XOR-based HoL-blocking Reduction Routing Mechanisms for Direct Networks

Roberto Peñaranda[a,*], Crispín Gómez[a], María Engracia Gómez[a], Pedro López[a]

[a] *Universidad Politécnica de Valencia, Spain*

Abstract

Routing is a key design parameter in the interconnection network of large parallel computers. Routing algorithms are classified into two different categories depending on the number of routing options available for each source-destination pair: deterministic (there is one path available) and adaptive (there are several ones). Adaptive routing has two opposed effects on network performance. On one hand, it provides routing flexibility that may help on avoiding a congested network area, thus improving network performance. On the other hand, it also may increase the Head-of-Line blocking effect due to more destination nodes sharing the port queues. Usually, adaptive routing uses virtual channels to provide routing flexibility and to guarantee deadlock freedom. Deterministic routing is simpler, which implies lower routing delay and it introduces less Head-of-Line blocking effect. In this paper, we propose an adaptive and HoL-blocking reduction routing algorithm for direct topologies that tries to combine the good properties of both worlds: It provides routing flexibility but also reduces the Head-of-Line blocking effect. To do that, this paper proposes several functions which use the XOR operation to efficiently distribute the packets among virtual channels based on their destination node. The resulting routing mechanisms have different properties depending on whether they enforce routing flexibility or Head-of-Line blocking reduction.

*Corresponding author
Email address: ropeaceb@gap.upv.es (Roberto Peñaranda)

Keywords: direct topology, adaptive routing, deterministic routing, Head of Line blocking, routing algorithms

1. Introduction

A key component in the performance of large parallel computers is the interconnection network. Performance of these systems is increasingly determined by how data is communicated among the huge number of computing resources. Latency and throughput are the key performance metrics of interconnection networks [1, 2]. Latency is the elapsed time between message injection into the network and its arrival at its destination, and it is the sum of two components, one related to the time required to traverse the network in absence of traffic (base latency) and the other one related to the delay suffered by messages due to contention. If minimal routing is used, as commonly done, then the base latency is constant for each source-destination pair as the number of hops does not change. The second component of latency depends on network contention. Throughput refers to the maximum amount of data the network can deliver per time unit. The main goal is to minimize message latency while maximizing network throughput. To achieve this goal, we have to consider, among others, two main parameters [1, 2]: topology and routing. The topology provides the connection pattern among the nodes. This paper focuses on direct topologies, which are one of the options used to build large parallel machines. In fact, several machines that have occupied the topmost positions of the Top500 list of supercomputers [3] are based on direct topologies, like the ones that occupy the 3rd, 4th and 5th positions in the June 2016 list.

The routing algorithm decides the paths followed by messages through the network. A routing algorithm can be either deterministic or adaptive. In deterministic routing, an injected packet traverses a unique, predetermined path between each source-destination pair. Opposite to this, adaptive routing schemes allow several paths for each source-destination pair. This, on the one hand, helps avoiding congested network areas by allowing packets to take alternative

paths to reach their destination. However, this flexibility has a negative impact on packet contention because it may increase the Head-of-Line (HoL) blocking effect. This effect occurs when a packet at the head of a queue blocks, and prevents the rest of packets in that queue from advancing, even if they could do so because the required resources are free. The HoL-blocking effect may be highly pernicious and may limit the throughput of the switch up to about 58% of its peak value [4, 5, 6]. In order to reduce the HoL-blocking effect, it is very important to isolate as much as possible those packets destined to different nodes [7, 8]. However, adaptive routing tends to spread packet destinations all over the network, which may have a very negative effect when a destination is saturated since it will spread the congestion to other network areas and prevent more packets to arrive to other non-saturated destinations.

Adaptive and deterministic routing algorithms have different properties. While adaptive routing algorithms outperform deterministic ones [1] for some traffic patterns because of their routing flexibility, thus improving network throughput and reducing message latency deterministic routing better isolates destinations reducing the HoL blocking effect, which enables deterministic routing to outperform adaptive routing for some other traffic patterns such as traffic with hot-spot destinations. Moreover, adaptive routing usually leads to a more complex implementation and it is more deadlock-prone [9, 10]. Adaptive routing usually relies on the use of virtual channels (VCs) [11] to avoid deadlocks. On the other hand, adaptive routing requires a selection function to choose the path that will be finally used, as several paths are available for each packet. As a consequence, routing delay for adaptive routing is usually higher compared to deterministic routing [12, 13, 14].

In this paper we focus on combining the good properties of adaptive (routing flexibility) and deterministic routing (reduced HoL blocking effect) to design a hybrid routing algorithm. The idea behind this routing algorithm is to take advantage of virtual channels, usually used in adaptive routing to provide flexibility, but with a revisited aim: confining destinations in subsets of virtual channels in order to reduce the HoL-blocking effect while providing some degree

of flexibility. In order to select the VCs that can be used by a given packet, the proposed routing algorithm uses a XOR function of the destination identifier which provides a balanced usage of VCs for all traffic patterns. A deterministic version of the proposed routing algorithm was presented in [15] and a first version of the adaptive routing algorithm based on the XOR function was published in [16]. The current paper unifies both proposals under a common framework, explaining in more depth how the use of the XOR operation helps reducing the Head-of-Line blocking effect both for deterministic and adaptive routing. This paper also include new performance evaluation results.

The rest of the paper is organized as follows. Section 2 introduces some background on routing in direct topologies. In Section 3, we present some previous deterministic routing algorithms that use virtual channels to reduce the HoL-blocking effect. In Section 4.1, we present the XOR-based HoL-blocking reduction deterministic routing algorithm. And, in Section 4.2, we extend the proposal of Section 4.1 to propose the HoL-blocking reduction adaptive routing algorithm that is able to combine the benefits of deterministic and adaptive routing algorithms. These algorithms are evaluated in Section 5. Finally, some conclusions are drawn.

2. Direct Topologies

A direct network consists of a set of nodes, each one being directly connected to a subset of other nodes in the network. The most popular direct topologies organize nodes in an orthogonal n–dimensional space. The regularity of these networks greatly simplifies their deployment and routing algorithm implementation. The movement of a packet in a dimension does not modify the number of remaining hops in the other dimensions to reach the packet destination. The most commonly–used direct topologies are the mesh, the torus, and the hypercube. These topologies have been used in several of the most powerful supercomputers (see the Top500 list [3]).

The distance between source and destination nodes is computed as the sum

of the offsets in each dimension. Minimal routing algorithms will reduce one of those offsets at each routing step. The simplest minimal routing algorithm, known as dimension-order routing (DOR) [1], consists of reducing an offset to zero before considering the offset in the next dimension. For n-dimensional meshes, to enforce deadlock-freedom, DOR routes packets by crossing dimensions in strictly increasing (or decreasing) order.

However, in tori, crossing network dimensions in order is not enough to obtain a deadlock-free routing algorithm as the channel dependency graph is cyclic [1]. Cycles are broken by splitting each physical channel into two virtual channels (VCs) [17]. More than two VCs may be used for performance improvement purposes [11]. Another technique used to avoid deadlocks in tori with deterministic routing is the bubble flow control mechanism [18]. This mechanism avoids deadlocks in each ring of the torus by ensuring that there is always an empty buffer that allows packets to advance along the ring. This technique does not rely on virtual channels.

Many adaptive routing algorithms have been published in the literature [19, 20, 21, 22]. Fully adaptive routing [23, 1] in meshes and tori allows packets to reduce dimension offsets following in any order. Therefore, all the minimal paths between each source-destination pair can be used by packets. However, this may introduce cycles and deadlock-freedom has to be ensured with additional mechanisms. According to [23], VCs may be used to cross network dimensions in any order if deadlock freedom is guaranteed by providing an *escape path* to packets. This escape path is provided by means of a deadlock-free routing algorithm (for instance, DOR) in another set of VCs. Notice that with the bubble flow control mechanism, only one VC is required for escape path implementation in tori and meshes, and the remaining VCs can be used for adaptive routing.

In the routing algorithtms analyzed in this paper we will assume the bubble flow control mechanism to break cycles in torus.

3. Related Work

As mentioned in the previous section, adaptive routing provides flexibility in the path followed by packets and in the use of VCs since it uses VCs with complete freedom, except the ones used as escape channels. This routing freedom has two opposite effects over performance. The positive one is that temporally congested network areas can be avoided and therefore, for some traffic patterns, packets can make a better use of the network resources. However, the negative effect is that packets with different destination nodes may be highly interleaved in the switch queues, which significantly increases the HoL-blocking effect with hot-spot traffic patterns, leading to degrade network performance for all the network (as we can see in Section 5). On the other hand, deterministic routing does not provide that flexibility, which may negatively affect performance for some traffic patterns, but its contribution to the HoL-blocking effect is lower.

The idea of reducing the HoL-blocking effect by using VCs has been pursued before by previously proposed deterministic routing algorithms. The key idea behind these proposals is to classify destinations into VCs, according to some criteria. Virtual Output Queueing at network level (VOQnet) [24] needs as many VCs as nodes in the network and associates each destination to a different VC. VOQnet completely removes the HoL-blocking effect from the network, but the required number of VCs is unaffordable even in small networks since it grows linearly with the network size. However, it is often used for comparison purposes since it provides an upper bound that could be achieved by completely removing HoL-blocking from a network. Another option is Virtual Output Queueing at switch level (VOQsw) [25], which requires as many VCs as switch output ports, and associates the set of reachable destinations through a given output port to the same VC. Therefore, VCs are selected according to the next output port the packet will use. VOQsw leads to a worse classification of packets than the one obtained with VOQnet and it is also not scalable, as the number of required VCs depends on switch degree.

Destination-Based Buffer Management (DBBM) was introduced in [26], as

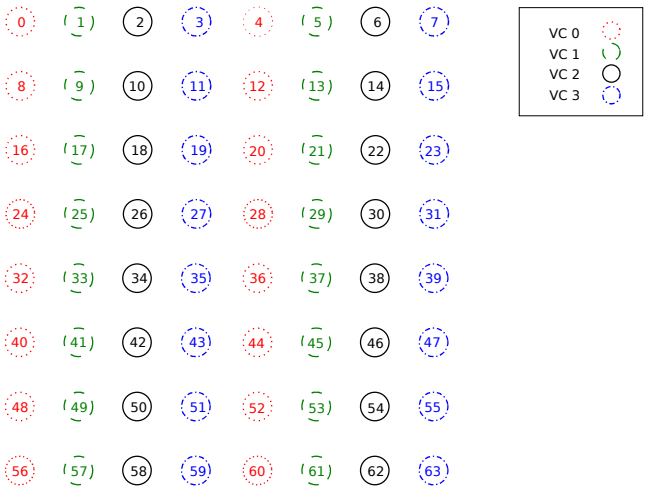


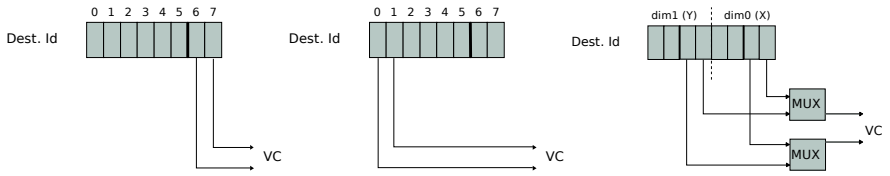
Figure 1: How DBBM assigns destinations to VCs in a 8×8 mesh with 4 VCs.

Dim	VC#0	VC#1	VC#2	VC#3
X	8	16	16	16
Y	7	No dest.	No dest.	No dest.

Table 1: How many destinations DBBM assigns to node 0 VCs in a 8×8 mesh with 4 VCs.

an attempt to obtain a scalable version of VOQnet. This mechanism selects VCs by using the destination node identifier modulo the number of VCs. While it works for other topologies, when using DBBM in a 2D mesh or torus, all the nodes in a given column are assigned to the same VC, as shown in Figure 1 for an 8x8 mesh with 4 VCs per physical channel. Indeed, Table 1 shows the number of destinations assigned to different VCs for a node of the network (node 0). For instance, VC#0 of the X-dimension is used to reach 8 nodes (the 4^{th} row), while VC#1 of the X-dimension is used to reach 16 nodes (the 1^{st} and 5^{th} row). As it can be seen, all the VCs in the Y-dimension are never used but one per port. This lack of classification in the last dimension (Y) lead to congestion due to the HoL-blocking effect that, at the end, could be propagated to the whole network due to upstream flow control pressure.

If we analyze the implementation complexity of the VC selection mechanism,



(a) DBBM (b) BBQ (c) IODET

Figure 2: Implementation of VC selection for a 256-node 2D network and 4 VCs.

DBBM is very simple, provided that the number of VCs is a power of two. This mechanism uses the modulo operation by the #VCs and its implementation is as easy as selecting the $\log(\#VCs)$ least significant bits of the packet destination (see Figure 2(a)). Additionally, notice that, as VC assignment depends only on the packet destination, packets use the same VC while it traverses the network. This is a nice feature, as VC assignment can be done once at the source node, and the rest of nodes that a packet crosses across the network merely forwards the packet through the same VC from which the packet arrived to, like in virtual networks [27], without requiring VC transitioning [17]. Furthermore, this fact also leads to a reduced switch complexity. As there is not need to move packets from one input VC to another output VC inside switches, the internal switch of the nodes can be implemented as one independent switch per VC, instead of deploying a fully-connected crossbar. We will further analyze switch complexity later considering all these aspects.

Band-Based Queuing (BBQ) mechanism [28] was proposed in order to overcome the bad classification of packets in the last dimension of DBBM. BBQ also uses some bits of the destination identifier to choose the VC for each packet, but opposite to DBBM, BBQ uses the destination $\log(\#VCs)$ most significant bits (see Figure 2(b)). That is, BBQ divides the network in as many horizontal bands as VCs, in such a way that the nodes in each column are distributed as much as possible among the VCs. However, the problem is that all the nodes inside each horizontal band use the same VC, and, therefore they may suffer from HoL-blocking in the first dimension. As in DBBM, BBQ never changes

Dim	VC#0	VC#1	VC#2	VC#3
X	8	16	16	16
Y	1	2	2	2

Table 2: How IODET assigns destinations to node 0 VCs in a 8×8 mesh. #VCs is 4

the VC of a packet during its path in the network. It can be assigned once at injection time keeping the same VC along its path in the network.

In-Order DETerministic routing (IODET) [29] follows a different approach and it selects the VC by considering not the whole destination identifier but the component of the packet destination corresponding to the dimension in which the packet is being routed. The VC to be used by a packet is obtained by performing the modulo operation of the dimension coordinates of the destination. That is, given a packet destined to node $\{p_{n-1}, ..., p_1, p_0\}$, when routed in dimension d it will use the VC given by $p_d \bmod \#VCs$. This mechanism does a better job classifying packets than DBBM as can be seen in Table 2, which shows the number of destinations assigned to different VCs for node 0 for an 8x8 mesh with 4 VCs. As it can be seen, all the VCs in both dimensions are used when applying IODET to distribute destination among VCs.

Considering the implementation complexity of the VC selection, IODET is also very simple, as displayed in Figure 2(c) which shows an example for 4 VCs. As it can be seen, the least significant bits of the component for each dimension is used to select the VC. However, as the assignment of destinations to VCs depends on the dimension the packet is traversing, the VC is changed when there is a dimension change and, therefore, in those nodes the new VC to use must be computed. As a consequence, the node internal switch must allow the change in the VC assignment and the switch implementation is not as easy as the DBBM one. We will analyze this issue later.

4. XOR-based HoL-blocking Reduction Routing

In this section, we present a mechanism to assign destinations to VCs. In order to obtain the VC, the mechanism applies a function to the destination identifier A function that considers all the components of the identifier should be used. In particular, the mechanism uses of bitwise XOR function. Bitwise XOR has extensively been used for hashing, reducing conflict misses in caches [30] or in branch predictors [31]. We apply the XOR-based destination distribution to two different routing algorithms: first a deterministic routing mechanism (XORDET) [15] is designed, and then an adaptive version (XORADAP) [16] is proposed. The idea of this second algorithm is to combine the good properties of both, deterministic and adaptive routing, to adapt to any traffic pattern to obtain optimized performance results.

This XOR function, opposite to the previously presented deterministic algorithms that use a subset of the node destination bits, distributes destinations among VCs by performing a bitwise XOR operation to all the bits of the destination node, as follows. Assume that there are o assignment options available, that can be virtual channels or groups of virtual channels (as we will see later). Then, $l = \log_2 o$ bits are required to denote an assignment option. If destination identifiers are n bits long, then, for each destination, the assignment option to use is obtained by performing l XOR operations in parallel. In each XOR operation $\frac{n}{l}$ bits of the destination are XORed, taking them in an interleaved fashion. In particular, given a packet destined to node $\{p_{n-1}, ..., p_1, p_0\}$, it will use the assignment option given by the bits $\{VC_{l-1}, ..., VC_1, VC_0\}$, computed as follows:

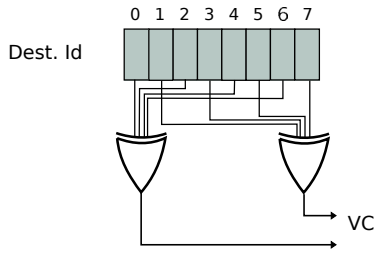
$$VC_0 = p_0 \oplus p_{0+l} \oplus p_{0+2l} \ldots \oplus p_{0+(\frac{n}{l}-1)l}$$

$$VC_1 = p_1 \oplus p_{1+l} \oplus p_{1+2l} \ldots \oplus p_{1+(\frac{n}{l}-1)l}$$

$$VC_{l-1} = p_{l-1} \oplus p_{l-1+l} \oplus p_{l-1+2l} \ldots \oplus p_{l-1+(\frac{n}{l}-1)l}$$

Notice that the number of options (VCs or groups of VCs) must be a power of two.

As we can see, the assignment only depends on the destination identifier.



(a) XORDET

Figure 3: Implementation of VC selection in XORDET for a 256-node 2D network and 4 VCs.

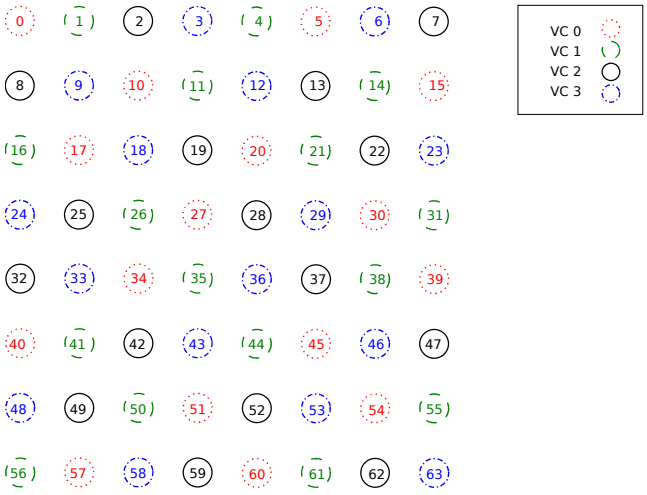


Figure 4: How XORDET assigns destinations to VCs in a 8×8 mesh with 4 VCs.

Indeed, as it does not depend on the particular topology, the XOR function should work well to distribute destinations on any topology. In this paper, we have only focused on direct topologies and specifically on Torus topologies.

4.1. XORDET: XOR DETerministic Routing

XORDET is a deterministic routing algorithm that reduces the HoL-blocking effect and performs a balanced distribution of destinations among VCs. For doing that, this algorithm uses the XOR function that was described above. In this case, we use this function to obtain the virtual channel to use.

Dim	VC#0	VC#1	VC#2	VC#3
X	14	14	14	14
Y	1	2	2	2

Table 3: How many destinations XORDET assigns to node 0 VCs in a 8×8 mesh with 4 VCs.

Figure 3 shows how the VC selection will be implemented in a network with 4 VCs and 8-bit node identifiers. Each VC bit is obtained by XORing 4 bits of the node destination identifier in an interleaved fashion. Notice that XORDET implementation of VC selection is very simple, as only some XOR gates are required per source node. In particular, for v VCs, $l = \log v$ XOR gates are required. Each one of them will have $\binom{n}{l}$ inputs, n being the number of bits of the destination identifier. If n is not divisible by l, some gates will have an extra input. Notice that this implementation assumes that the number of VCs is a power of two. On the other hand, we would like to highlight that, as in DBBM, the assignment of destinations to VCs does not change as packet travels through the network. Therefore, this assignment can be performed only once when the packet is injected into the network. As a consequence, the network could be considered as several virtual independent networks, without interconnection among them, and the internal node switch can be implemented as several independent switches. As a consequence, the implementation of XORDET is very simple (as in DBBM) but, as will we shown in Section 5, it also allows the VCs to maximize its utilization (as in IODET).

XORDET is able to isolate traffic destined to different nodes and also balancing destinations among VCs. Figure 4 shows how destinations are distributed among VCs in a network with 64 nodes and 4 VCs. As it can be seen, XORDET does a very good job, as traffic destined to either rows or columns will be distributed among the VCs.

Table 3 shows how many destinations are assigned to each VC of node 0 of the network.

As shown, XORDET balances node destinations among VCs which will bal-

ance traffic for uniform random traffic pattern. But XORDET is a deterministic routing algorithm and this means that, for some adversarial traffic patterns, it may suffer from performance drops due to the limited routing flexibility. While XORDET works very well to avoid congestion caused by hot-spots, for some other traffic patterns, adaptive routing is able to outperform deterministic routing in general and, in particular, XORDET. For this reason, in this paper we propose an adaptive HoL-blocking reduction routing algorithm that is able to combine the routing flexibility provided by adaptive routing with the destination isolation provided by HoL-blocking reduction routing to obtain optimized performance results for all traffic patterns.

4.2. XORADAP: XOR ADAPtive Routing

As mentioned above, deterministic routing lacks flexibility to adapt to some adversarial traffic patterns while adaptive routing does not encourage HoL-blocking effect reduction, which is very important for some traffic patterns. In particular, with a hot-spot node in the network, a HoL-blocking reduction deterministic algorithm that isolates the hot-spot traffic works better than adaptive routing [15] that spreads that traffic over the network avoiding other traffic to progress in the network. Let us analyze what happens in this case.

With adaptive routing, the problem arises in the VCs of all the network dimensions that provide the routing flexibility (i.e. the ones that can be used to cross the network dimensions without following any order). When there is a hot-spot node, adaptive routing trends to distribute traffic among all the available VCs, filling all the buffers with packets destined to the hot-spot node. Those packets will interfere with other traffic flows all over the network, thus creating the HoL-blocking problem.

On the contrary, HoL-blocking reduction algorithms like IODET or XORDET have a very good behavior with hot-spot traffic because they confine the hot-spot traffic in just one of the VCs, allowing the rest of traffic to progress normally across other VCs. These routing algorithms also work well with uniform random traffic pattern as it will be shown, but they obtain a poor performance for some

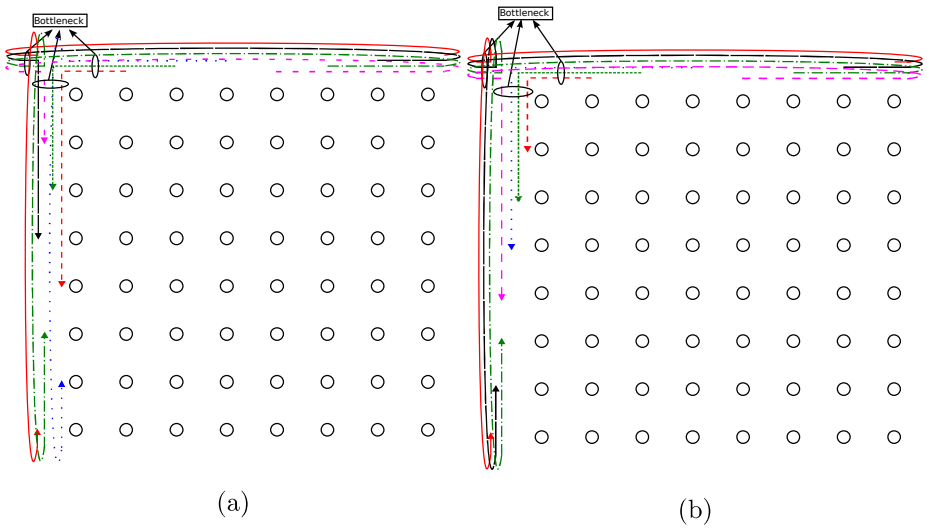


Figure 5: Paths of source-destination pairs of the first row with different pattern traffics: (a) Bit-reversal and (b) Matrix Transpose.

adversarial traffic patterns. For instance, consider the bit-reversal or matrix transpose traffic patterns [1]. In these cases, if deterministic routing is used, a lot of source-destination pairs will use the same links due to the destination distribution leaving many links unused. This fact creates a bottleneck since many messages have to cross the same link. We can see this behavior in Figure 5. This figure shows the paths used by the source nodes belonging to the first row in a 2D torus for the bit-reversal and matrix transpose traffic patterns using a deterministic routing algorithm. In particular, DOR was used. As it can be observed in the figure, the links of the topmost leftmost node become a bottleneck with deterministic routing. For these kinds of traffic patterns, adaptive routing can take advantage of all the network resources, providing a better utilization of the network links and therefore, improving overall network performance for this adversarial traffic pattens.

In order to provide flexibility for adversarial traffic patterns and also reduce the negative effects of HoL-blocking, we propose an adaptive HoL blocking reduction routing algorithm. In this routing algorithm, VCs are organized as in a

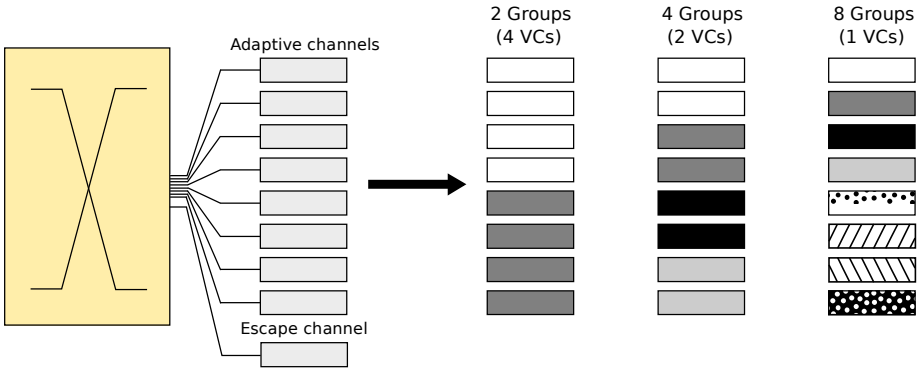


Figure 6: How XORADAP may assign VCs to groups with 8 VCs.

fully adaptive routing algorithm: there is a *group* of adaptive VCs, that can be used to cross the network dimensions in any order, and also there is an escape channel. We assume that bubble flow control is used in the escape channel. However, this routing algorithm confines each node destination identifier in a subset of the adaptive VCs instead of allowing the use of any of them. Contrary to deterministic routing, the routing algorithm allows crossing the network dimensions following any order (and therefore allowing more flexibility) but restricting the use of VCs depending on the destination node and thus confining the congested destinations in some VCs and allowing the packets located in the rest of VCs to progress. As a consequence, it provides some degree of flexibility but, at the same time, it limits the impact of the HoL-blocking effect because only a subset of the VCs can be used for a given destination.

To assign destinations to VCs, any mechanism could be used. In this paper, we propose to use the bitwise XOR function. For this reason, the resulting routing algorithm will be referred to as XORADAP (XOR ADAPtive). The VC assignment works as follows. We split the adaptive VCs into several groups. Each group can be composed of 1, 2 or more VCs. Given a packet, it will be forwarded to one of those groups depending on the packet destination, and any of the VCs of that group could be used.

As mentioned above, we use the same function to classify destinations as

XORDET, but, in this case, we select a group of VCs for each destination instead of just a single VC to classify traffic. In particular, with g groups of VCs, $l = \log g$ XOR gates are required. Notice that the number of groups of VCs must be a power of two.

As stated above, each group is composed of one or more VCs. Several configurations can be used. If there are V_a VCs available for adaptive routing, each group may contain from 1 to V_a virtual channels. Notice that, if we use only one group with all the virtual channels, we obtain the generic fully adaptive algorithm. On the contrary, if each group has only one VC, we obtain an adaptive version of XORDET, that allows packets to cross dimensions following any order. Figure 6 shows an example of the different configurations that can be set for 9 VCs, one escape VC and 8 adaptive VCs. In this case, three configurations are possible for XORADAP: 2, 4 and 8 groups (with 4, 2 and 1 VC per group, respectively). In addition to those configurations, the one using only one group of VCs is also possible, which leads to the fully adaptive routing algorithm, as stated above. As it can be seen, the resulting network is a set of different virtual networks, each one with several VCs. This means that packets of the different virtual networks are not mixed together, effectively separating flows. The escape channel can be used by all the groups of virtual networks.

4.3. Implementation issues

As stated above, the different routing algorithms analyzed in this paper demand different implementation complexity in the internal switch of the nodes. A fully demultiplexed crossbar [11] provides the highest flexibility, allowing connections among all input VCs to all the output VCs (i.e. it can map any input VC onto any output VC). In fact, such a switch is required for adaptive routing, where any input port can forward packets to any output port. However, in the case of deterministic routing, some of the connections provided by the internal switch are unused due to routing restrictions. For instance, if DOR deterministic routing is used, a packet can only use those ports that connect to the same or higher dimensions than the one it arrived. Therefore, the switches could be sim-

plified if routing restrictions are considered, most important, without affecting performance.

Let us consider the routing algorithms proposed in this paper. In XORDET, as the VC is selected as a function of the destination node, packets do not change the VC while they travel across the network, thus leading to a even simpler internal switch design than the traditional deterministic routing with the same number of VCs. Indeed, in XORDET, VC assignment can be done once at the source node, and the rest of nodes that a packet crosses across the network merely forwards the packet through the same VC from which the packet arrived to. Traditional deterministic routing with multiple VCs would have to select the output VC to forward the packet. As a consequence, as there is no need to move packets in a switch from one input VC to another output VC, the internal switch of the nodes can be implemented as one independent switch per VC (i.e. several virtual networks), instead of deploying a fully-connected crossbar, which is cheaper and faster, as switch delay depends on the number of switch ports [12, 13, 14].

In the same way, XORADAP also simplifies switch implementation. In this case, packets may change the VC used but they do not change the assigned group of VCs. The internal switch of the nodes can be implemented as one independent switch per group of VCs. Therefore, we could use a simpler internal switch design than fully adaptive routing. Notice that, for a configuration of one group of all of the adaptive VCs, the complexity will be the same as fully adaptive routing.

We will analyze in more depth switch complexity for different routing algorithms in Section 5.2.

Concerning routing mechanics, deterministic routing only requires applying the routing function [1] while adaptive routing requires the use of both the routing and the selection function [1]. In any case, both the output port and the VC to be used will be returned by the routing algorithm. For both XORDET and XORADAP, a few XOR logic gates are required at the source nodes to compute the corresponding VC or group of VCs, respectively. In XORADAP,

a selection function is also required to select the VC inside the assigned group. However, the number of routing choices is smaller than with fully adaptive routing. As routing delay depends on the number of routing choices [12, 13, 14], XORADAP may lead to a faster implementation than fully adaptive routing.

5. Experimental Evaluation

In this section, we evaluate the HoL-blocking reduction routing algorithms described in this paper (XORADAP and XORDET) by simulation, comparing them with previously proposed ones. We have used a simulation environment developed at our research group. A prior version of this tool was used to provide evaluation results in [1]. First, we will compare XORDET with other HoL-blocking reduction deterministic algorithms like IODET, DBBM, BBQ, VOQnet and VOQsw. We will also consider a fully adaptive routing algorithm and a DOR deterministic routing which allows packets to use all the VCs of the selected dimension, that is a deterministic routing algorithm without destination node classification. Notice that this latter algorithm is actually partially adaptive (as it allows several routing options) and does not guarantee in-order delivery of packets. For this reason, we will refer to it as Out of Order DETerministic routing (OODET). To guarantee deadlock-freedom in tori, the bubble flow control mechanism [18] was used (either in all the VCs for deterministic routing or in the escape VC for fully adaptive routing). Therefore, when a packet is going to be injected to a new ring of the torus topology, either because it is a newly injected packet or it is going to change to another network dimension, the routing algorithm will allow to use the next output port if there is enough buffer space to store more than one packet, providing a bubble to avoid deadlocks.

Regarding adaptive and escape channels, the algorithm firstly provides the adaptive channels that are available to use. If they cannot be used because they are full due to congestion or deadlocks in the adaptive channels, the algorithm provides the escape channel.

After the evaluation of XORDET, we will evaluate XORADAP to analyze how it behaves under different traffic patterns and we will show how a hybrid approach is able to combine the best of two worlds and obtain good performance results for any traffic pattern.

Regarding the number of VCs per physical channel, it must be a power of two in XORDET. In XORADAP, the number of groups of VCs must be also a power of two and an escape channel is required. To perform a fair comparison, for traditional fully adaptive routing we will use the same number of VCs as the one used in XORADAP. Each node has a switch based on a full crossbar with 4-packet queues both at their input and output ports. Packet length is 16 flits. We assume a pipelined router with a latency of 4 clock cycles, and switch and link bandwidth is assumed to be one flit per clock cycle. To avoid HoL-blocking at injection that will interfere in the obtained results, source nodes implement VOQnet. This means that messages with different destinations do not harm the injection of each other. We have modeled different network sizes with different number of dimensions: 64 and 256 nodes with 2 dimensions and 512 nodes with 3 dimensions.

Regarding network traffic, we have considered several widely-used synthetic traffic patterns [1]: uniform random, matrix transpose, and bit-reversal. In addition, as we are interested in analyzing the impact of the HoL-blocking effect, we also evaluated a hot-spot traffic pattern, whose parameters will be described in detail later.

5.1. Performance analysis

5.1.1. XORDET evaluation

First, we will analyze the behavior of XORDET versus the other HoL-blocking reduction deterministic routing algorithms. Figure 7 shows the obtained results for a 2D torus with 256 nodes and uniform random traffic pattern. With only a few number of VCs (4 or 8), any HoL-blocking reduction algorithm is able to reach nearly the same performance as VOQnet, which is the upper bound. The exception is DBBM that, due to its poor destination classification

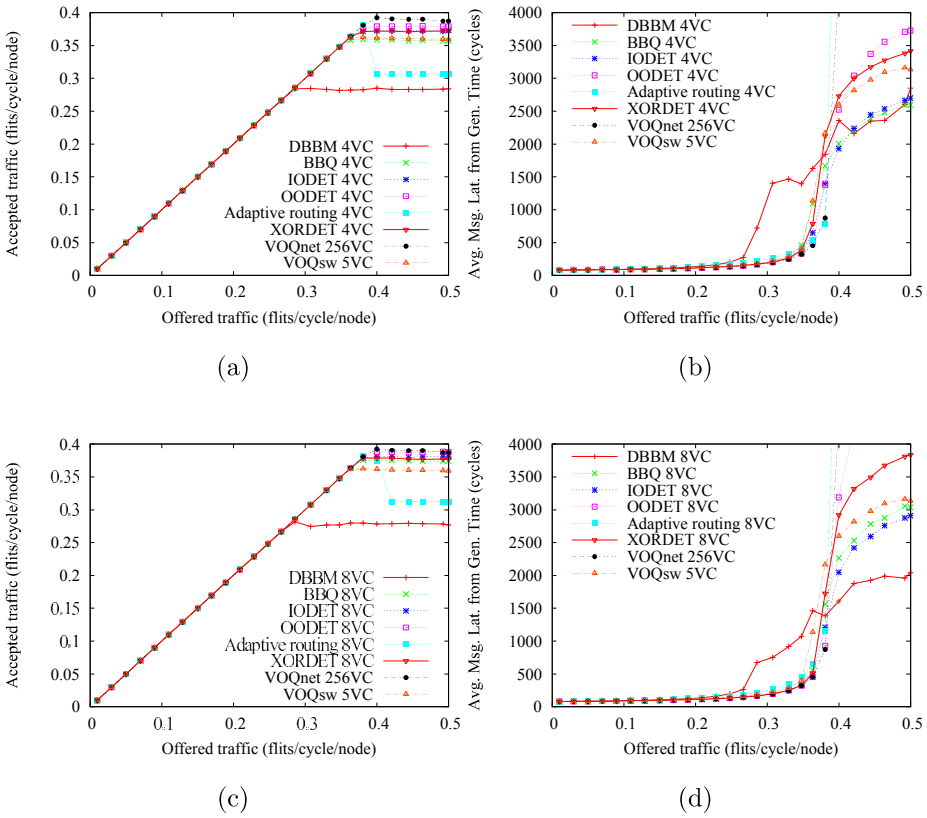


Figure 7: Average packet latency and accepted traffic vs offered load. 256-node 2D-torus. Uniform random traffic pattern. (a,b) 4 VCs and (c,d) 8 VCs.

in the last dimension (see Section 3), it obtains a worse performance. Notice the importance of the dimension ordering followed by the routing algorithm. Unexpectedly, BBQ, which also considers a subset of the node destination identifier bits to classify packets, works quite well. The difference between DBBM and BBQ is that the former consider the least significant bits while the latter considers the most significant ones. As XORDET considers all the node destination bits to classify packets, it should not be affected by changes in the dimension ordering followed by the routing algorithm. On the other hand, fully adaptive routing suffers the typical performance rollback after network saturation [32]. Figure 8 shows the results for a 3D torus with 512 nodes and uniform random

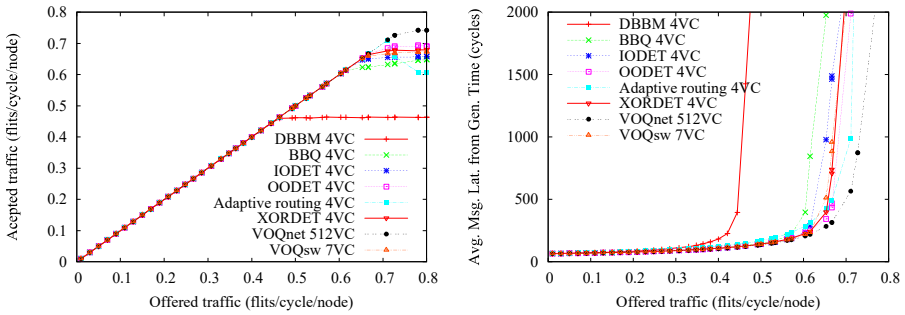


Figure 8: Average packet latency and accepted traffic vs offered load. 512-node 3D-torus. Uniform random traffic pattern. 4VCs.

traffic. As it can be seen, results are qualitatively the same.

In Figure 7, the traditional DOR routing following XY order was used. However, using other deterministic routing algorithms could be interesting. For instance, in [33], X+Y+Z+X-Y-Z- direction-order routing was proposed instead of dimension order routing for fault tolerance purposes. Direction order routing allows routing packets through non-minimal paths by routing in both directions of a dimension and, therefore, offers greater flexibility to avoid faults. For 2D networks, the X+Y+X-Y- direction order routing counterpart would be used. Packets are routed following an ascending dimension order, but taking first the positive dimension directions, and then the negative ones. Figure 9 shows evaluation results for the different HOL-blocking reduction mechanisms but using X+Y+X-Y- with minimal paths as the baseline deterministic routing. We can see how the fact of traversing dimensions in a different order (depending on the directions) changes the behavior of some algorithms like BBQ, which drives down its performance significantly. This effect is similar to the one produced by DBBM before and is due to the fact of using a subset of the destination node identifier bits to select the VC to use. The dimension ordering followed by the routing algorithm may generate an unfair use of the VCs, overloading some of them while others are barely used. However, XORDET or IODET, which

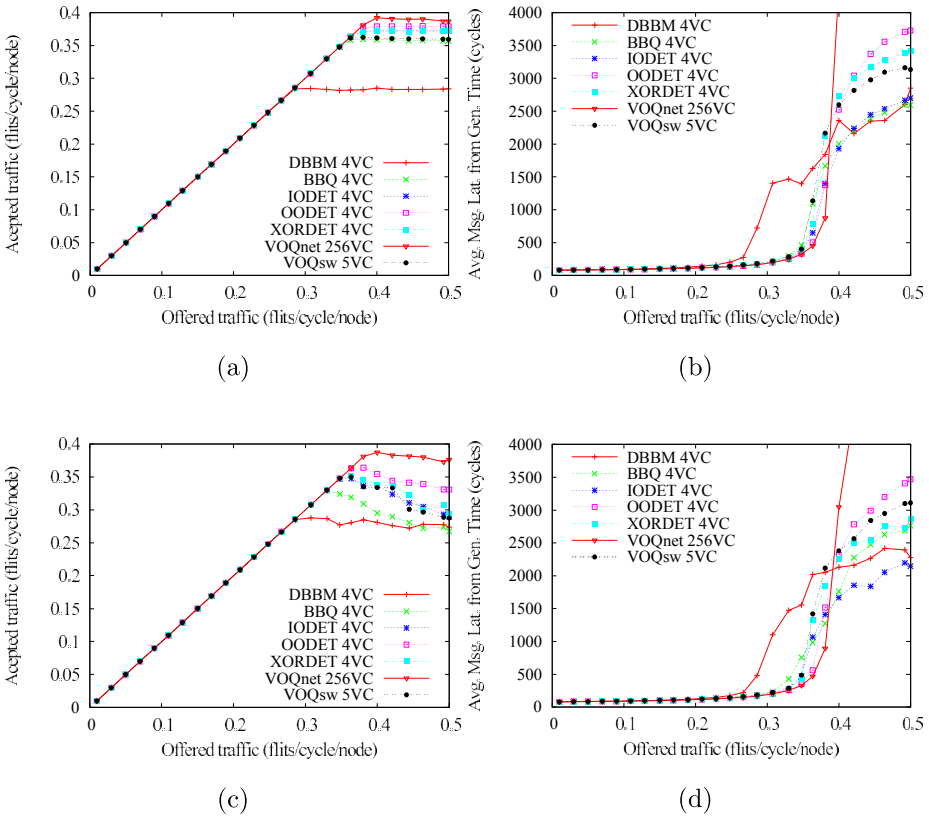
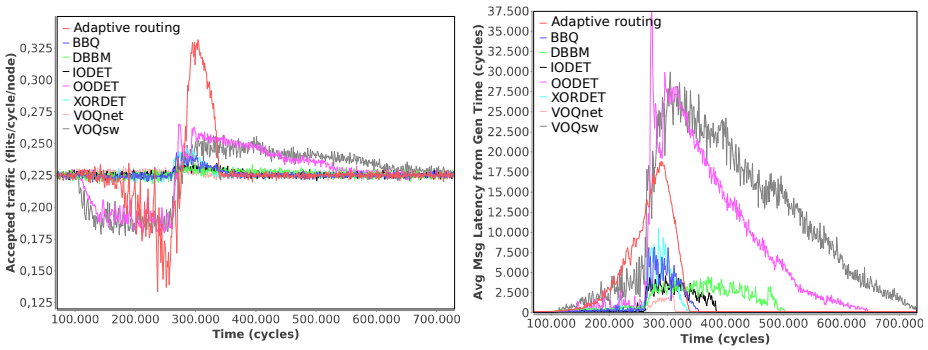


Figure 9: Average packet latency and accepted traffic vs offered load. 256-node 2D-torus. Uniform random traffic pattern. (a,b) XY routing and (c,d) X+Y+X-Y- routing. 4 VCs.

consider all the destination node identifier bits, are less affected by the change in the routing algorithm and obtain roughly the same performance as the one obtained with XY routing.

After analyzing the behavior of the different algorithms under uniform random traffic pattern, next we analyze them under other traffic patterns. First, we will analyze a scenario where the HoL-blocking reduction ability of the routing algorithm may have a great impact. Assume that we have uniform random traffic pattern in the network, but we also introduce a hot-spot node: 25% of network nodes send packets only to one node (the hot-spot node) during some



(a) (b)

Figure 10: Results for hot-spot. 256-node 2D-torus and 8 VCs.

period of time. Traffic injection rate to the hot-spot is computed in such a way that it does not exceed the node ejection bandwidth (1 flit/cycle). The hot-spot traffic starts at clock cycle 100,000 and is active until a number of packets (10,000 in our experiment) have been delivered. This corresponds to clock cycle 260,000. In addition, the remaining nodes (75%) continue generating traffic following a uniform random traffic pattern, that is, sending packets to all the destinations except the hot-spot node. Therefore, during this period of time, the network has two traffic flows: 75% of nodes generate packets with an uniform random traffic pattern and 25% generate packets destined to the hot-spot node. In such a situation, a HoL-blocking reduction routing algorithm should be able to isolate the traffic destined to the hot-spot (i.e. hot flows), thus avoiding interfering the other flows (i.e. cold flows). On the other hand, a fully adaptive algorithm mixes the different flows, spreading the possible congestion to the whole network. To perform this experiment, we have implemented large injection queues at source nodes so that they always can queue a packet if the packet cannot be injected into the network.

This scenario is evaluated in Figure 10 for a 256-node 2D torus. We can see a completely different behavior for the different analyzed routing algorithms. On the one hand, fully adaptive routing, OODET and VOQsw rapidly spread con-

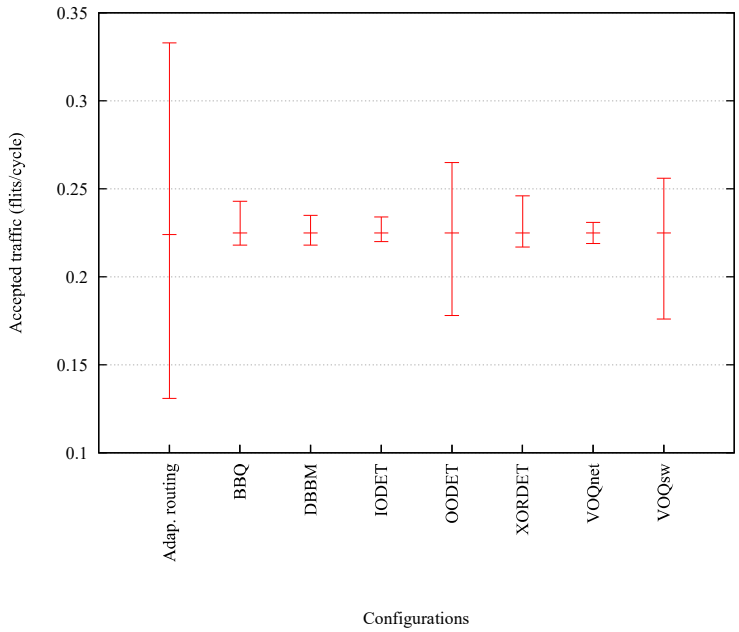


Figure 11: How the hot-spot traffic affects the different routing algorithms.

gestion as packets destined to the hot-spot node interferes other packets, leading to a high reduction in the delivered traffic rate (Figure 10.(a)) and strongly increasing latency (Figure 10.(b)). Only when the hot-spot traffic disappears and after a high number of cycles, the network recovers. Notice that, after the hot-spot traffic is removed, accepted traffic increases for some cycles, due to the high number of messages that have been queued at the injection nodes.

On the other hand, the HoL-blocking reduction deterministic routing algorithms show a much better behavior, close to the one of VOQnet (which requires 256 VCs) without impacting the network throughput and latency in spite of the hot-spot traffic. The exception is DBBM, which requires a higher number of cycles to recover from the hot-spot traffic. This is because the injected uniform random traffic pattern is on the edge of saturation in DBBM. For a better understanding of this behavior, Figure 11, shows the maximum, minimum and average values of accepted traffic in Figure 10.(a). The average value represents the accepted traffic corresponding to uniform random traffic pattern, when it is

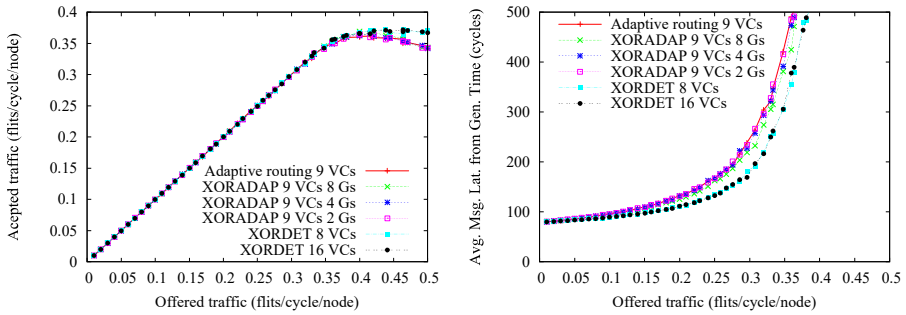
not affected by the hot-spot. The minimum value is reached when the hot-spot is active. And, finally, the maximum value is the one reached after the end of the hot-spot traffic, where queued messages at the injection nodes begin to be received. The closer the three plotted values, the better the behavior of the routing algorithm as it is less affected by the hot-spot traffic. We can see how fully adaptive routing, OODET and VOQsw are strongly affected, obtaining a minimum value more distant to the average value than the remaining routing algorithms.

To summarize, XORDET and IODET were able to reach (with only a few VCs) the same performance as fully adaptive algorithm for uniform random traffic pattern (see Figure 7). Contrary to DBBM and BBQ, they are less affected by changes in the routing algorithm (i.e. the order in which dimensions are crossed, see Figure 9) and they are able to efficiently isolate the hot-spot traffic (see Figure 10). The advantage of XORDET versus IODET is that it is simpler to implement at the internal switch. Remember that XORDET uses virtual networks, but IODET performs VCs changes in the network, which requires additional internal switch connections.

5.1.2. XORADAP evaluation

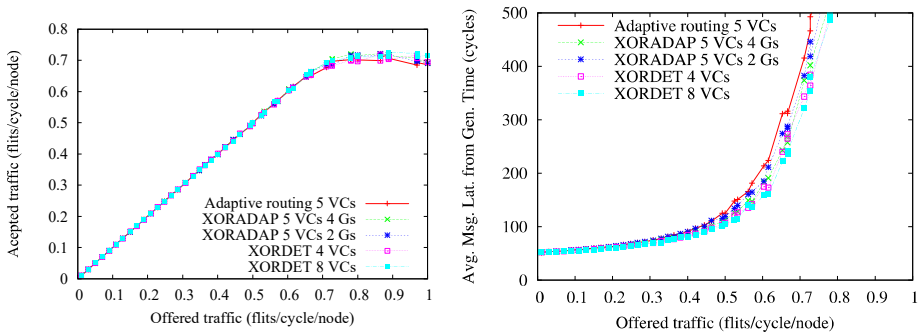
We will first analyze XORADAP with uniform random traffic pattern. Figure 12 shows the results (8 VCs for fully adaptive routing and 1 VC for the escape path). For XORADAP, we selected three different configurations with 9 VCs: two groups with 4 VCs each, 4 groups with 2 VCs and 8 groups with only one VC per group. Remember that more groups of VCs leads to a better packet classification but a lower routing flexibility. Regarding fully adaptive routing, we used the same number of VCs as XORADAP for the sake of fairness. As the number of VCs in XORDET must be a power of two, we evaluated it by using both 8 and 16 VCs.

We can see that all the routing algorithms evaluated obtain roughly the same throughput. Notice, though, that the fully adaptive algorithms suffer a performance degradation after its saturation point [32]. Regarding latency,



(a) (b)

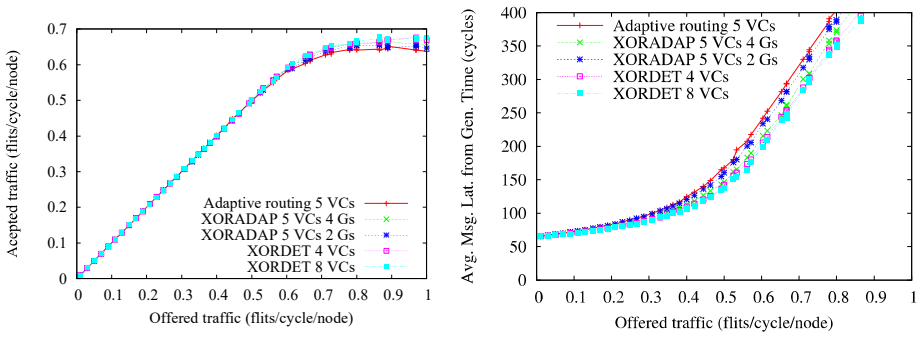
Figure 12: 256-node 2D-torus. Uniform random traffic pattern. (a) Accepted traffic and (b) average packet latency vs. offered traffic.



(a) (b)

Figure 13: 64-node 2D-torus. Uniform random traffic pattern. (a) Accepted traffic and (b) average packet latency vs. offered traffic.

(Figure 12.(b)), for medium to high traffic rates (i.e. 0.3 flits/cycle/node), a higher routing flexibility (i.e., fully adaptive routing or XORADAP with less number of groups of VCs) leads to higher latency values due to the HoL-blocking effect generated by interfering traffic flows. We can see how the XORADAP routing algorithm with more groups of VCs, less adaptive behavior, obtains a slightly lower latency. Both configurations of XORDET obtain the lowest latency values, with almost no differences between them.

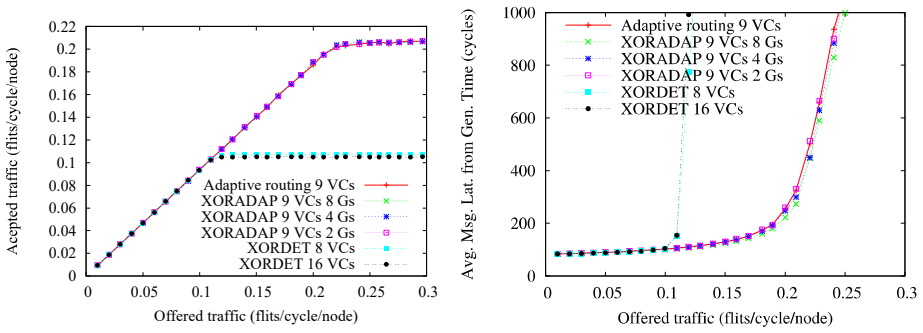


(a) (b)

Figure 14: 512-node 3D-torus. Uniform random traffic pattern. (a) Accepted traffic and (b) average packet latency vs. offered traffic.

Let us analyze networks with different geometry. Figure 13 shows some results for a smaller network with a lower number of nodes per dimension (64-node 2D torus). Figure 14 shows results for a larger network with a higher number of dimensions (512-node 3D torus). In addition, we also tested a different number of virtual channels per physical channel. In particular, 5 VCs (4 adaptive channels plus 1 escape channel) were used for XORADAP and fully adaptive routing. In this case, for XORADAP, we have two groups with 2 VCs each and 4 groups with only one VC per group. 4 VCs and 8 VCs were used in XORDET. As it can be seen, in both cases, the network shows the same behavior we saw in the 256-node 2D torus. As expected, network throughput is higher in these configurations, since we have 8 nodes per dimension instead of 16 and, thus, a better bisection bandwidth. However, the results are qualitatively the same obtained for the 256-node network: more routing flexibility (i.e. fully adaptive routing or XORADAP with a low number of groups of VCs) leads to slightly higher latency.

As we mentioned in Section 4.2, there are some adversarial traffic patterns that significantly impact the performance of the network with deterministic routing algorithms. Nevertheless, XORADAP obtains good performance results



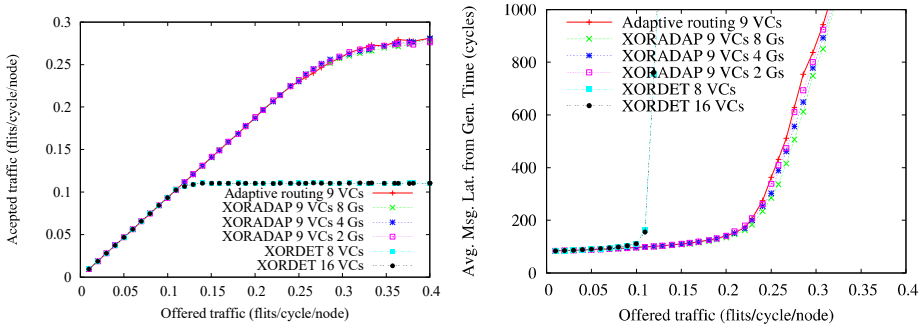
(a) (b)

Figure 15: 256-node 2D-torus. Matrix transpose traffic. (a) Accepted traffic and (b) average packet latency vs. offered traffic.

not only with uniform random traffic pattern, but it is also able to obtain good results with those adversarial traffic patterns.

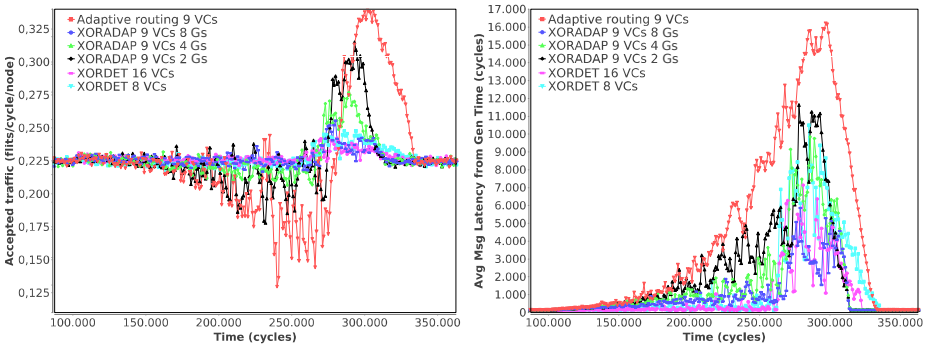
To illustrate this behavior, we have conducted some experiments with the matrix-transpose and bit-reversal traffic patterns. In Figure 15, we compare the behavior of XORDET, fully adaptive routing and the different configurations of XORADAP for the matrix transpose traffic pattern in a 256-node 2D torus. 9 VCs (8 adaptive channels and 1 escape channel) were used in fully adaptive and XORADAP routing algorithms, and 8 VCs and 16 VCs in XORDET. In XORADAP, the three aforementioned configurations were tested: two groups with 4 VCs each, 4 groups with 2VCs and 8 groups with only one per group.

As expected, XORDET obtains a significantly lower throughput than any adaptive algorithm, in spite of using more VCs. In particular, fully adaptive routing more than doubles XORDET performance. This is the weakest point of deterministic routing. It is not able to efficiently cope with adversarial traffic patterns. The poor behavior of XORDET, and, in general, of any deterministic routing, is due to the unbalanced distribution of traffic for this pattern, which leads to over-utilization of some links while other are unused [34]. Concerning the hybrid routing algorithm proposed in this paper, XORADAP, it obtains roughly the same results as fully adaptive routing, since it takes advantage of



(a) (b)

Figure 16: 256-node 2D-torus. Bit-reversal traffic. (a) Accepted traffic and (b) average packet latency vs. offered traffic.



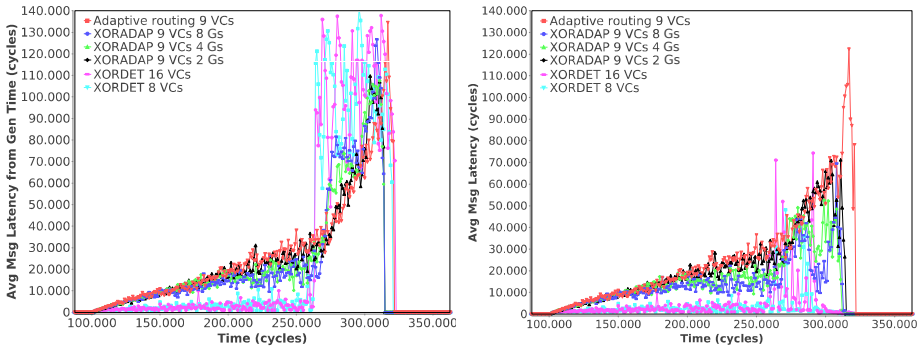
(a) (b)

Figure 17: 256-node 2D-torus. Uniform random traffic pattern with hot-spot. Accepted traffic (a) and average packet latency (b) vs. simulation time.

its flexibility making a better use of the links.

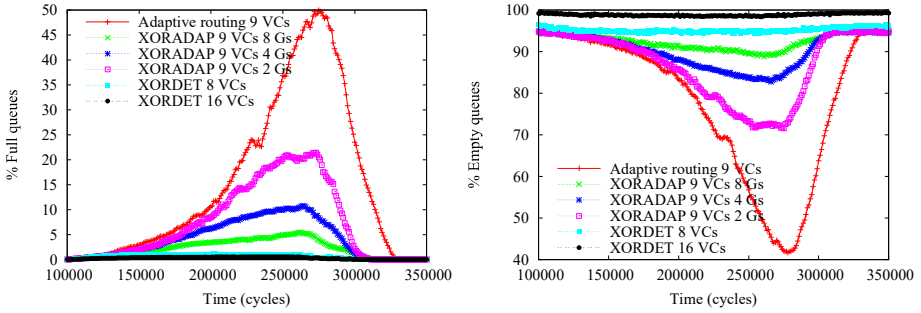
We can see a similar behavior for the bit-reversal traffic pattern in Figure 16. Again, there is a bottleneck when using XORDET deterministic routing. In particular, fully adaptive routing achieves almost 3X throughput than deterministic routing. Any of the configurations of XORADAP is able to reach the same performance obtained with fully adaptive routing.

Considering the results presented up to now, we can confirm that XORADAP



(a) (b)

Figure 18: 256-node 2D-torus. Uniform random traffic pattern with hot-spot. Average packet latency (a) and network latency (b) for packets destined to the hot-spot vs. simulation time.



