hungary, 2004; 97–104.
31. GALGO - Albacete Research Institute of Informatics Supercomputer Center homepage. http://www.i3a.uclm.es/galgo 2017.
32. HPC challenge benchmark. http://icl.cs.utk.edu/hpcc/index.html.
33. Suzumura T, Ueno K, Sato H, Fujisawa K, Matsuoka S. Performance characteristics of Graph500 on large-scale distributed environment. *2011 IEEE International Symposium on Workload Characterization (IISWC)*, Austin, TX, USA, 2011; 149–158, doi:10.1109/IISWC.2011.6114175.
34. Phillips JC, Braun R, Wang W, Gumbart J, Tajkhorshid E, Villa E, Chipot C, Skeel RD, Kalé L, Schulten K. Scalable molecular dynamics with NAMD. *Journal of Computational Chemistry* 2005; **26**(16):1781–1802, doi: 10.1002/jcc.20289.
35. NAMD- Scalable Molecular Dynamics. http://www.ks.uiuc.edu/Research/namd/ 2016.

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls