Combining Source-adaptive and Oblivious Routing with Congestion Control in High-performance Interconnects using Hybrid and Direct Topologies

PEDRO YEBENES, JOSE ROCHER-GONZALEZ, JESUS ESCUDERO-SAHUQUILLO, PEDRO JAVIER GARCIA, FRANCISCO J. ALFARO, and FRANCISCO J. QUILES,
University of Castilla-La Mancha, Spain
CRISPÍN GÓMEZ and JOSE DUATO, Technical University of Valencia, Spain

17

Hybrid and direct topologies are cost-efficient and scalable options to interconnect thousands of end nodes in high-performance computing (HPC) systems. They offer a rich path diversity, high bisection bandwidth, and a reduced diameter guaranteeing low latency. In these topologies, efficient deterministic routing algorithms can be used to balance smartly the traffic flows among the available routes. Unfortunately, congestion leads these networks to saturation, where the HoL blocking effect degrades their performance dramatically. Among the proposed solutions to deal with HoL blocking, the routing algorithms selecting alternative routes, such as adaptive and oblivious, can mitigate the congestion effects. Other techniques use queues to separate congested flows from non-congested ones, thus reducing the HoL blocking. In this article, we propose a new approach that reduces HoL blocking in hybrid and direct topologies using source-adaptive and oblivious routing. This approach also guarantees deadlock-freedom as it uses virtual networks to break potential cycles generated by the routing policy in the topology. Specifically, we propose two techniques, called *Source-Adaptive Solution for Head-of-Line Blocking Avoidance* (SASHA) and *Oblivious Solution for Head-of-Line Blocking Avoidance* (OSHA). Experiment results, carried out through simulations under different traffic scenarios, show that SASHA and OSHA can significantly reduce the HoL blocking.

CCS Concepts: • **Computer systems organization** → *Redundancy*; • **Networks** → Network reliability;

Additional Key Words and Phrases: HPC, interconnection networks, hybrid and direct topologies, source-adaptive and oblivious routing, congestion management, HoL blocking

This article is a completely new submission, not published previously in any conference or journal.
This work has been jointly supported by the Spanish MINECO and European Commission (FEDER funds) under the project TIN2015-66972-C5-2-R (MINECO/FEDER), by Junta de Comunidades de Castilla-La Mancha under the project SBPLY/17/180501/000498, and by the Excma. Diputacion de Albacete under the project DIPUAB18ESCUDEROSAHUQUIL. Pedro Yebenes was funded by the predoc grant BES-2013-063681, from the Spanish MINECO and European Commission (FEDER funds). Jesus Escudero-Sahuquillo is funded by the University of Castilla-La Mancha (UCLM), with a contract for accessing the Spanish System of Science, Technology and Innovation (SECTI), for the implementation of the UCLM research program (UCLM resolution date: 31/07/2014).
Authors' addresses: P. Yebenes, J. Rocher-Gonzalez, J. Escudero-Sahuquillo (corresponding author), P. J. Garcia, F. J. Alfaro, and F. J. Quiles, University of Castilla-La Mancha, Computing Systems Department, Albacete, 02071, Spain; emails: pedroyebenes@gmail.com; {jose.rocher, jesus.escudero, pedrojavier.garcia, fco.alfaro, francisco.quiles}@uclm.es; C. Gómez and J. Duato, Technical University of Valencia, Computing Engineering, Valencia, 46022, Spain; emails: crigore@gap.upv.es, jduato@disca.upv.es.

1544-3566/2019/04-ART17
https://doi.org/10.1145/3319805

ACM Reference format:
Pedro Yebenes, Jose Rocher-Gonzalez, Jesus Escudero-Sahuquillo, Pedro Javier Garcia, Francisco J. Alfaro, Francisco J. Quiles, Crispín Gómez, and Jose Duato. 2019. Combining Source-adaptive and Oblivious Routing with Congestion Control in High-performance Interconnects using Hybrid and Direct Topologies. *ACM Trans. Archit. Code Optim.* 16, 2, Article 17 (April 2019), 26 pages.
https://doi.org/10.1145/3319805

1 MOTIVATION

The number of computing end nodes in the most powerful high-performance computing (HPC) systems has increased importantly over the decades, as it shows in the evolution of the Top-500 list. In these systems, the interconnection network is a central element that must offer efficient communication among the end nodes. If the network does not satisfy the communication requirements of the applications supported by HPC systems, then it may become the primary system bottleneck and markedly degrade the performance of the whole system. Hence, the performance of the interconnection network is always a priority for interconnect researchers, designers, and manufacturers.

Indeed, in the past decade, many solutions focused on improving different aspects of high-performance interconnection networks have been developed, such as routing policies, power efficiency, or switching features. One of these aspects is the network topology, which defines how the system end nodes are interconnected. The network topology must offer high-communication bandwidth and low-latency while being able to interconnect thousands of end nodes. Moreover, the connection pattern of the topology should provide alternative routes to guarantee that in the event of faults, the routing algorithm can still communicate all the end nodes of the network. These requirements have been accomplished traditionally by direct network topologies, such as Tori topologies, which have been used by some of the most powerful supercomputers in the world, such as *Titan* [3] and *K-Computer* [2]. Although direct networks are significantly cheaper to build compared to other network topologies such as CLOS or Fat-tree networks, they offer a smaller bisection bandwidth that is not likely to be enough for the HPC systems of the future. Note that direct networks leverage communication performed mostly among neighbor end nodes, while HPC applications require many-to-many communication patterns nowadays. In recent years, designers and researchers have proposed several hierarchical network topologies, such as Dragonflies [23] or Slim-flies [6], which focus on reducing the number of required network devices by employing connection patterns of reduced diameter. Based on this idea, hybrid network topologies, such as KNS [28], have been also proposed in the last years for interconnecting thousands of processing and storage end nodes. Like Dragonflies and Slim-flies, KNS topologies offer an excellent performance/cost ratio, mainly because they allow short routes and provide path diversity, which can be leveraged by efficient routing algorithms. Indeed, apart from the topology, the routing algorithm is another critical aspect that impacts on network performance. In that sense, many routing techniques have been designed to leverage the network topologies mentioned above. The proposed routing algorithms are either deterministic, which always communicate two end nodes using the same path, or adaptive/oblivious, which select among all the available routes between two end nodes.

Although the topology and routing proposals mentioned above achieve good performance, when the network traffic load is high, the network may reach the saturation point, then its performance may drop dramatically due to the congestion effects. Congestion is a natural phenomenon in interconnection networks, which arises when several packet-flows persistently contend for the use of network resources, usually links or switch ports. In lossless networks, like those commonly used in HPC systems, the effect of flow control causes that traffic clogging the internal

network paths ends up filling buffers as congestion propagates from where it originates back to the end nodes. Note that packet discarding is not allowed in lossless networks. Hence, congestion situations lead to the growth of congestion trees [15], which propagate quickly from where they originate (i.e., the root of the congestion) throughout the entire network, due to the flow-control backpressure. However, congestion by itself is not responsible for the network performance degradation. Indeed, in the network points where congestion originates, the congested links and ports are used at their maximum capacity. In situations where congestion episodes generate congestion trees spreading throughout the network, the performance degradation occurs as the effect known as *Head-of-Line (HoL) blocking* appears, which leads to wasting link bandwidth and spoiling network performance. In general, HoL blocking happens at the input ports of the switches when a packet, which must access to a free output port, is blocked by the packet at the head of its queue, which is waiting for accessing another output port that is busy at this moment. Hence, a packet that could potentially continue its traversal through the network stops. The HoL blocking effect causes that the average packet latency eventually increases and network throughput decreases. Unfortunately, HoL blocking may also occur in other situations. When this effect appears only in the switch where the congestion originates, it is known as *low-order* HoL blocking [21]. By contrast, when this effect originates in a switch different from where congestion originates, due to the backpressure of the flow control mechanism in upstream switches, it is known as *high-order* HoL blocking [20]. Therefore, HoL blocking spreads throughout the network, reaching the end nodes, if we do not take any countermeasure.

Note that current high-speed network technologies for HPC systems, such as InfiniBand or Intel Omni Path, do not discard packets when congestion appears, in contrast to computer networks like Internet, because of the overhead introduced by packet retransmission.[1] Therefore, HoL blocking is a threat to the performance of interconnection networks used in HPC systems, so that dealing with this threat is an essential issue for current commercial products. In that sense, many solutions have been proposed specially designed to deal with HoL blocking. Many of these techniques divide the buffer space at the switch port into different queues, then mapping packets to these queues so that some traffic flows do not share queues with other traffic flows. Some proposals based on this idea eliminate the HoL blocking [10, 11, 13], but they require additional resources not supported by current commercial switches. However, there exist feasible queuing schemes that eliminate HoL blocking partially [4, 9, 32]. Among the latter, the most efficient schemes are those specially designed for specific network topologies and routing algorithms, as they are aware of both topology and routing features, and this knowledge allows to leverage the network resources available to deal with HoL blocking. These "tailored" queuing schemes are known as topology- and routing-aware. In that sense, we proposed the topology- and routing-aware queuing scheme called BBQ (Band-based Queuing) [34], which is tailored to KNS topologies using the Hybrid-DOR deterministic routing algorithm [28]. BBQ significantly reduces the HoL blocking utilizing a small number of queues per port. However, the Hybrid-DOR deterministic routing algorithm does not exploit the path diversity of KNS topologies, while adaptive and oblivious routing algorithms could be used in KNS, alternatively to deterministic ones, offering the possibility of routing packets through alternative routes. Note that adaptive and oblivious routing algorithms can be used to balance the traffic in the network, so that the link utilization could be more efficient and congestion appearance may be prevented or delayed. Moreover, when congestion situations are strong and the routing algorithm by itself is not able to "dissolve" the congestion trees, we could combine these routing algorithms with queuing schemes further to reduce HoL blocking. However, note that the BBQ

[1] Although Ethernet technology, commonly used in the Internet Data-centers, is evolving to be lossless, it is not still being adopted by the HPC industry in their systems.

technique was devised assuming the use of deterministic routing so that queuing schemes combined with adaptive/oblivious routing are still an open issue for KNS topologies, which we think it is worth analyzing.

In this article, we propose a new approach to deal with HoL blocking in KNS and direct topologies using source-adaptive and oblivious routing algorithms, which could be included with minimal hardware support in current high-speed network technologies, such as InfiniBand. Our approach uses virtual networks (VNs) to separate the traffic flows in different layers. We use separate buffer resources assigning to the VNs different network routes. We define two disjoint set of routes in the network topology (i.e. KNS and Tori), which are associated, respectively, with each one of these VNs to prevent deadlocks. Each of the VNs is composed of several queues, so that traffic flows within the VNs are mapped to these queues based on a queuing scheme to reduce HoL blocking inside that VN. Based on this idea, we have designed two solutions for HoL blocking avoidance and deadlock prevention in KNS and direct topologies, called OSHA (*Oblivious Solution for HoL Blocking Avoidance*) and SASHA (*Source Adaptive Solution for HoL Blocking Avoidance*), which use oblivious and source-adaptive routing, respectively. Note that OSHA and SASHA differ on the type of routing algorithm used in the network. While OSHA is proposed for networks using oblivious routing (i.e., the VN to map the traffic flows is selected randomly regardless the network status), SASHA looks at the occupancy of queues of network interface and maps traffic flows to the VN with lower occupancy. Moreover, both OSHA and SASHA leverage the queues at each VN to reduce the HoL blocking using the same queuing scheme, called Dynamic Band-based Queuing (DBBQ). In general, OSHA and SASHA proposals improve network performance while requiring just a reduced set of network resources, i.e., they need a small number of queues (2 or 4) per port. To evaluate these proposals, we have performed extensive simulation experiments in several KNS and Tori topologies, where we have generated synthetic and trace-based traffic patterns modeling different congestive episodes. The obtained results show that SASHA and OSHA are feasible alternatives to reduce HoL blocking in KNS and Tori topologies using source-adaptive and oblivious routing algorithms. In summary, the main contributions of this article are the following:

- We analyze the congestion dynamics in KNS and Tori topologies using source-adaptive and oblivious routing.
- For these network configurations, we propose a new approach that uses two virtual networks (VNs) to balance the traffic between two disjoint sets of alternative routes. In this way deadlocks are prevented. Besides, several several queues are used per VN to reduce HoL blocking inside the VNs.
- Based on this approach, we propose two techniques: OSHA for KNS and Tori networks using oblivious routing, and SASHA for networks using source-adaptive routing. Both of them use two VNs assigning them a different set of routes, and a queuing scheme, called DBBQ that is based on BBQ [34]. DBBQ maps traffic flows to the queues within a VN, reducing HoL blocking inside that VN.
- We have evaluated OSHA and SASHA through simulation experiments modeling different switch architectures and traffic patterns in networks connecting up to 13K end nodes, looking also at their scalability.
- We provide some implementation details of SASHA and OSHA in InfiniBand-based networks.

The rest of this article is organized as follows. Section 2 overviews the background and concepts used in the article, related to network topologies, routing algorithms, and mechanisms to reduce HoL blocking. Section 3 describes our proposals OSHA and SASHA as well as the DBBQ queuing scheme. Section 4, discusses the implementation of our proposals in InfiniBand-based networks.

Section 5 shows the evaluation results of our proposals. Finally, in Section 6, some conclusions are drawn.

2 BACKGROUND DESCRIPTION

2.1 KNS Network Topologies

Network topologies for HPC systems are traditionally classified as either direct or indirect. On the one hand, direct topologies usually adopt an orthogonal structure, where the end nodes are organized in a n-dimensional space and connected in each dimension according to a ring or array arrangement. 2D or 3D direct topologies are relatively easy to build as each topology dimension is mapped to a physical dimension. Direct topologies with a small number of dimensions tend to have a large number of end nodes per dimension, which also leads to an increase in communication latency. Direct topologies with more than three dimensions imply not only increasing the wiring complexity but also the length of their links when they are mapped to our 3D physical space, thereby increasing the communication latency and negatively impacting performance. For instance, there are several examples of direct topologies in the most powerful supercomputers included in the Top-500 ranking. In the last list of June 2018, we can see the *Gemini* topology [3] in the *Titan* supercomputer (USA), and the *Tofu* interconnect [2] in the *K-Computer* (Japan). On the other hand, the most common indirect topologies are multistage interconnection networks (MINs) where switches are arranged in a set of stages. Indirect topologies usually provide better performance than direct topologies for a large number of end nodes, at the cost of using a high number of switches and links and increasing the wiring complexity, which grows with the size of the network. One example of MIN topologies widely used in HPC systems is the Fat-tree [24]. However, Fat-trees are naturally deadlock-free topologies that offer a higher bisection than direct topologies, a high number of alternative routes and reduced diameter. However, both direct and indirect topologies present significant drawbacks when the number of end nodes to be interconnected is enormous.

Recently, apart from hierarchical topologies, such as Dragonflies [23], Slim-flies [6], or Projective networks [8], hybrid topologies have been proposed to get the benefits from both direct and indirect topologies. In general, the objective of hybrid topologies is to provide high-performance like indirect topologies, but at a similar cost compared to direct topologies. With this objective in mind, the k-ary n-direct s-indirect (KNS) family of topologies were proposed [28]. Specifically, KNS topologies organize end nodes in n dimensions, as in a direct network topology, each dimension having k end nodes, but the end nodes of a given dimension are not interconnected as in meshes or Tori. Instead, these end nodes are connected utilizing an indirect subnetwork such as a simple crossbar or switch. Note, however, that the indirect subnetwork may also consist of several switches arranged in several stages, like Fat-tree networks, instead of a single switch requiring the use of expensive high-radix switches. The number of stages in the indirect subnetwork is given by the s parameter. The three parameters k, n, and s give the name to KNS topologies.

Figure 1 shows an example of a two-dimension (2D) 16-node KNS topology connecting 4 end nodes per dimension, and using 4-port switches in the indirect network (i.e., one stage). Note that the number of end nodes that this topology can interconnect is given by $N = k^n$. The number of switches S is given by the formula $S = (N \times n)/k$. Note that each end node is connected to a different switch in each of the network dimensions. Thanks to this feature, KNS topologies offer a rich path diversity. In Figure 1 two end nodes can communicate through several routes. Moreover, KNS topologies allow routes whose length is short on average, leading to lower network latencies when compared to other network topologies. Both properties ease the implementation of efficient routing algorithms, such as the Hybrid-DOR, source-adaptive and oblivious (See Section 2.2). The KNS family of topologies [28] has been evaluated in comparison to other topologies (Tori,

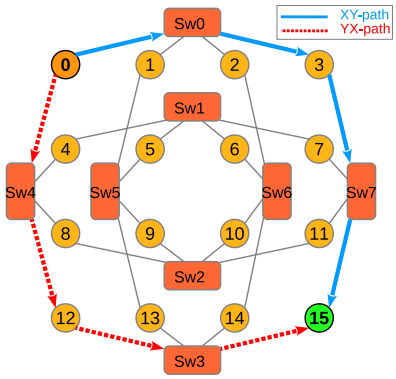


Fig. 1. Example of a 16-node 2D-KNS topology (*4*-ary *2*-direct *1*-indirect). Two paths (XY and YX) are shown for packets from end node 0 to end node 15.

Fat-trees, etc.), comparing cost, performance and complexity, offering a better performance/cost ratio than indirect topologies, if the number of end nodes to connect is high.

Finally, it is worth mentioning that, to build KNS topologies in real systems, it is required that network interfaces offer support for re-routing packets among their ports without the intervention of the host node hardware. For instance, in Figure 1, we need to route packets at node #3 from one port to another to follow the XY path (colored in blue). This means that the network interfaces should offer offloaded capabilities to perform the re-routing function. As far as we know this support is not yet ready in commercial products. However, note that the last InfiniBand-based network interfaces, such as EDR ConnectX-5 and HDR ConnectX-6 HCAs, offer virtual switching offload capabilities in the hardware (i.e., Enhanced vSwitch) "for future protocols" (e.g., the Hybrid-DOR routing for KNS). Therefore, when the re-routing functionality is ready in the network interfaces (which is doable with the current technology), we could build KNS networks.

2.2 Routing Algorithms

The routing algorithm determines the path followed by a packet from its source to its destination, and it should guarantee full connectivity and deadlock-freedom. An efficient routing algorithm balances the traffic flows smartly through the available routes in the network, regardless of the behavior of the traffic generated by the applications. Moreover, efficient routing algorithms minimize the number of hops between any two end nodes, using minimal routes (i.e., routes in the network performing as fewer hops as possible) to reduce the packet latency. Routing algorithms can be classified according to several orthogonal criteria. For instance, the following classification is based on the number of paths available for each source-destination pair:

- *Deterministic routing*. Packets sent from a given source to a given destination always follow the same path. This type of algorithms is the simplest to be implemented in hardware, and therefore these algorithms are widely used in commercial products. A well-known deterministic routing algorithm is Dimension-order Routing (DOR), used in direct topologies such as Meshes and Tori. In this algorithm, dimensions are crossed in a given order, so that all the packets between two end nodes always cross first the intermediate nodes in the first dimension, then the nodes in the second dimension, and so on.
- *Oblivious routing*. Packets from a given source to a given destination can follow different paths available in the topology. For each packet, the path is selected without taking into account the state of the network, using a selection policy such as random or round-robin.

- *Adaptive routing*. Like in oblivious routing, packets sent from a given source to a given destination can follow several paths offered in the topology. However, for each packet, the path is selected based on the state of the network, usually on the occupancy of the buffers.

In KNS topologies, a deterministic routing algorithm is proposed derived from DOR, called Hybrid-DOR [28]. The Hybrid-DOR algorithm implements a policy where network dimensions are crossed in an established order to guarantee deadlock-freedom, as it happens with the DOR algorithm in the direct networks. Specifically, the *X* dimension is crossed first, then the *Y* dimension (i.e., an *X*-*Y* routing algorithm). As it can be seen in Figure 1, the number of hops of each path is independent of the number of end nodes per dimension, and it is always two hops to communicate whatever pair of end nodes. As mentioned above, Hybrid-DOR provides acceptable performance for KNS networks. However, the use of oblivious or adaptive routing algorithms has not been explored yet. In Figure 1, it could be possible for a routing algorithm select between the XY paths (i.e., a packet is first routed in the X dimension, then in the Y) and YX paths (i.e., a packet is first routed in the Y dimension, then in the X). To the best of our knowledge, the benefits of using either oblivious or adaptive routing algorithms in KNS topologies have not been explored yet. Moreover, the effects of HoL blocking in KNS and direct topologies using source-adaptive or oblivious routing algorithms are an open issue.

2.3 Solutions to HoL Blocking

High-performance interconnection networks can use a congestion control technique to alleviate the degradation of network performance derived from congestion. This, for instance, occurs in the InfiniBand-based networks [17, 19]. Nowadays, the two most popular approaches to congestion control in HPC systems are *injection throttling* (i.e., the one used in InfiniBand-based networks) and *queue-based flow-separation*. Injection throttling reduces the packet-injection rate at the source end nodes when the switches detect congestion, and this alleviates the congestion situation after a while. Although injection throttling does not directly eliminate HoL blocking, this effect disappears once congestion is removed. Moreover, injection throttling is not scalable with network size as the delay from congestion detection to reaction at the sources grows with the network diameter [14]. By contrast, queue-based flow-separation strategies are based on having at each port a set of queues (i.e., Virtual Channels (VCs) [9]) to store separately different packet flows, preventing or reducing the impact of HoL blocking. Hence, queue-based solutions directly deal with HoL blocking.

Theoretically, the most efficient queue-based solutions are those that dynamically allocate queues to isolate the packet flows that contribute to congestion (usually referred to as "hot flows"), so that the HoL blocking that these flows could cause over others ("cold flows") is prevented. Some of these solutions, such as the one described for ATLAS [22], identify hot flows based on their final destination and assume that congestion originates only at endpoints. Other solutions such as *Regional Explicit Congestion Notification* (RECN) [11] and *Efficient and cost-effective Congestion-Control* (EcoCC) [14] consider that congestion may originate also at internal points of the network. However, these techniques require additional resources such as mechanisms to detect and separate hot flows, specific control messages, and Content-Addressable Memories (CAM) to keep track of congested points at each port. These resources are not supported by current commercial interconnects.

However, other queue-based solutions map packet flows to queues using a static mapping policy, independently of the traffic conditions. Although these techniques do not require specific resources to evaluate and store information about network status, some of them are not an affordable implementation. For instance, Virtual Output Queues at the network level (VOQnet) [10] uses at each

switch port as many queues as destinations are in the network so that any packet is mapped to the queue corresponding to its destination. Thus, packets share queues only with other packets addressed to the same destination. This queuing scheme prevents low- and high-order HoL blocking, but it is unfeasible due to the high number of queues required per port.

By contrast, other static queuing schemes are far more feasible as they require a reduced number of queues per port. Among them, Virtual Output Queues at switch level (VOQsw) [4] uses at each port as many queues as output ports are in the switch so that each incoming packet is mapped to the queue assigned to its next output port. Hence, VOQsw prevents the low-order HoL blocking, but not the high-order one. Other similar (although not identical) queuing schemes that partially reduce HoL blocking are Dynamically Allocated Multi-queues (DAMQs) [32] or Destination-based Buffer Management (DBBM) [26].

In general, the static queuing schemes mentioned above are not aware of the routing algorithm and network topology. As a consequence, the reduced set of queues per port is not always efficiently leveraged to reduce HoL blocking, and the performance of these techniques may drop in specific topologies when HoL blocking appears. By contrast, other queuing schemes take into account the network configuration, such as Flow2SL [12] and vFTree [18], which are specially designed for fat-trees topologies using deterministic routing algorithm [16, 35], to exploit their characteristics and reduce the HoL blocking more effectively. Similarly, we devised a static queuing scheme specially tailored to KNS topologies using the hybrid-DOR (deterministic) routing algorithm, called Band-based Queuing (BBQ) [34]. Specifically, BBQ reduces HoL blocking in KNS networks using a static queuing scheme based on a simple policy to map packets to queues at each switch port. The idea is to virtually divide the KNS topology into several areas (or bands), and map traffic flows to them so that this reduces HoL blocking. Specifically, at input ports, the BBQ mapping policy selects the queue to store each incoming packet according to the formula in Equation (1):

$$SelectedQueue = \frac{Destination \times \#Queues}{\#EndNodes}, \tag{1}$$

where *Destination* is the destination of the packet, *#Queues* is the number of queues that BBQ configures at each switch port (i.e., the number of queues each port buffer is divided into), and *#EndNodes* is the number of end nodes in the network. This queue-selection policy virtually divides the KNS network into a given number of horizontal bands, each one consisting of one or more consecutive rows of destinations. Destinations in different rows result in a different *SelectedQueue* when they are introduced as a "Destination" in Equation (1). Thus, packets whose destinations are in different rows are always stored in different queues. As a result, packet flows addressed to different rows cannot produce HoL blocking to each other.

Although BBQ significantly improves the performance of KNS networks, its effectiveness is limited by the use of the Hybrid-DOR routing algorithm. We have observed that in KNS networks using the XY Hybrid-DOR version, the links in the X-dimension may be prematurely congested, since packets always move first on the X dimension, and then in the Y one. This situation leads to an unbalanced utilization of the network links and a diminution of the system performance, even if BBQ is used. Therefore, we think that the performance of KNS topologies can be further improved if we exploit their path diversity (i.e., the YX paths) using minimal-path routes. This idea can be extended to direct networks as well. Therefore, to exploit the available routes in KNS and direct topologies, we will use oblivious or adaptive routing algorithms, combined with a suitable queuing scheme aware of both the topology and the routing properties.

3 OSHA AND SASHA DESCRIPTION

In this section, we propose a new approach to deal with congestion and HoL blocking in KNS and direct topologies using oblivious and source-adaptive routing. This approach leverages some of the properties of these topologies and their routing algorithms, such as path diversity and traffic balancing, to reduce HoL-blocking while preventing deadlocks. First, we detail how to exploit path diversity to prevent deadlocks by dividing the network resources in two virtual networks (VNs). Second, we propose *Oblivious Solution for Head-of-Line Blocking Avoidance* (OSHA) for KNS and direct networks using oblivious routing, and *Source-Adaptive Solution for Head-of-Line Blocking Avoidance* (SASHA) for KNS and direct networks using source-adaptive routing algorithms. OSHA and SASHA divide the VNs in several queues and apply the same queuing scheme (called DBBQ) separately to each VN to map traffic flows to the different queues, reducing HoL blocking. Note that we use two different names (OSHA and SASHA) to distinguish when we apply our approach to networks using oblivious routing from when we use source-adaptive routing. Although, hereafter, we refer to these names as "techniques," they correspond to the same approach for reducing HoL blocking in KNS and direct networks using oblivious and source-adaptive routing.

3.1 Deadlock-Freedom Provision

As we have just explained, OSHA and SASHA are both based on the same approach to guarantee deadlock-freedom. Specifically, this approach defines two VNs to offer dedicated buffering to two disjoint set of routes offered by the specific routing algorithm applied to the given network topology. For instance, Figure 2 shows a 16-node 2D KNS topology, using a single switch in the indirect subnetwork. In this figure, physical links connecting end-nodes to switches are divided in two links, to show that the XY and YX paths offered by the Hybrid-DOR routing in the KNS topology can be associated to two different VNs. Note that the paths offered by the "XY" Hybrid-DOR routing (i.e., first the links in the X axis are chosen, then those in the Y axis) are depicted by bold lines colored in blue, and the paths of the "YX" Hybrid-DOR routing are depicted (i.e., first the links in the Y axis are chosen, then those in the X axis) by dashed lines colored in red.

As it has been mentioned above, the typical routing algorithms for direct networks is the well-known dimension-order routing (DOR) policy, which restricts the routes a packet can take to one dimension, then to another dimension (and so on, in the case of more than two dimensions). The DOR algorithm is deadlock-free as it applies the restrictions to the routes, then preventing cycles in the channel dependency graph. Based on DOR, the Hybrid-DOR policy for KNS is deadlock free as well, as it restricts the routes as the DOR algorithm does. However, to leverage the path diversity of both KNS and direct topologies, and to balance link utilization, OSHA and SASHA work with some degree of adaptiveness in the topology. Specifically, based on the DOR routing, packets sent from a given source to a given destination in a two-dimensional (2D) KNS or direct topology can follow two different routes across the network (see Figure 2):

(1) XY path: The X dimension is crossed first, then the Y dimension.
(2) YX path: The Y dimension is crossed first, then the X dimension.

Note that using XY and YX paths simultaneously introduces cycles in the channel-dependency graph (CDG) that could lead to deadlocks if no measures are taken to prevent this situation. For instance, we can consider in Figure 2 that four traffic flows F1, F2, F3, and F4 are generated: F1 is generated from end node 0 to end node 5, F2 from 1 to 4, F3 from 5 to 0, and F4 from 4 to 1. We also assume that it is possible to use XY and YX paths and there are not VNs. Therefore, in this situation a deadlock could appear if F1 uses an XY path, F2 uses a YX path, F3 uses an XY path, and F4 uses a YX path. Note that this situation could be prevented if we separate in different buffering resources packets following XY paths from packets following YX paths.

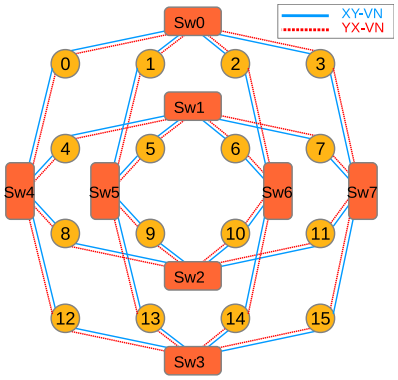


Fig. 2. A 16-node 2D-KNS topology where each physical link between an end node and a switch is decomposed in two virtual links, creating two virtual networks (VNs) in the topology: the XY-VN and the YX-VN.

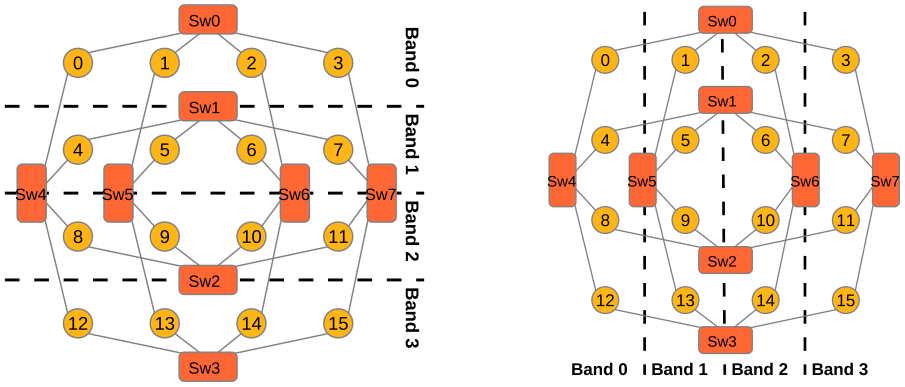
In the case of three-dimensional (3D) KNS and direct topologies (e.g., Tori) using the DOR routing, we could define several types of paths as well: XYZ, XZY, YXZ, YZX, ZXY, and ZYX. Note that in this case, we would need six VNs to guarantee deadlock prevention, if we follow the approach of separating types of paths in different VNs. Although other approaches could be used to reduce the number of required VNs, we decided to limit the adaptiveness degree to two types of paths, since the number of queues is limited in current network technologies (e.g., InfiniBand-based switches offer eight VLs). To increase the adaptivity degree optimizing the VNs is left for future work.

Therefore, we assume two VNs and assign to them two disjoint set routes, offered by the routing algorithm. The VN selected by a packet cannot be changed once that packet is injected in the network, as we describe in the next section. Thanks to this approach cyclic dependencies are not possible in the CDG of the KNS and direct topologies using oblivious and source-adaptive routing. Therefore, OSHA and SASHA prevent deadlocks, while they can reduce the HoL blocking using two VNs, and several queues per VN, as we explain in Section 3.3.

3.2 Virtual Network Selection

The VN in which a packet is going to be mapped is selected by the source end node based on the routing algorithm (i.e., oblivious or source-adaptive). As we have described before, this VN cannot be changed once the packet is injected into the network. In the case of 2D topologies, the queues of the first VN are used exclusively by packets that follow XY paths, whereas the queues of the second VN are used exclusively by packets that follow YX paths. In the case of KNS of direct topologies using more dimensions, we need to select only two type of paths among the available ones (e.g., XYZ and YXZ).

Hence, a packet following a XY path will never be stored in queues assigned to packets following YX paths (and vice versa). OSHA and SASHA differ in the criterion for the VN selection, since that criterion depends on the routing algorithm. On the one hand, OSHA uses a round-robin policy so that packets injected consecutively from a source end node are shuffled between both VNs. As a consequence, the XY and YN VNs receive the same amount of traffic regardless the traffic conditions in the network, thereby "proactively" balancing the load of X-dimension and Y-dimension links. On the other hand, SASHA selects the VN depending on the occupancy of the output queues of the source end node. In this case, the queue with lower occupancy among those where the packet could be mapped (according to the queuing scheme) is selected, then the VN that includes this queue is selected. In this way, SASHA "reactively" balances the utilization of



(a) Horizontal bands (XY-path)

(b) Vertical bands (YX-path)

Fig. 3. DBBQ divides a 16-node 2D-KNS topology into four horizontal and vertical bands.

both VNs, based on the occupancy at source nodes. It is worth mentioning that SASHA only needs the information of the occupancy of the queues at source end nodes in order work properly, not requiring additional information from switches, control messages and notifications to/from the network.

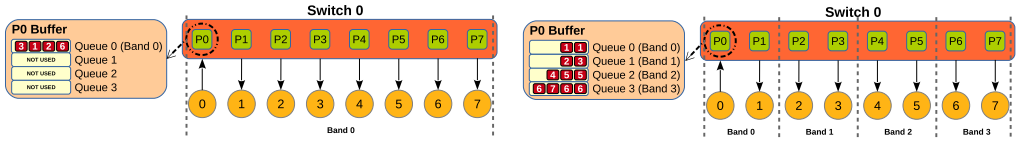
Note that SASHA can react to congestion in a VN, in contrast with OSHA, which is not aware of any sign of congestion. For instance, if a VN is congested, OSHA keeps shuffling packets between both VNs. By contrast, in the same scenario, SASHA stops using the congested VN and would only use the other one while congestion lasts. However, SASHA requires a mechanism to monitor the queue occupancy at egress ports of end nodes of the network, as we describe in Section 4.

3.3 HoL Blocking Reduction

As we have explained before, an additional benefit of using two separate VNs is that HoL blocking cannot happen between two packet flows assigned to different VNs, as these flows would never share queues. However, HoL blocking is still possible among flows assigned to the same VN. To reduce this "intra-VN" HoL blocking, OSHA and SASHA divide each VN in several queues, and use an efficient and suitable queuing scheme, which operates independently at each VN, to map packets to a specific queue inside the corresponding VN.

Note that we do not restrict the queuing scheme that could be used at each VN, since similar ideas to those of OSHA and SASHA could be applied to other network topologies. However, to provide an efficient HoL blocking reduction, we have adapted the BBQ technique (see Section 2.3) to OSHA and SASHA, since BBQ is a successful queuing scheme for KNS networks. Based on BBQ, we have proposed a new queuing scheme, called Dynamic Band-based Queuing (DBBQ), which selects horizontal or vertical bands depending on the VN selected. Figure 3 shows an example of DBBQ for a 16-node 2D-KNS topology.

Note the total number of queues required per port (n) is the number of queues per port required by the queuing scheme multiplied by the number of VNs (two in our case). So at each port we need a number of queues between 0 and $\frac{n}{2} - 1$ for the first VN (XY path), whereas queues between $\frac{n}{2}$ and $n - 1$ are used by the second VN (YX path). For instance, in the example of 3, we can use four queues per port (one per band), so that queues 0 and 1 could be used by the first VN whereas queues 2 and 3 by the second one. However, more queues can be used per VN to increase the efficiency in HoL blocking reduction.



(a) Packets mapping using BBQ in YX bands

(b) Packets mapping using DBBQ in YX bands

Fig. 4. Buffer occupancy of switch port #0 in 16-node 2D KNS network using BBQ (left) and DBBQ (right).

Although we could use BBQ to reduce HoL blocking inside each VN, it does not fit them optimally, since it was devised considering the use of the deterministic "XY" Hybrid-DOR routing. Indeed, as explained in Section 2.3, BBQ efficiency is derived from the fact that packets traversing the same horizontal band but are addressed to different bands do not share queues, so that HoL blocking is prevented among them. However, when YX paths are used by oblivious or source-adaptive routing algorithms, packets traversing the same horizontal band are already in their destination band. Thus, if we use BBQ in this case, all of these packets will be stored in the same queue, the remaining queues being wasted. Figure 4(a) shows this problem. As we can see, the mapping performed by BBQ is unfortunate, and all the packets are mapped to the same band (i.e., to the same queue).

To overcome this problem, we propose an alternative queuing scheme that we call *Dynamic Band-based Queuing* (DBBQ). Basically, DBBQ divides virtually and dynamically the network into horizontal or vertical bands depending on the VN used to send the packet (first or second VN): in the case of the VN #1 the bands are horizontal like in Figure 4(a), but vertical bands are used in the case of the VN #2, like in Figure 4(b). Note that for 3D KNS topologies, we assume that bands in 2D topologies transform into horizontal or vertical planes. Like in BBQ, each band (or plane) corresponds to an exclusive queue where the packets addressed to this band (or plane) are mapped. Note that, in 3D KNS or direct topologies, packets following XYZ paths are mapped to VN #1, then to queues corresponding to horizontal bands (or planes), and packets following YXZ paths are mapped to VN #2, then to queues corresponding to vertical bands (or planes). In this way, we do not waste queues when packets are mapped to VN #2 (YXZ paths), as in this case packets traversing a vertical band (or plane), but addressed to different bands (or planes), never share queues. Figure 4(b) shows the mapping performed by DBBQ using VN #2. As can be seen, this mapping policy solves the problems of BBQ (see Figure 4(a)). Note that in the case of 3D KNS and direct topologies, we assume the use of two VNs, as well, therefore the routes in the network are limited to XYZ and YXZ paths.

DBBQ uses two different formulae to calculate the queue where each packet is stored based on the XY and YZ paths for 2D topologies, and the XYZ and YXZ paths for 3D topologies. Specifically, the formula used by DBBQ when a packet is sent to VN #1 is shown in Equation (2), which is similar to the one used by BBQ (see Equation (1)), but now the $\#QueuesPerVN$ parameter represents the number of queues configured for each VN, instead of the total number of queues per port. However, the formula used for packets sent to VN #2 divides the network into vertical bands. We show this formula in Equation (3), where $Destination$ is the destination of the packet, $\#QueuesPerVN$ is the number of queues configured for each VN, and $\#NodesPerDim$ is the number of end nodes in each dimension:

$$SelectedQueueVN1 = \frac{Destination \times \#QueuesPerVN}{\#EndNodes}, \tag{2}$$

$$SelectedQueueVN2 = \frac{(Destination \,\%\, \#NodesPerDim) \times \#QueuesPerVN}{\#NodesPerDim}. \tag{3}$$

Note that the result of both formulae is a number of queue relative to the number of queues assigned to each VN, i.e., not relative to the total number of queues per port.

4 INFINIBAND IMPLEMENTATION

In this section, we provide some details to implement OSHA and SASHA in InfiniBand-based networks. The InfiniBand Architecture (IBA) offers flexibility to build different network topologies and implement different routing algorithms. We assume that the network interfaces of the last generation of IBA hardware can re-route packets in the end node cards without any software participation, as vSwitch and offloading capabilities have been included (see Section 2.1). The vSwitch functionality is in charge of routing packets from the first port to the second in two-port IBA host channel adapters (HCAs). In the case of 3D KNS topologies, we would need offloading functionality to perform the communication between two HCAs through PCIe, without involving the CPU. As far as we know, a similar offloading capability is available in current IBA hardware in the technology GPUdirect, which communicates GPU devices and HCAs without the CPU (nor software) intervention.

In IBA, the routing algorithm is implemented by the Subnet Manager (SM), a software entity that discovers the network topology, assigns local identifiers (LIDs) to all the end nodes, and populates the switch routing tables based on the selected routing algorithm. Besides, IBA considers that HCAs at end nodes can be connected to the network through different ports, so that the SM assigns LIDs to the ports of every HCA. Indeed, the addressing of the same end node by means different LIDs allows that several can be configured in the network to reach the same end node. This functionality is required to implement either OSHA or SASHA in IBA. Furthermore, the SM needs to discover XY and YX paths (XYZ and YXZ paths in the case of 3D topologies) to implement the oblivious and source-adaptive routing policies, and the Hybrid-DOR routing. Note that Hybrid-DOR is based on the classic DOR routing algorithm for direct networks, which is already implemented in current versions of the SM, such as OpenSM (included in the IBA software stack OFS). Therefore, it would be easy to implement both XY and YX Hybrid-DOR by doing some changes in OpenSM. Note that using one or another routing depends on the VN selection, and we still need to find a way for mapping traffic flow to queues inside each VN.

In that sense, OSHA and SASHA require to manage the buffers at switch ports to implement the respective queues of the two separate VNs. The buffering at the ports in IBA devices is channeled through Virtual Lanes (VLs), where each VL has its flow control. VLs are assigned to packets based on their Service Level (SL), which is placed in the packet header before their injection in the network. The SL cannot be modified once the packet is injected into the network. As the packet traverses the fabric, the SL determines which VL is used to store the packet in the next visited port. In the switches, we need to configure the correspondence between SL and VLs. For this purpose, each IBA device port has an SL-to-VL mapping table, which is consulted each time an incoming packet needs to know the VL in where it is stored based on the SL at its header. We have defined the criterion SL=VL to map packets to VL based on the SL. However, it is possible for the packet that traverses the network to modify the VL it is stored. This functionality is available in some strategies [30] that differ from us in the way the SL-to-VL tables are filled in. It is important to mention that IBA defines a maximum of 16 different SLs and VLs, although some manufacturers, such as Mellanox, build IBA devices supporting 16 SLs but only 9 VLs. As one of these VLs is reserved for control packets, a maximum of 8 VLs per port is available in total, so that, to implement OSHA and SASHA, each VN would consist of a maximum of 4 VLs. Note that a specific VL would belong to only one VN so that packets assigned to different VNs would never share a VL.

Besides, as OSHA and SASHA need to know in advance the VN and the VL where a packet must be stored, we need to implement in OpenSM a method to calculate first the VN, then the VL for

Table 1. Evaluated Network Topology Configurations

#	Topology	#Endnodes (also Routers)	#Switches (Indirect networks)	#Switch ports (Endnodes per dimension)
1	2D-KNS (48-ary 2-direct 1-indirect)	2,304	96	48
2	3D-KNS (24-ary 3-direct 1-indirect)	13,824	1,728	24
3	2D-Torus (48×48)	2,304	—	—
4	3D-Torus ($24 \times 24 \times 48$)	13,824	—	—

every packet prior their injection. As the OpenSM is a centralized process in an end node or a switch in the network, the decision to select the VN can be delegated to the connection manager (CM), available at each HCA. As mentioned in Section 3, OSHA and SASHA use different criteria to select the VN. On the one hand, OSHA (oblivious routing approach) uses a round-robin policy, whereby the CM has to keep track of the VN selected for the last packet injected from a given source, then selecting the other VN for the next packet to be injected from that source. On the other hand, SASHA (source adaptive routing approach) would require to check the occupancy of the VLs at the end node, where a packet could be mapped depending on the VN, selecting the VN corresponding to the VL with lower occupancy. This occupancy can be estimated in runtime looking at the end-node performance counters, which store monitoring data regarding the status of the inbound and outbound traffic at the end node ports. We propose to use two port counters: *interval* that measures the number of seconds per interval, and *PortVLXmitWait* that measures the number of hardware cycles in the last *interval*, which the VL could not send data due to insufficient credits (i.e., contention). Individually, SASHA could configure the CM to compare the *PortVLXmitWait* values for all the VLs at the end-nodes network adapters, then selecting the VL with the lowest *PortVLXmitWait* value and the VN that VL belongs to. Moreover, a small *interval* value should be considered to react quickly to congestion.

Note that packets must be mapped to the queues (i.e., VLs) of each VN according to either Equations (2) or (3), described in Section 3.3. Therefore, OpenSM needs to calculate the SL of each packet before their injection in the network, based on those formulae. Precisely, the application of the DBBQ formulae (Equations (2) and (3)) in an IBA context to compute the packet SL (and VL) would require to define an ordering to the end nodes of the topology (from 0 to $N-1$, being N the number of end nodes), so that LIDs can be mapped to this ordering. In this way, it would be easy to apply and implement the above formulae in OpenSM.

5 EVALUATION

In this section, OSHA and SASHA are evaluated based on simulation results obtained in different network configurations where different traffic scenarios are modeled. For this evaluation, we have used a custom-made simulator [33] based on the OMNeT++ framework. The simulation tool is a discrete-event system simulator that accurately models interconnection networks at packet level. This means that we model the network components, such as network interfaces and switches components (i.e., buffers, crossbars) and links with high granularity. For instance, we are able to generate a message in the network interfaces, which is later split into packets, and then injected one by one in the network. Also, the timing is modeled so that events machine triggers the network behavior conveniently. Note that the OMNeT++ framework offers many interesting features such as a simulation kernel, a graphical interface (GUI), and an analysis tool, and it is widely used by the community, due to its accuracy, when modeling link transfers, buffers, and so on.

Specifically, we have modeled the configurations presented in Table 1 for KNS and Tori topologies. For the sake of clarity, the Tori direct topologies are included in the experiments to show other

environments where the proposed queuing schemes can also operate satisfactorily. Note that each configuration has a different size, as we also want to test the OSHA and SASHA scalability.

For all the network configurations, we assume serial full-duplex pipelined links with 12.5GB/s (i.e., 100Gbps) of link bandwidth, 6ns of link propagation delay (i.e., a length of 1.2m and a delay of 5ns/m), both for switch-to-switch and node-to-switch links. These values are similar to those used in the InfiniBand specification. Regarding the switch model, we assume an input-queued (IQ) switch architecture, i.e., buffers are present only at switch input ports. We have modeled switches with 24 and 48 ports, as these are common configurations in several commercial switches. Similarly, in all the cases the switching technique is Virtual Cut-through, the flow-control policy is credit-based, and packet MTU size is 4KB. The switch buffers size is 128KB (i.e., 32 packets can be stored), and they are organized in queues (or virtual channels, VCs) where queuing schemes can be applied.

Furthermore, the IQ-switch architecture can be improved to natively reduce HoL blocking. This can be done by implementing virtual output queues (VOQs) through demultiplexed accesses to the crossbar. The use of VOQs requires that each input port owns a separate entry at buffers to the crossbar per output port in the switch. In this way several packets from the same queue can be simultaneously forwarded if they are addressed to different output ports, thus low-order HoL blocking is completely prevented. Note that VOQs are orthogonal to queues, and indeed the flow-control policy is applied only at queue-level. Hence, the combination of VOQs and queuing schemes increase the HoL blocking reduction. To analyze the performance of OSHA and SASHA proposals, we compare them to other state-of-the-art solutions for HoL blocking reduction. In more detail, we have modeled the following schemes:

- **Single Queue** (1VC). In this scheme, there is a single queue per buffer and deterministic routing is used. This configuration represents the situation when no queuing scheme is used for HoL blocking reduction. We use this configuration to measure the real impact of HoL blocking on network performance, when we do not have any mechanism to reduce congestion effects.
- **Band-based Queuing** (BBQ-4VC). This scheme is used in KNS and direct topologies when deterministic XY DOR routing is used (see Section 2.3). We have modeled BBQ with four queues (or VCs) per buffer. It is used to show the best performance obtained in networks using distributed routing together with HoL blocking prevention. Note that BBQ cannot be combined with oblivious and source-adaptive routing, since the use of alternative routes without using VNs incurs in cycles in the channel dependency graph, then potential deadlocks may appear.
- **Oblivious Solution for Head-of-line Blocking Avoidance** (OSHA). We have modeled OSHA with two VNs, and two different configurations for the number of queues used per VN. Specifically, OSHA-2VC uses one queue per VN (i.e., two VCs are used in total), and OSHA-4VC-DBBQ uses two queues per VN (i.e., four VCs are used in total) to reduce HoL blocking. In this latter case, DBBQ is used as the queuing scheme the VNs. The VN to inject a packet is selected according to an oblivious (round-robin) policy (see Section 3).
- **Source-Adaptive Solution for Head-of-Line Blocking Avoidance** (SASHA). SASHA is also modeled with two VNs. SASHA-2VC uses two queues per VN (i.e., two VCs in total), and SASHA-4VC-DBBQ also uses two VNs and two queues per VN (i.e., four VCs in total). The VN to inject a packet is selected according to a source-adaptive criterion (see Section 3), so that the VN is selected in the source end node according to the lower occupancy.

Finally, we assume that end nodes are connected to switches through network interfaces, modeled with as many admittance queues as end nodes are in the network. Each newly generated

Table 2. Zipf Probability Distribution (in %) for the First Ten Preferred Destinations (i = 1 to 10) for a 64-node Network

s	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
1	21.1	10.5	7.0	5.3	4.2	3.5	3.0	2.6	2.3	2.1
2	61.4	15.3	6.8	3.8	2.5	1.7	1.3	1.0	0.8	0.6
3	83.2	10.4	3.1	1.3	0.7	0.4	0.2	0.2	0.1	0.1

packet is stored in the admittance queue assigned to its destination so that HoL blocking is prevented at traffic-generation level. Packets are transferred from admittance queues to injection queues, which are organized according to the same scheme used in the switches. Injection queues are flow-controlled from the switch input ports at link-level.

5.1 Traffic Modeling

Regarding traffic modeling in simulation experiments, we have used several communications patterns being representative in current HPC applications. Indeed, the threads of a parallel application tend to use preferred destinations due to the characteristics of the common collective-communication schemes. For that reason, as we want to use a more realistic traffic distribution, we have modeled the Zipf traffic distribution [7]. This traffic model is based on Zipf's law, and it is suitable for the evaluation of high-performance interconnects [25].

Specifically, Zipf's law is applied to interconnection networks as a traffic distribution by using Expression (4), where N is the number of end nodes in the network, s is a positive real number that indicates the distribution order, and $P_N(i)$ is the probability that a packet is addressed to the i'th most-preferred destination, the value of i ranging from 1 to N. That is, Equation (4) establishes a ranking of destinations according to the probability of being the destination of a packet:

$$P_N(i) = \frac{i^{-s}}{\sum_{j=1}^{N} j^{-s}}. \tag{4}$$

Table 2 shows an example of the computed Zipf probability distribution for the first tenth preferred destinations in a 64-node network topology, when the value of s is 1, 2, or 3. Note that the value of the s parameter affects strongly to the probability obtained. If the value of s is zero, then Zipf behaves as the uniform one; the higher the value of s parameter, the higher the probability of the packet being sent to the first destinations in the ranking. Note also that Equation (4) does not assign a probability to a specific end node, but to a position in the ranking, i.e., the ith preferred destination could be whatever end node in the network. Therefore, it is necessary to associate each value of i (so each value of $P_N(i)$) to the identifier of an actual end node. In our experiments, we have made this association randomly, for each source of packets, so that each source has its own, different ranking of preferred destination end nodes.

The problem with the Zipf traffic is that, with high values of s, it behaves as a many-to-one traffic pattern where all the end nodes send traffic to a few destinations. Then, congestion situations do not generate a significant amount of HoL blocking to measure the efficiency of queuing schemes. For this reason, we have modeled hot-spot traffic scenarios intended to represent extreme scenarios with intense traffic being suddenly generated from many sources towards the same (one or more) preferred destinations (i.e., hot-spot destinations). Hence, one or several congestion trees are created whose roots are located at the hot-spot destinations. In these situations, congestion trees rapidly spread throughout the network affecting traffic flows that do not contribute to congestion, thus suffering HoL blocking. Specifically, in hot-spot traffic scenarios, a percentage of

sources (in our case 75% of source end nodes) generate traffic flows with a uniform (i.e., random) distribution of destinations, while the remaining sources (in our case 25%) generate a large traffic burst addressed to a single hot-spot destination.

In all experiments where Zipf and hot-spot traffic patterns are used, we measure as performance metrics the *average packet latency* in μs as a function of the *accepted traffic* normalized against the maximum bandwidth of the network links. The accepted-traffic metric measures the amount of traffic that the network can absorb an deliver, and it depends on the traffic generation rate. This traffic rate is incremental, so that we increase that from 0% of the link speed to 100%, and simulate 11 load points. For each traffic load point, the network warms up for 1ms. After the warming period, the performance metrics are recorded in a steady state during 2ms. Therefore, we simulate 3ms of time for each load point.

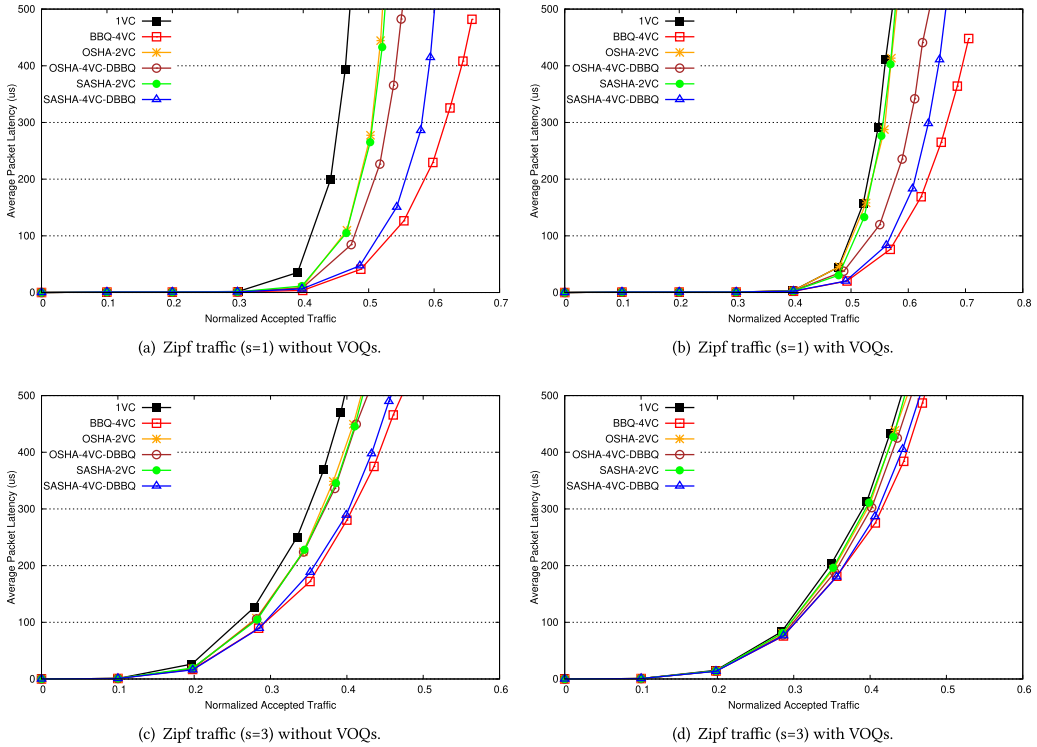
Finally, we have used real-application traces from the HPCC (*High-performance Computing Challenge*) benchmark [1], obtained with the VEF framework [5]. HPCC is a suite of MPI-based tests widely used to measure the performance of processor, memory subsystem and the interconnection network of HPC clusters We have chosen the PTRANS test from HPCC, which generates a moderate load in the network. PTRANS servers to analyze if the observed behavior with synthetic traffic also happens with trace-based traffic. Note that we have used traces obtained for 576 MPI tasks, so that we need four traces in a 2,304-node network, such as those modeled in the network configurations #1 and #3 in Table 1. The performance metric we have obtained from traces execution is the the *execution time* (in milliseconds).

5.2 Zipf Traffic Results

Figure 5 shows performance results for 2304-node 2D-KNS topologies (Network Configuration #1 of Table 1) when Zipf traffic is generated ($s = 1$ and $s = 3$).

Note that switches are configured with and without VOQs, so that in the former case low-order HoL blocking is natively prevented. 1-VC configuration (i.e., a single VC per switch port) always obtains the worst results as it is unable to deal with HoL blocking. BBQ-4VC always achieves the best results, since it fits the network and routing properties. When Zipf is configured with $s = 1$, it generates an all-to-all communication pattern where the 1st destination (see Table 2) receives a higher percentage of traffic, but all the remainder end nodes also receive a representative amount of traffic. Therefore, the traffic pattern generates a weak congestion tree addressed to the 1st destination, while the background traffic addressed to the remainder destinations follows a uniform (i.e., random distribution). For this reason, OSHA-2VC and SASHA-2VC suffer from the high-order HoL blocking generated by the traffic pattern, as they only have two VNs and one queue per VN. OSHA-4VC outperforms OSHA-2VC and SASHA-2VC, but its performance is still far from that of BBQ-4VC (i.e., deterministic routing), because oblivious routing spreads HoL blocking in the available VCs. SASHA-4VC behaves like BBQ-4VC, because source-adaptive routing is able to mitigate moderate congestion situations. By contrast, Zipf configured with $s = 3$ sends most of the traffic to the 1st destination, generating a many-to-one traffic pattern, while the background traffic is reduced, HoL blocking appearance is minimal. Hence, we do not see significant variations between 1VC (the worst case) and BBQ-4VC (the best case).

Similarly, Figure 6 shows performance results for 13,824-node 3D-KNS topologies (Network Configuration #2 of Table 1). Note that the series have virtually identical shape and values than in the previous figure; thus, we can draw similar conclusions as before, regardless the size of the topology is 6× bigger. It is worth mentioning that 3D-KNS topologies increase path diversity, and this favors the adaptive routing algorithms that spread better the traffic in 3D-KNS topologies than in two 2D KNS. For this reason, SASHA-4VC-DBBQ achieves the best results in this case, while SASHA-2VC performance equals that of BBQ-4VC using half the VCs or queues.



(a) Zipf traffic (s=1) without VOQs.

(b) Zipf traffic (s=1) with VOQs.

(c) Zipf traffic (s=3) without VOQs.

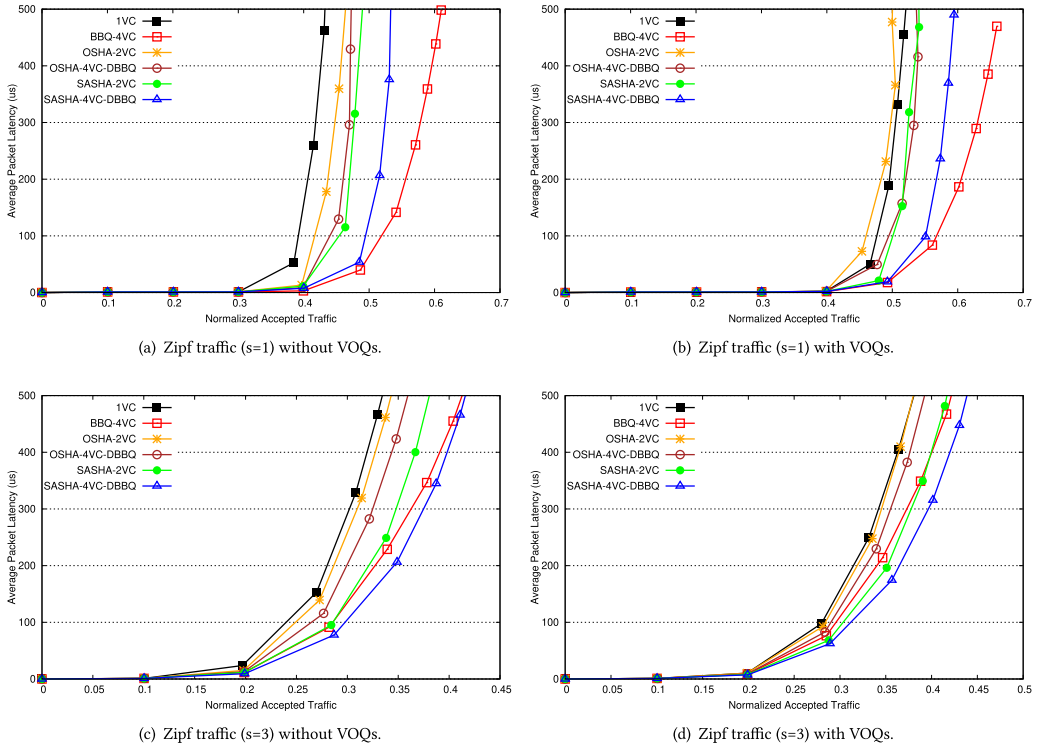
(d) Zipf traffic (s=3) with VOQs.

Fig. 5. Average packet latency (microseconds) versus normalized accepted traffic for 2,304-node 2D-KNS topologies (Network Configuration #1 of Table 1) with and without VOQs, when Zipf traffic is generated ($s = 1$ and $s = 3$).

Figure 7 shows performance results for 2,304-node 2D-Torus topologies (Network Configuration #3 of Table 1). As we can see, the saturation point in Tori topologies is significantly lower compared to KNS topologies, because the bisection bandwidth of KNS topologies is considerably higher compared to Tori. Moreover, Tori topologies are proposed to deal with traffic patterns of applications, which mostly communicate neighbor end nodes. Hence, Zipf traffic that communicates all-to-all when $s = 1$ and many-to-one when $s = 3$ spoils the performance of Tori topologies, regardless of the used queuing schemes and routing policy.

However, in Figure 7, we can observe that differences among techniques are bigger when Zipf traffic is configured with $s = 3$, instead of what happened in KNS topologies. This means that congestion trees in Tori are more dramatic when the traffic pattern is many-to-one, and queuing schemes are required. As network diameter is also bigger, then path diversity is leveraged by the adaptive and oblivious routing algorithm better than in the deterministic case. For this reason, SASHA and OSHA (configured with two and four queues) outperform BBQ-4VC in Tori topologies. It is important to mention that switches using VOQs do not achieve a significant performance increment, compared to switches without VOQs, because congestion dynamics in Tori lead to aggressive congestion trees that spread throughout the entire network generating high-order HoL blocking mostly. Thus, VOQs become useless in these scenarios.

Figure 8 shows performance results for 13,824-node 3D-Torus topologies (Network Configuration #4 of Table 1). As these networks use three dimensions, the path diversity and network bisection bandwidth increase compared to 2D Tori. Note that the number of end nodes in this



(a) Zipf traffic (s=1) without VOQs.

(b) Zipf traffic (s=1) with VOQs.

(c) Zipf traffic (s=3) without VOQs.

(d) Zipf traffic (s=3) with VOQs.

Fig. 6. Average packet latency (microseconds) versus normalized accepted traffic for 13,824-node 3D-KNS topologies (Network Configuration #2 of Table 1) with and without VOQs, when Zipf traffic is generated ($s = 1$ and $s = 3$).

figure is 6× higher than the previous figure, and the performance results are similar than before, except for BBQ-4VC that achieves the best results favored by the increased bisection bandwidth.

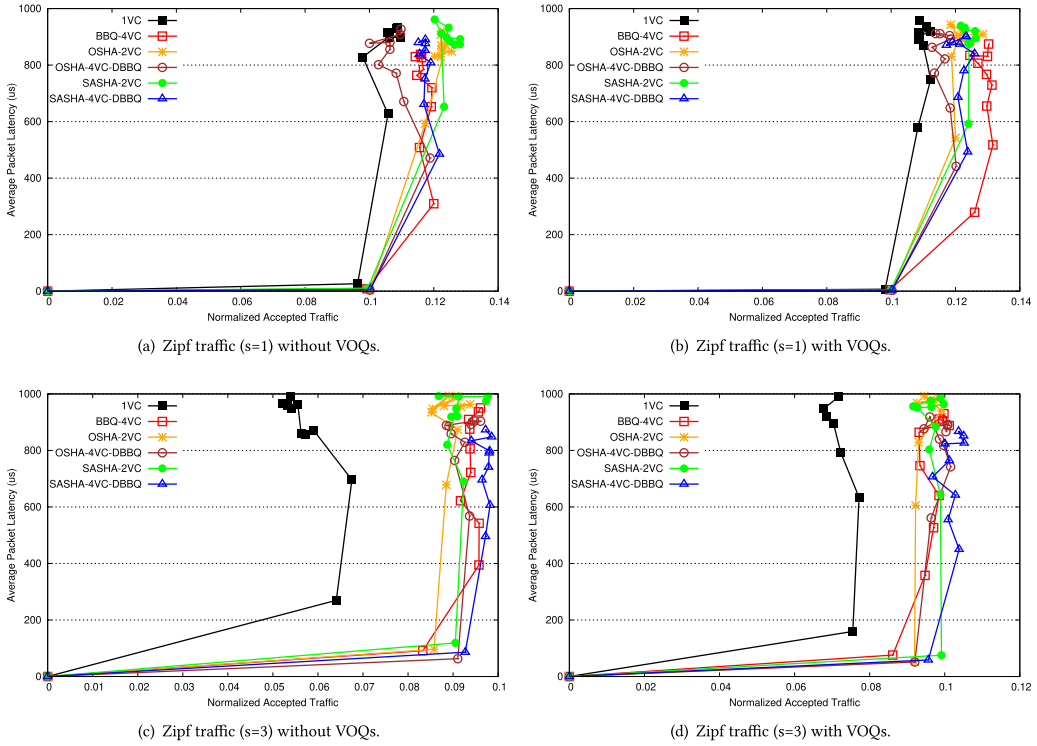
As a conclusion, OSHA and SASHA achieve reasonably good performance results when Zipf traffic is generated in the network. Next section shows performance results when hot-spot scenarios creating stronger congestion situations are generated in the network.

5.3 Hot-Spot Traffic Results

To understand the following results, we need to detail how we set up the simulation experiments using hot-spot traffic. In this traffic pattern 25% of the end nodes generate hot-spot traffic at a given rate of its link bandwidth limit, which corresponds to the the load point in the network.[2] Meanwhile, 75% of the end nodes generate traffic addressed to a uniform (i.e., random) distribution of destinations. As described in Section 5.1, each load point in the plots corresponds to 3ms of simulation time, where 1ms is devoted to warm-up the network (i.e., load it with enough traffic to perform accurate performance metrics) and 2ms are used to measure statistics. After the warm-up period, we mark a set of packets generated to random destinations to measure the packet latency when they arrive at their destinations.

Figure 9 shows performance results for 2D- and 3D-KNS networks (i.e., network Configurations #1 and #2 of Table 1) when we generate the hot-spot traffic scenario described in Section 5.1.

[2]Note that we generate 11 points with different traffic generation load ranging from 0% to 100% of link bandwidth.



(a) Zipf traffic (s=1) without VOQs.

(b) Zipf traffic (s=1) with VOQs.

(c) Zipf traffic (s=3) without VOQs.

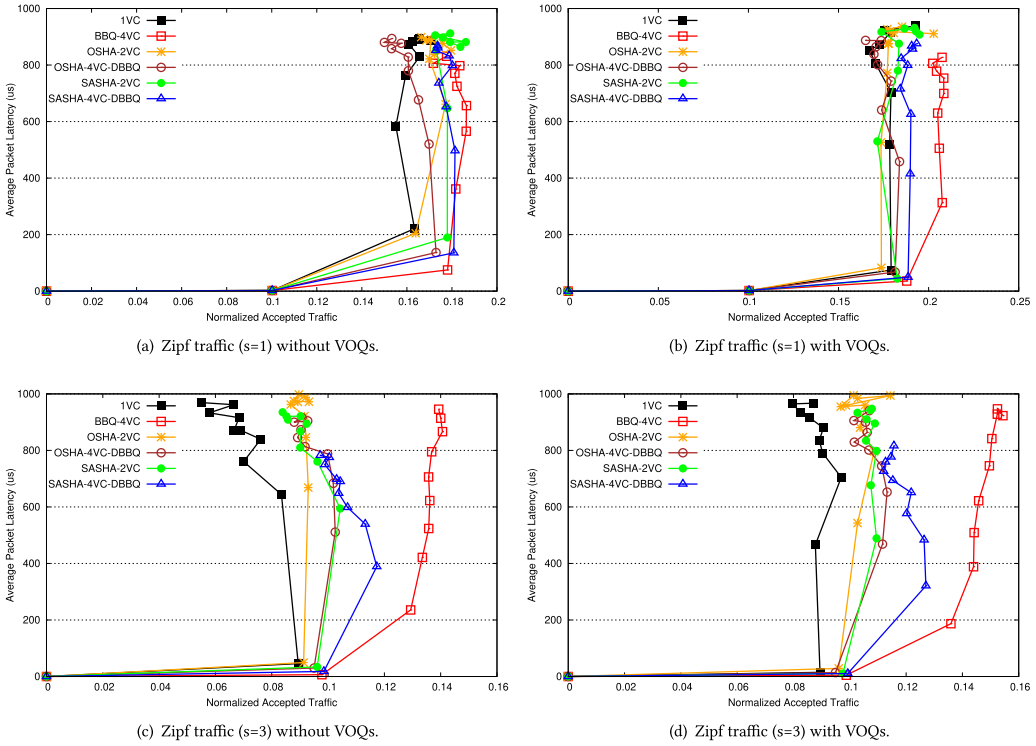
(d) Zipf traffic (s=3) with VOQs.

Fig. 7. Average packet latency (microseconds) versus normalized accepted traffic for 2,304-node 2D-Torus topologies (Network Configuration #3 of Table 1) with and without VOQs, when Zipf traffic is generated ($s = 1$ and $s = 3$).

As we can see, in Figure 9(a) all the queuing schemes but BBQ-4VC and SASHA-2VC-DBBQ suffer from the effects of internal contention at switches near the hot-spot end node (low-order HoL blocking). In particular, SASHA-4VC-DBBQ balances better the traffic through alternative routes to mitigate this problem, as the source-adaptive routing realizes of congested routes at small load rates, while oblivious and deterministic routings do not. The strange effect with latency at small loads occurs because the HoL blocking impacts more on these techniques when the traffic load is moderate. The same happens to SASHA-4VC-DBBQ when VOQs are used. When VOQs are used (Figure 9(b)) the internal contention at switches disappears (as well as low-order HoL blocking); thus, the previous effects with small load rates in the network disappear. In these scenarios, 1VC and OSHA are not able to deal with congestion effects. In the case of 1VC, a single VC or queue is not enough to deal with HoL blocking, regardless of the routing algorithm (in this case 1VC uses deterministic routing). In the case of OSHA, oblivious routing ends up spreading the congestion throughout all the paths and queues of the network.

However, SASHA-4VC-DBBQ performs efficiently (at the level of BBQ-4VC), and the network can absorb traffic until 60% of traffic load.[3] Note that the network latency does not augment after 60%, since the length of the Y-axes is in the order of microseconds (μs), as we want to show the effects of congestion in queuing schemes other than BBQ-4VC and SASHA-4VC-DBBQ. When the

[3]Note that the maximum load that the network can absorb is 75% that coincides with the maximum generation rate of 75% of end nodes generating random traffic.



(a) Zipf traffic (s=1) without VOQs.

(b) Zipf traffic (s=1) with VOQs.

(c) Zipf traffic (s=3) without VOQs.

(d) Zipf traffic (s=3) with VOQs.

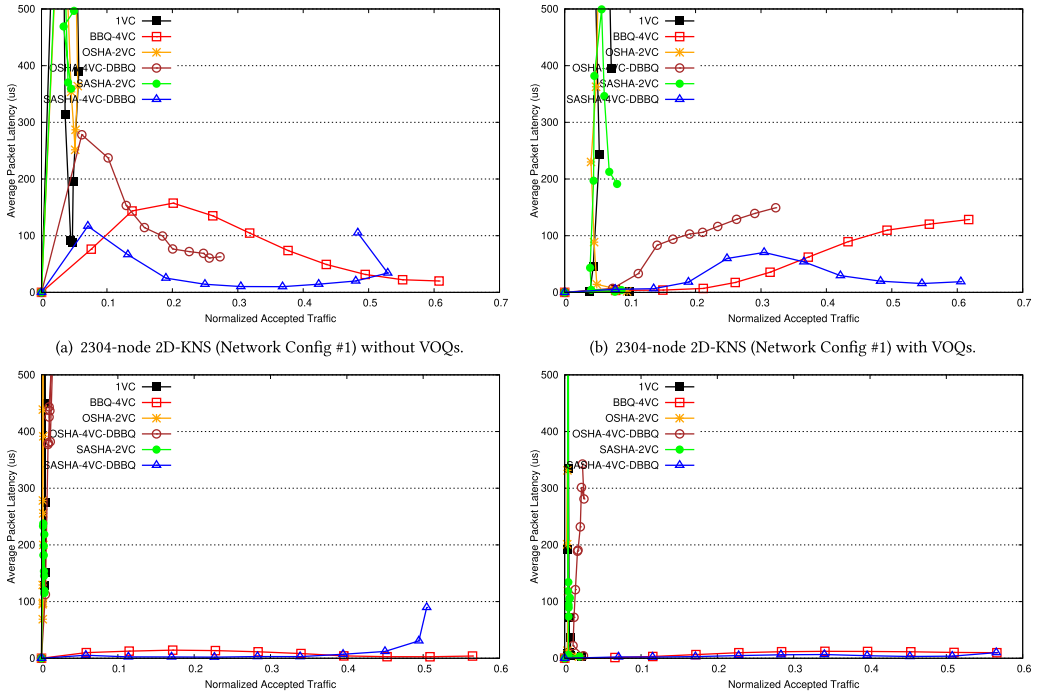
Fig. 8. Average packet latency (microseconds) versus normalized accepted traffic for 13,824-node 3D-Torus topologies (Network Configuration #4 of Table 1) with and without VOQs, when Zipf traffic is generated ($s = 1$ and $s = 3$).

KNS size increases to 13,824 end nodes (Figures 9(c) and 9(d)), the congestion problems augment as well. Again, BBQ-4VC and SASHA-4VC-DBBQ deal with HoL blocking effects in a proper manner when deterministic and source-adaptive routing algorithms are used, respectively.

Figure 10 shows performance results for 2D- and 3D-Tori (network Configurations #3 and #4 of Table 1) when we generate hot-spot traffic. 1VC and OSHA behave like in KNS topologies. However, there is a difference in Tori networks, since OSHA-4VC-DBBQ deals better with HoL blocking, close to the performance of SASHA-4VC-DBBQ. As Tori topologies have longer diameter, they avoid congestion spreading quickly. This effect favors the oblivious routing, as it helps to randomize traffic routes when network traffic load is not too high. Note that the traffic load the network can absorb increases when 3D-Tori are used regardless the network size, since the path diversity also augments with the third dimension. BBQ-4VC achieves the best results in 3D-Tori, due to the benefits of the routing algorithm. SASHA-4VC-DBBQ also achieves good results, close to those of BBQ-4VC and improving OSHA-4VC-DBBQ. As we mentioned before, adaptive and oblivious routing algorithm can be counterproductive when strong congestion scenarios appear [29], since it is dramatic for the network performance to spread congestion trees by the effect of routing algorithms.

5.4 Real-Traffic Traces Results

Figure 11 shows the execution-time results (in milliseconds) for 2D-Tori and KNS topologies (i.e., network Configurations #1 and #3 of Table 1) when the PTRANS trace-based traffic has been generated.



(a) 2304-node 2D-KNS (Network Config #1) without VOQs.

(b) 2304-node 2D-KNS (Network Config #1) with VOQs.

(c) 13824-node 3D-KNS (Network Config #2) without VOQs.

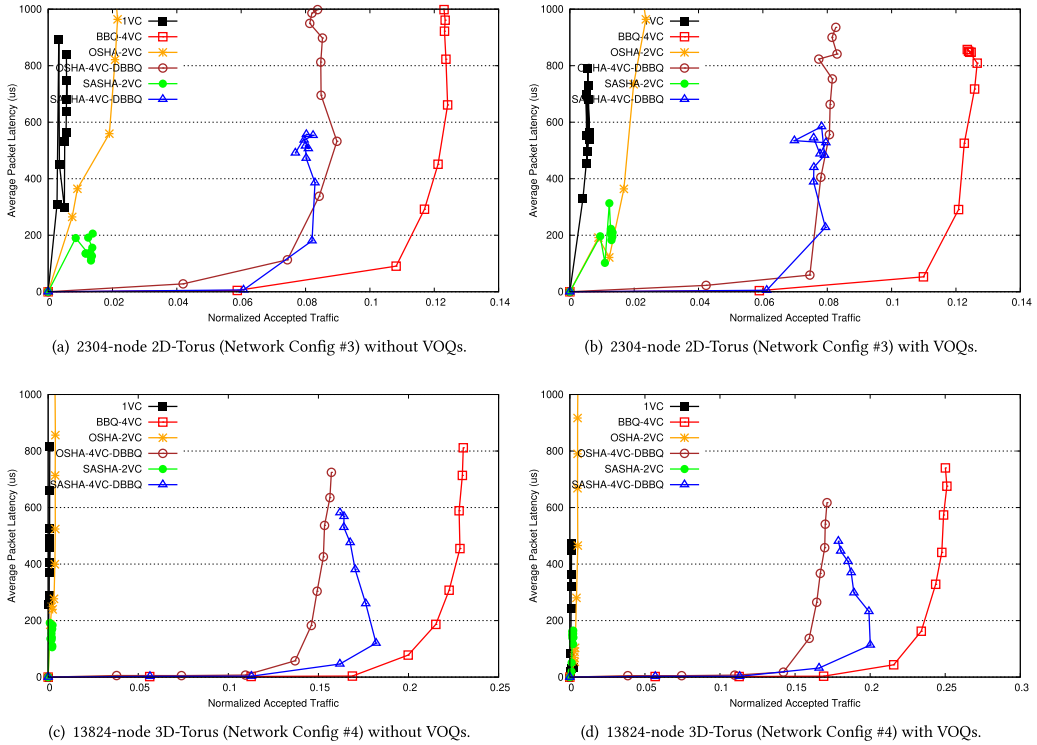
(d) 13824-node 3D-KNS (Network Config #2) with VOQs.

Fig. 9. Average packet latency (microseconds) versus normalized accepted traffic for network Configurations #1 and #2 of Table 1 with and without VOQs, when Hot-spot traffic is generated.

As we have mentioned in Section 5.1, the metric analyzed in this case is the execution time of the application (in milliseconds). Note that the results obtained for all the network configurations are very similar, because the PTRANS generates a moderate load, which does not saturate the KNS topologies. However, we observe a similar trend to that shown in Figures 5 and 9, when a moderate load is generated (i.e., between 30% and 60% of traffic load). By contrast, in 2D-Tori topologies (with and without VOQs) the traffic load generates contention due to the smaller bisection bandwidth. In these scenarios, SASHA improves the execution time significantly. Another important observation is that 2D-KNS topologies obtain around 520ms of execution time, while 2D-Tori networks obtain 1,200ms when SASHA is used.

5.5 Area and Energy Consumption Considerations

In this section, we discuss the area and energy overhead of switch buffers when several VCs are used for HoL blocking reduction. We have used the CACTI tool v7.0 [27] using its SRAM modeling. CACTI is an analytic tool that takes a set of cache/memory parameters as input and calculates its access time, power, cycle time, and area. We assume 128KB SRAM memories (block size is 64 bytes) using a 22nm technology mode, and a link speed equal to 100Gbps (i.e., 12.5GB/s). By means of the CACTI tool, we obtained that the SRAM area of is 0.26mm^2, and the power consumption, which is determined by the total dynamic read energy/access, is equal to 0.09nJ. Note that the switch organization based on VOQs may use stacked buffers connected to several stacked crossbars. We have omitted this study, since the specific details of VOQ-based switches are not available, as far as we know. We think that the area and energy results obtained with CACTI are reasonable, compared with recently published results from the industry [31]. Note that the number VCs used



(a) 2304-node 2D-Torus (Network Config #3) without VOQs.

(b) 2304-node 2D-Torus (Network Config #3) with VOQs.

(c) 13824-node 3D-Torus (Network Config #4) without VOQs.

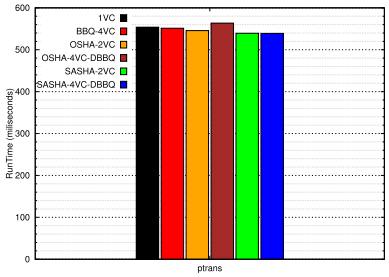
(d) 13824-node 3D-Torus (Network Config #4) with VOQs.

Fig. 10. Average packet latency (microseconds) versus normalized accepted traffic for network Configurations #3 and #4 of Table 1 with and without VOQs, when Hot-Spot traffic is generated.

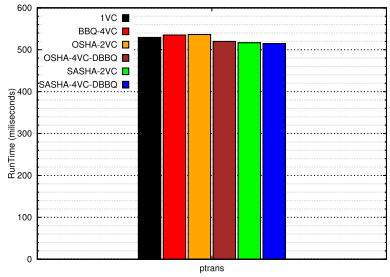
per switch depends on the network hardware features, since the buffer size is fixed, and the VCs required by OSHA and SASHA techniques depends on those offered by the hardware. Finally, note that a thorough study of the power consumption when different traffic patterns are executed in the network is expected as future work.