(a) (b)

Figure 19: 256-node 2D-torus. Uniform random traffic pattern with hot-spot. Completely full (a) and empty (b) queues in the network vs. simulation time.

achieves its first design goals. It is as good as fully adaptive routing for adversarial traffic patterns, thus improving XORDET and deterministic routing in general.

Next, we will analyze XORADAP behavior in the hot-spot scenario, where the HoL-blocking reduction is very important. Figure 17 shows the results for the same experiment performed in Section 5.1.1. Remember that the hot-spot traffic starts at clock cycle 100,000 and it is active until clock cycle 260,000.

As expected, XORADAP helps to achieve a better behavior than fully adaptive routing. In particular, XORADAP configurations with more groups of VCs can better isolate the hot-spot traffic flows, obtaining a more stable value of accepted traffic (in fact very close to XORDET in the best case -XORADAP with 8 groups of VCs-) and a smaller average packet latency. On the other hand, if we use a XORADAP configuration with a few number of groups of VCs, two for example, we obtain a result more close to fully adaptive routing, but with smaller impact on the variability of delivered traffic and reducing packet latency with respect to fully adaptive routing.

Figure 18 confirms this behavior It shows the average message latency and network latency for packets destined to the hot-spot. As we can see in Figure 18-(a), latency strongly increases at cycle 260,000 (i.e. when the hot-spot traffic becomes inactive) for XORDET and XORADAP. However, the network latency (i.e. without considering the time spent at source queues) plotted in Figure 18-(b) does not show the peak. As a consequence, this increase is due to packets that were waiting for long at the injection queues. As the routing algorithm restricted the resources (i.e. the VCs) they can use, packets destined to the hot-spot can not enter the network and must wait at the injection queues. Once the network is able to accept more traffic, these packets can be injected into the network, but the high time they waited at the source injection queues leads to a very high latency. Again, XORADAP with a high number of groups of VCs shows a behavior close to XORDET, while XORADAP with a low number of groups of VCs is close to fully adaptive routing. Therefore, despite for uniform random and adversarial traffic patterns the number of XORADAP groups of VCs did not affect the performance results, for hot-spot traffic a configuration with a high number of groups of VCs seems the best design option.

Another interesting evidence of the behaviour of the evaluated routing algorithms is shown in Figure 19, which shows the percentage of completely full (Figure 19-(a)) or empty (Figure 19-(b)) VC queues in a 256-node 2D network. In order to perform a fair comparison, the shown percentages are relative to the number of VCs of the routing algorithm. As it can be seen, fully adaptive

routing tends to fill up queues along the time the hot-spot is active, which is a symptom of spreading congestion. On the contrary, XORDET keeps most queues empty thanks to the traffic classification it performs. As expected, XORADAP shows a behavior that is half-way between fully adaptive and XORDET routing, depending on the number of groups of VCs.

The analysis shown before demonstrates that XORADAP also achieves its second design goal. It can be as good as a HoL-blocking reduction deterministic routing algorithm to classify and isolate traffic, outperforming fully adaptive routing under hot-spot traffic. To sum up, XORADAP routing algorithm combines the flexibility of adaptive routing with HoL-blocking reduction, being able to efficiently cope with varying networks loads, including uniform random traffic, adversarial or hot-spot traffic. Indeed, for a given number of VCs, several configurations are possible.

5.2. Switch Complexity Analysis

As stated in Section 4.3, routing algorithms that do not perform virtual channel transitions may lead to a simpler implementation of the router switch, as it can be composed of as many small crossbars as virtual channels instead of a larger monolithic crossbar. To quantify this claim, this section estimates the complexity of the switch by using a simple model that takes into account the number of required connections at the internal switch for each configuration.

Several internal switch configurations can be used with virtual channels [11]. We will assume a fully demultiplexed crossbar to implement the internal switch of routers. Although multiplexed crossbar configurations lead to less hardware, it requires more complex arbitration and also internal speedup.

However, although a full crossbar (i.e. with a number of ports equal to the product of the number of physical channels per the number of VCs) is able to cope with any of the analyzed routing algorithms, some connections are not actually required for some of the analyzed routing algorithms. By removing these connections, switch could be simpler. For instance, in all the routing algorithms, packets are never forwarded to the same port it arrived. Indeed,

with DOR, packets may only be forwarded to dimensions higher than the one they entered the router. As a consequence, if the crossbar is implemented by, for instance, using a multiplexer at each output port, the ones corresponding to higher dimensions will have more inputs (and hence, more switching elements) than the ones located in the lower dimensions' ports. On the other hand, with fully adaptive routing, a packet entering through a port may be forwarded to any other output port. When several VCs are used and/or the number of network dimensions is high, the number of internal switch ports grows considerably. In addition, the injection and ejection ports must be also taken into account in any case.

To quantify switch complexity, we will measure the number of required switching elements per switch. We assume that an $i-$input multiplexer needs i switching elements. Table 4 shows the expressions of the number of switching elements for each analyzed routing algorithm taking into account its routing restrictions, n being the number of network dimensions, v the number of VCs per physical channel, and g the number of groups of VCs in the case of XORADAP routing algorithm. As an example, in OODET, output ports of last dimension can be requested by all input ports of lower dimensions (for each VC and for each direction) and by some ports of the same dimension (for each VC from the another direction) and, finally, by the injection port. So, we have $2v(n-1)+v+1$ possible requests per virtual channel for each direction of the last dimension and, in general, $2v(i-1)+v+1$ per virtual channel for each direction of the i dimension. Moreover, we have to add the necessary connections to the ejection port from each virtual channel per direction per dimension, $2vn$. The total number of required switching elements is then given by:

$$2v\sum_{i=1}^{n}(2v(i-1)+v+1)+2vn$$

Table 5 shows the number of required switching elements for the routing algorithms analyzed in this paper for different network configurations (number of network dimensions and number of VCs). Notice that we count an extra virtual channel in adaptive algorithms because XORADAP needs a number of virtual

Routing	Switching Elements
Fully adaptive	$2v\sum_{i=1}^{n}(2nv-2n+2i-v+1)+2vn$
OODET	$2v\sum_{i=1}^{n}(2v(i-1)+v+1)+2vn$
IODET	$2v\sum_{i=1}^{n}(2v(i-1)+1+1)+2vn$
XORADAP	$2\sum_{i=1}^{n}(2nv-2n+2i-v+1)+2(v-1)\sum_{i=1}^{n}(\frac{2nv}{g}-\frac{2n}{g}+2i-\frac{v}{g}+\frac{1}{g})+2vn$
XORDET, DBBM, BBQ	$2v\sum_{i=1}^{n}(2(i-1)+1+1)+2vn$

Table 4: Number of switching elements for each routing algorithm.

#Dim	#VC	Fully adaptive	OODET	IODET	XORADAP 2 Gs	XORADAP 4 Gs	XORADAP 8 Gs	XORDET, DBBM, BBQ
2	2(+1)	120	48	40	94	-	-	32
3	2(+1)	270	96	84	210	-	-	60
4	2(+1)	480	160	144	368	-	-	96
6	2(+1)	1080	336	312	816	-	-	192
2	4(+1)	320	160	112	224	176	-	64
3	4(+1)	750	336	264	510	390	-	120
4	4(+1)	1360	576	480	912	688	-	192
6	4(+1)	3120	1248	1104	2064	1536	-	384
2	8(+1)	1008	576	352	624	432	336	128
3	8(+1)	2430	1248	912	1470	990	750	240
4	8(+1)	4464	2176	1728	2672	1776	1328	384
6	8(+1)	10368	4800	4128	6144	4032	2976	768
2	16(+1)	3536	2176	1216	2000	1232	848	256
3	16(+1)	8870	4800	3360	4830	2910	1950	480
4	16(+1)	16048	8448	6528	8880	5296	3504	768
6	16(+1)	37536	18816	15936	20640	12192	7968	1536

Table 5: Comparison of the number of switching elements

channels that needs to be power of to plus the escape channel. As expected, XORDET, DBBM and BBQ requires a crossbar with the fewest number of switching elements. This is due to two facts. First, messages traversing a given dimension cannot return to the previous dimensions, since DOR routing is used. Therefore, we could dispose those switching elements that connect input ports with output ports of lower dimensions. Second, the VC used by a given packet does not change along the path in the network, like in virtual networks. Hence, there is no need to have a crossbar connection (and the corresponding switching

elements) to allow packets to perform VC transitioning in the same dimension and in the dimension changes.

Both IODET and OODET with DOR routing take also advantage of the first mentioned issue, that is, the connections to previous dimensions can be removed internally at the switch. But regarding the use of virtual networks, neither IODET nor OODET cannot use them. This is because IODET changes the VC when the dimension changes and OODET allows using any of the VCs in the dimension which is being currently crossed. However, for IODET, connections among input and output VCs of the same dimension are not required, but for OODET they are required. Regarding connections to the output VCs of higher dimensions, they are required by both routing algorithms. The worst case is the fully adaptive routing, as it requires a crossbar that enables almost all combinations of physical and VCs (the only exception are connections related to escape paths, which must be traversed in order). On the other hand, XORADAP also needs all combinations to different physical channels since dimensions are used in any order, but connecting only VCs of the same group. There is no need to have a crossbar connection to allow packets to perform VC transitions among different groups of VCs. The more the number of groups of VCs, the less the number of VCs per group and the simpler the crossbar. The connections of escape paths are also required. XORADAP requires a switch complexity that is always lower than fully adaptive routing and, depending on the number of groups of VCs, it can be strongly reduced.

In all cases, as the number of VCs is increased, the number of required switching elements further increases, specially for fully adaptive routing with a high number of dimensions. The deterministic routing design, XORDET, obtains the lowest number of required switching elements. Therefore, XORDET is a good option because of its low complexity. But if what is required is an algorithm which provides flexibility in addition to HoL-blocking reduction, XORADAP is a very good choice because it strongly reduces these requirements compared to fully adaptive routing algorithm, specially for the configurations with a high number of groups of VCs.