6 CONCLUSIONS

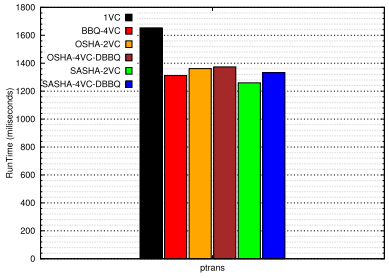
In this article, we have proposed a new approach to deal with HoL blocking in KNS and direct network topologies using oblivious and source-adaptive routing. We configure the network to use two VNs offering independent buffer space to store packets routed through the multiple routes offered by the routing algorithms. The use of two VNs guarantees that no deadlocks appear. Each VN needs to use several queues to separate packets, thus reducing HoL blocking. As our approach is valid for either oblivious or source-adaptive routing, we refer to it as OSHA (*Oblivious Solution for Head-of-Line Blocking Avoidance*) and SASHA (*Source-Adaptive Solution for Head-of-Line Blocking Avoidance*), based on the routing. OSHA and SASHA use a new queuing scheme at each VN, called Dynamic Band-based Queuing (DBBQ), which maps traffic flows to the available queues, so that HoL blocking is reduced. We have evaluated OSHA and SASHA by means of extensive simulation experiments, modeling KNS and direct networks up to 13K end-nodes, under synthetic (Zipf and hot-spot) and trace-based traffic patterns. In the light of the obtained results, we can conclude that OSHA and SASHA deal efficiently with HoL blocking, requiring a reduced set of queues, when we use oblivious or source-adaptive routing in KNS and direct networks, regardless the network



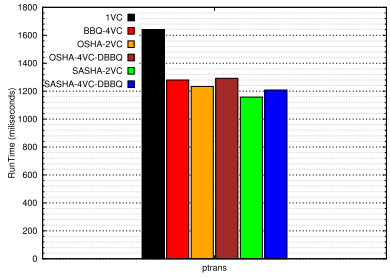
(a) 2304-node 2D-KNS (Network Config #1) without VOQs.



(b) 2304-node 2D-KNS (Network Config #1) with VOQs.



(c) 2304-node 2D-Torus (Network Config #3) without VOQs.



(d) 2304-node 2D-Torus (Network Config #3) with VOQs.

Fig. 11. Execution time for network Configurations #1 and #3 of Table 1 with and without VOQs, when the PTRANS traces are used.

size. The proposed solutions are easy to implement in real products, as they do not introduce extra overhead.

REFERENCES

[1] M. A. Heroux and J. Dongarra. 2013. Toward a New Metric for Ranking High Performance Computing Systems. SAND2013 - 4744.

[2] Y. Ajima, T. Inoue, S. Hiramoto, Y. Takagi, and T. Shimizu. 2012. The Tofu interconnect. *IEEE Micro* 32, 1 (Jan. 2012), 21–31. DOI: https://doi.org/10.1109/MM.2011.98

[3] R. Alverson, D. Roweth, and L. Kaplan. 2010. The Gemini system interconnect. In *Proceedings of the 18th IEEE Symposium on High Performance Interconnects*. 83–87. DOI: https://doi.org/10.1109/HOTI.2010.23

[4] T. Anderson, S. Owicki, J. Saxe, and C. Thacker. 1993. High-speed switch scheduling for local-area networks. *ACM Trans. Comput. Syst.* 11, 4 (Nov. 1993), 319–352.

[5] F. J. Andujar, J. A. Villar, F. J. Alfaro, J. L. Sanchez, and J. Escudero-Sahuquillo. 2016. An open-source family of tools to reproduce MPI-based workloads in interconnection network simulators. *J. Supercomput.* 72, 12 (2016), 4601–4628. DOI: https://doi.org/10.1007/s11227-016-1757-0

[6] M. Besta and T. Hoefler. 2014. Slim fly: A cost effective low-diameter network topology. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 348–359. DOI: https://doi.org/10.1109/SC.2014.34

[7] L. Breslau, Pei Cao, Li Fan, G. Phillips, and S. Shenker. 1999. Web caching and Zipf-like distributions: Evidence and implications. In *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 1. 126–134.

[8] C. Camarero, C. Martínez, E. Vallejo, and R. Beivide. 2017. Projective networks: Topologies for large parallel computer systems. *IEEE Trans. Parallel Distrib. Syst.* 28, 7 (July 2017), 2003–2016. DOI: https://doi.org/10.1109/TPDS.2016.2635640

[9] W. J. Dally. 1992. Virtual-channel flow control. *IEEE Trans. Parallel Distrib. Syst.* 3, 2 (1992), 194–205. DOI: https://doi.org/10.1109/71.127260

[10] W. J. Dally, P. Carvey, and L. Dennison. 1998. Architecture of the Avici terabit switch/router. In *Proceedings of the 6th IEEE Symposium on High-Performance Interconnects (Hot Interconnects'98)*. 41–50.

[11] J. Duato, I. Johnson, J. Flich, F. Naven, P. Garcia, and T. Nachiondo. 2005. A new scalable and cost-effective congestion management strategy for lossless multistage interconnection networks. In *Proceedings of the 11th International Symposium on High-Performance Computer Architecture (HPCA'05)*. 108–119. DOI: https://doi.org/10.1109/HPCA.2005.1

[12] J. Escudero-Sahuquillo, P. J. Garcia, F. J. Quiles, S.-A. Reinemo, T. Skeie, O. Lysne, and J. Duato. 2014. A new proposal to deal with congestion in InfiniBand-based fat-trees. *J. Parallel Distrib. Comput.* 74, 1 (2014), 1802–1819. DOI: https://doi.org/10.1016/j.jpdc.2013.09.002

[13] J. Escudero-Sahuquillo, Pedro J. Garcia, Francisco J. Quiles, Jose Flich, and Jose Duato. 2013. An effective and feasible congestion management technique for high-performance MINs with tag-based distributed routing. *IEEE Trans. Parallel Distrib. Syst.* 24, 10 (2013), 1918–1929. DOI: https://doi.org/10.1109/TPDS.2012.303

[14] J. Escudero-Sahuquillo, E. G. Gran, P. J. Garcia-Garcia, J. Flich, T. Skeie, O. Lysne, F. J. Quiles, and J. Duato. 2015. Efficient and cost-effective hybrid congestion control for HPC interconnection networks. *IEEE Trans. Parallel Distrib. Syst.* 26, 1 (2015), 107–119. DOI: https://doi.org/10.1109/TPDS.2014.2307851

[15] P. J. Garcia, J. Flich, J. Duato, I. Johnson, F. J. Quiles, and F. Naven. 2005. Dynamic evolution of congestion trees: Analysis and impact on switch architecture. In *Proceedings of the International Conference on High Performance Embedded Architectures and Compilers*. 266–285.

[16] C. Gomez, F. Gilabert, M. E. Gomez, P. Lopez, and J. Duato. 2007. Deterministic versus adaptive routing in fat-trees. In *Proceedings of the Communication Architecture for Clusters Workshop (CAC'07) in Conjunction with the IEEE International Parallel & Distributed Processing Symposium (IPDPS'07)*. 235.

[17] E. G. Gran, M. Eimot, S. Reinemo, T. Skeie, O. Lysne, L. P. Huse, and G. Shainer. 2010. First experiences with congestion control in InfiniBand hardware. In *Proceedings of the IEEE International Symposium on Parallel Distributed Processing (IPDPS'10)*. 1–12. DOI: https://doi.org/10.1109/IPDPS.2010.5470419

[18] W. L. Guay, B. Bogdanski, S.-A. Reinemo, O. Lysne, and T. Skeie. 2011. vFtree—A fat-tree routing algorithm using virtual lanes to alleviate congestion. In *Proceedings of the IEEE International Parallel & Distributed Processing Symposium (IPDPS'11)*. 197–208.

[19] M. Gusat, D. Craddock, W. Denzel, T. Engbersen, N. Ni, G. Pfister, W. Rooney, and J. Duato. 2005. Congestion control in InfiniBand networks. In *Proceedings of the 13th Symposium on High Performance Interconnects (HOTI'05)*. 158–159. DOI: https://doi.org/10.1109/CONECT.2005.14

[20] M. Jurczyk and T. Schwederski. 1996. Phenomenon of higher order head-of-line blocking in multistage interconnection networks under nonuniform traffic patterns. *IEICE Trans. Info. Syst.* E79-D, 8 (Aug. 1996), 1124–1129.

[21] M. J. Karol, M. G. Hluchyj, and S. P. Morgan. 1987. Input versus output queuing on a space-division packet switch. *IEEE Trans. Commun.* 35 (1987), 1347–1356.

[22] M. Katevenis, D. Serpanos, and E. Spyridakis. 1998. Credit-flow-controlled ATM for MP interconnection: The ATLAS I single-chip ATM switch. In *Proceedings of the 4th International Symposium on High-Performance Computer Architecture*. 47–56.

[23] J. Kim, W. J. Dally, S. Scott, and D. Abts. 2008. Technology-driven, highly-scalable dragonfly topology. *SIGARCH Comput. Archit. News* 36, 3 (June 2008), 77–88. DOI: https://doi.org/10.1109/ISCA.2008.19

[24] C. E. Leiserson. 1985. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE Trans. Comput.* 34, 10 (Oct. 1985), 892–901. http://dl.acm.org/citation.cfm?id=4492.4495.

[25] A. Martínez, F. J. Alfaro, J. L. Sánchez, F. J. Quiles, and J. Duato. 2007. A new cost-effective technique for QoS support in clusters. *IEEE Trans. Parallel Distrib. Syst.* 18, 12 (2007), 1714–1726.

[26] T. Nachiondo, J. Flich, and J. Duato. 2010. Buffer management strategies to reduce HoL blocking. *IEEE Trans. Parallel Distrib. Syst.* 21, 6 (June 2010), 739–753. DOI: https://doi.org/10.1109/TPDS.2009.63

[27] Ali Shafiee Naveen Muralimanohar and Vaishnav Srinivas. [n.d.]. CACTI v7.0—A Tool to Model Caches/Memories, 3D stacking, and off-chip IO. Retrieved from https://github.com/HewlettPackard/cacti.

[28] R. Peñaranda, C. Gómez Requena, M. E. Gómez, P. López, and J. Duato. 2016. The k-ary n-direct s-indirect family of topologies for large-scale interconnection networks. *J. Supercomput.* 72, 3 (2016), 1035–1062. DOI: https://doi.org/10.1007/s11227-016-1640-z

[29] J. Rocher-Gonzalez, J. Escudero-Sahuquillo, P. J. García, and F. J. Quiles. 2017. On the impact of routing algorithms in the effectiveness of queuing schemes in high-performance interconnection networks. In *Proceedings of the 25th IEEE HOTI*. 65–72. DOI: https://doi.org/10.1109/HOTI.2017.16

[30] T. Schneider, O. Bibartiu, and T. Hoefler. 2016. Ensuring deadlock-freedom in low-diameter infiniband networks. In *Hot Interconnects*. IEEE Computer Society, 1–8. Retrieved from http://dblp.uni-trier.de/db/conf/hoti/hoti2016.html#SchneiderBH16.

[31] Alexander Shpiner and Eitan Zahavi. 2016. Race cars vs. trailer trucks: Switch buffers sizing vs. latency trade-offs in data center networks. In *Proceedings of the 24th IEEE Symposium on High Performance Interconnects (HOTI'16)*. 53–59. DOI: https://doi.org/10.1109/HOTI.2016.021

[32] Y. Tamir and G. L. Frazier. 1992. Dynamically allocated multi-queue buffers for VLSI communication switches. *IEEE Trans. Comput.* 41, 6 (June 1992), 725–737. DOI: https://doi.org/10.1109/12.144624

[33] P. Yebenes, J. Escudero-Sahuquillo, P. J. Garcia, and F. J. Quiles. 2013. Towards modeling interconnection networks of exascale systems with OMNet++. In *Proceedings of the 21st Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP'13)*. 203–207. DOI: https://doi.org/10.1109/PDP.2013.36

[34] P. Yebenes Segura, J. Escudero-Sahuquillo, C. Gomez Requena, P. J. Garcia, F. J. Quiles, and J. Duato. 2013. BBQ: A straightforward queuing scheme to reduce HoL-blocking in high-performance hybrid networks. In *Proceedings of the International Conference on Parallel and Distributed Computing (Euro-Par'13)*, Vol. 8097. 699–712.

[35] Eitan Zahavi, Greg Johnson, Darren J. Kerbyson, and Michael Lang. 2010. Optimized InfiniBand fat-tree routing for shift all-to-all communication patterns. *J. Concurr. Comput. Pract. Exper.* 22, 2 (2010), 217–231.

Received June 2018; revised January 2019; accepted February 2019