6. Conclusions

This paper presents a XOR-based routing mechanism which reduces the HoL-Blocking effect taking advantage of the available virtual channels. Packets are classified in the VCs by performing a XOR bitwise function of their destination node identifiers. This mechanism is used to design both deterministic and adaptive routing algorithms (XORDET and XORADAP, respectively). XORDET deterministic routing obtains performance results close to traditional fully adaptive routing under uniform random traffic pattern. Most important, with hot-spot traffic situations, it is able to isolate the packets destined to the hot-spot node, reducing the interference to the rest of traffic. Furthermore, XORDET keeps the good properties of deterministic routing such as in-order delivery of packets, and simpler switch implementation. However, there are some adversarial traffic patterns which reduce accepted traffic rate with deterministic routing, and XORDET is not an exception. XORADAP routing algorithm tries to combine the best features of both adaptive routing (flexibility under some adversarial traffic patterns) and deterministic routing algorithms specially designed to reduce the HoL-blocking effect (destination isolation) to have the best performance results under any traffic pattern. The evaluation results show that XORADAP: (i) obtains similar performance results to either deterministic or fully adaptive routing with uniform random traffic pattern; (ii) achieves a similar behavior to traditional fully adaptive routing for those traffic patterns where routing flexibility is required to avoid bottlenecks by balancing link utilization; and (iii) copes with hot-spot traffic situations, being able to isolate the packets destined to hot-spots, reducing the interference to the rest of traffic, like HoL-blocking reduction deterministic routing does. Indeed, several configurations of the VCs are possible in XORADAP to enhance either its routing flexibility (i.e. adaptive routing behavior flavor) or its destination isolation (i.e. HoL-blocking reduction deterministic routing behavior flavor).

Acknowledgment

This work was supported by the Spanish Ministerio de Economía y Competitividad (MINECO) and by FEDER funds under Grant TIN2015-66972-C5-1-R and by Programa de Ayudas de Investigación y Desarrollo (PAID) from Universitat Politècnica de València.

References

[1] J. Duato, S. Yalamanchili, N. Lionel, Interconnection Networks: An Engineering Approach, Morgan Kaufmann Publishers Inc., USA, 2002.

[2] W. Dally, B. Towles, Principles and practices of interconnection networks, Morgan Kaufmann, 2004.

[3] TOP500 Supercomputer Site, `http://www.top500.org` Accessed 3 Feb 2016.

[4] M. Karol, M. Hluchyj, S. Morgan, Input vs. Output Queueing on a Space-Division Packet Switch, Communications, IEEE Trans. on 35 (12) (1987) 1347–1356. `doi:10.1109/TCOM.1987.1096719`.

[5] J. Bennett, C. Partridge, N. Shectman, Packet reordering is not pathological network behavior, Networking, IEEE/ACM Trans. on 7 (6) (1999) 789–798. `doi:10.1109/90.811445`.

[6] N. McKeown, V. Anantharam, J. Walrand, Achieving 100% throughput in an input-queued switch, in: INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE, Vol. 1, 1996, pp. 296–302 vol.1. `doi:10.1109/INFCOM.1996.497906`.

[7] C. Gómez, F. Gilabert, M. Gómez, P. López, J. Duato, Deterministic versus Adaptive Routing in Fat-Trees, in: Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International, 2007, pp. 1–8. `doi:10.1109/IPDPS.2007.370482`.

[8] X.-Y. Lin, Y.-C. Chung, T.-Y. Huang, A Multiple LID Routing Scheme for Fat-Tree-Based InfiniBand Networks, in: IPDPS, 2004.

[9] C. chun Su, K. G. Shin, Adaptive fault-tolerant deadlock-free routing in meshes and hypercubes, IEEE Transactions on Computers 45 (1995) 672–683.

[10] M. Thottethodi, A. R. Lebeck, S. S. Mukherjee, Blam: A high-performance routing algorithm for virtual cut-through networks, in: Proceedings of the 17th International Symposium on Parallel and Distributed Processing, IPDPS '03, IEEE Computer Society, Washington, DC, USA, 2003, pp. 45.2–.
URL http://dl.acm.org/citation.cfm?id=838237.838513

[11] W. Dally, Virtual-channel flow control, Parallel and Distributed Systems, IEEE Transactions on 3 (2) (1992) 194–205. doi:10.1109/71.127260.

[12] A. Chien, A cost and speed model for k-ary n-cube wormhole routers, in: Hot Interconnects '93, 1993.

[13] J. Duato, P. López, Performance evaluation of adaptive routing algorithms for k-ary n-cubes, in: K. Bolding, L. Snyder (Eds.), Parallel Computer Routing and Communication, Vol. 853 of Lecture Notes in Computer Science, Springer Berlin, Heidelberg, 1994, pp. 45–59.

[14] L.-S. Peh, W. J. Dally, A delay model and speculative architecture for pipelined routers, in: Proceedings of the 7th International Symposium on High-Performance Computer Architecture, HPCA '01, IEEE Computer Society, Washington, DC, USA, 2001, pp. 255–.
URL http://dl.acm.org/citation.cfm?id=580550.876446

[15] R. Peñaranda, C. Gómez, M. E. Gómez, P. López, J. Duato, HoL-blocking Avoidance Routing Algorithms in Direct Topologies, in: HPCC, IEEE Computer Society, 2014, pp. 11–18.

[16] R. Peñaranda, C. G. Requena, M. E. Gómez, P. López, XORAdap: A HoL-Blocking Aware Adaptive Routing Algorithm., in: M. Daneshtalab, M. Aldinucci, V. Leppänen, J. Lilius, M. Brorsson (Eds.), PDP, IEEE Computer Society, 2015, pp. 48–52.
URL http://doi.ieeecomputersociety.org/10.1109/PDP.2015.50

[17] W. Dally, C. Seitz, Deadlock-Free Message Routing in Multiprocessor Interconnection Networks, Computers, IEEE Transactions on C-36 (5) (1987) 547–553. doi:10.1109/TC.1987.1676939.

[18] C. Carrión, R. Beivide, J.-Á. Gregorio, F. Vallejo, A flow control mechanism to avoid message deadlock in k-ary n-cube networks., in: HiPC, IEEE Computer Society, 1997, pp. 322–329.
URL http://doi.ieeecomputersociety.org/10.1109/HIPC.1997.634510

[19] L. Gravano, G. D. Pifarré, P. E. Berman, J. L. C. Sanz, Adaptive Deadlock- and Livelock-Free Routing with All Minimal Paths in Torus Networks., IEEE Trans. Parallel Distrib. Syst. 5 (12) (1994) 1233–1251.
URL http://doi.ieeecomputersociety.org/10.1109/71.334898

[20] K. D. Underwood, E. Borch, A Unified Algorithm for Both Randomized Deterministic and Adaptive Rout in: IPDPS Workshops, IEEE, 2011, pp. 723–732.
URL http://doi.ieeecomputersociety.org/10.1109/IPDPS.2011.214

[21] A. Singh, W. J. Dally, A. K. Gupta, B. Towles, GOAL: A Load-Balanced Adaptive Routing Algorithm for Torus Networks., in: A. Gottlieb, K. Li (Eds.), ISCA, IEEE Computer Society, 2003, pp. 194–205.
URL http://doi.acm.org/10.1145/859618.859641

[22] L. G. Valiant, G. J. Brebner, Universal Schemes for Parallel Communication, in: STOC, ACM, 1981, pp. 263–277.
URL http://doi.acm.org/10.1145/800076.802479

[23] J. Duato, A Necessary and Sufficient Condition for Deadlock-Free Routing in Cut-Through and Store-and-Forward Networks, IEEE Transactions on Parallel and Distributed Systems 7 (1996) 841–854. doi:10.1109/71.532115.

[24] W. Dally, P. Carvey, L. Dennison, Architecture of the avici terabit switch/router, in:Proceedings of Hot Interconnects 6.

[25] T. E. Anderson, S. S. Owicki, J. B. Saxe, C. P. Thacker, High-speed switch scheduling for local-area networks, ACM Trans. Comput. Syst. 11 (4) (1993) 319–352. doi:10.1145/161541.161736.

[26] J. Duato, J. Flich, T. N. Frinós, A Cost-Effective Technique to Reduce HOL Blocking in Single-Stage and in: PDP, IEEE Computer Society, 2004, pp. 48–53.
URL http://doi.ieeecomputersociety.org/10.1109/EMPDP.2004.1271426

[27] T. Skeie, O. Lysne, I. Theiss, Layered shortest path (LASH) routing in irregular system area networks, in: 16th International Parallel and Distributed Processing Symposium (IPDPS 2002), 15-19 April 2002, Fort Lauderdale, FL, USA, CD-ROM/Abstracts Proceedings, 2002. doi:10.1109/IPDPS.2002.1016559.
URL http://dx.doi.org/10.1109/IPDPS.2002.1016559

[28] P. Yebenes, J. Escudero-Sahuquillo, C. Gomez, P. J. Garcia, F. J. Quiles, J. Duato, BBQ: a straightforward queuing scheme to reduce hol-blocking in high-performance hybrid networks, in: Euro-Par 2013, Springer, 2013, pp. 699–712.

[29] R. Peñaranda, C. Gómez, M. E. Gómez, P. López, J. Duato, IODET: A HoL-blocking-aware Deterministic Routing Algorithm for Direct Topologies., in: ICCS, IEEE Computer Society, 2012, pp. 702–703.

[30] A. González, M. Valero, N. Topham, J. M. Parcerisa, Eliminating cache conflict misses through xor-based placement functions, in: Proceedings of the 11th International Conference on Supercomputing, ICS '97, ACM, New York, NY, USA, 1997, pp. 76–83.

doi:10.1145/263580.263599.
URL http://doi.acm.org/10.1145/263580.263599

[31] S. McFarling, Combining branch predictors, Tech. rep., Technical Report TN-36, Digital Western Research Laboratory (1993).

[32] P. López, J. Duato, Deadlock-Free Adaptive Routing Algorithms for the 3D-Torus: Limitations and Solutions, in: A. Bode, M. Reeve, G. Wolf (Eds.), PARLE'93, Parallel Architectures and Languages Europe, Vol. 694 of Lecture Notes in Computer Science, Springer Berlin, Heidelberg, 1993, pp. 684–687.

[33] M. E. Gómez, J. Duato, J. Flich, P. López, A. Robles, N. A. Nordbotten, T. Skeie, O. Lysne, A New Adaptive Fault-Tolerant Routing Methodology for Direct Networks., in: L. Bougé, V. K. Prasanna (Eds.), HiPC, Vol. 3296 of Lecture Notes in Computer Science, Springer, 2004, pp. 462–473.

[34] T. Hoefler, T. Schneider, A. Lumsdaine, Multistage switches are not crossbars: Effects of static routing in high-performance networks., in: CLUSTER, IEEE Computer Society, 2008, pp. 116–125.
URL http://doi.ieeecomputersociety.org/10.1109/CLUSTR.2008.4663762