This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2409865, IEEE Transactions on Parallel and Distributed Systems

Assessing the Suitability of King Topologies for Interconnection Networks

E. Stafford, J.L. Bosque, C. Martínez, F. Vallejo, R. Beivide, C. Camarero, E. Castillo

Abstract—In the late years many different interconnection networks have been used with two main tendencies. One is characterized by the use of high-degree routers with long wires while the other uses routers of much smaller degree. The latter rely on two-dimensional mesh and torus topologies with shorter local links. This paper focuses on doubling the degree of common 2D meshes and tori while still preserving an attractive layout for VLSI design. By adding a set of diagonal links in one direction, *diagonal networks* are obtained. By adding a second set of links, networks of degree eight are built, named *king networks*. This research presents a comprehensive study of these networks which includes a topological analysis, the proposal of appropriate routing procedures and an empirical evaluation. King networks exhibit a number of attractive characteristics which translate to reduced execution times of parallel applications. For example, the execution times NPB suite are reduced up to a 30%. In addition, this work reveals other properties of king networks such as perfect partitioning that deserves further attention for its convenient exploitation in forthcoming high-performance parallel systems.

1 Introduction and Related Work

Although a lot of research on interconnection networks has been conducted in the last decades, constant technological changes demand new insights about this key component in modern computers.

Many high-performance system networks are based on torus topologies, [1], [2], [3]. Notwithstanding, some recent and interesting papers advocate for networks with high-radix routers in large-scale supercomputers [4], [5], [6]. The advent of economical optical signaling enables this kind of topologies that use long global wires. The design scenario is very different when addressing on-chip networks. So far, multi-ring networks have been the most common choice for on-chip networks, [7]. Nevertheless, with current VLSI technology, the planar substrate in which networks are implemented suggests the use of 2D mesh-like topologies. This has been the case of Tilera [8] and the Intel's Teraflop research chip [9], with 64 and 80 cores respectively arranged in a 2D mesh. Evolving from rings, 2D tori have been considered, [10]. On-chip networks with higher degree than traditional 2D meshes or tori have also been recently explored [11]. In addition, multi-level trees are being considered for forthcoming many-core chips using up to thousands of simple execution engines orchestrated by a much smaller number of control engines, [12]. Such networks entail the use of long wires in which repeaters and channel pipelining are needed. Forthcoming technologies such as on-chip high-speed signaling and optical communications could favor the use of higher degree on-chip networks.

This paper, explores an intermediate solution: doubling the radix of a common 2D mesh while still preserving an attractive layout for a planar design. By adding diagonal links in one direction, *diagonal networks* with radix six, are obtained. Next, by adding a second set of links perpendicular to the former, networks of degree eight are created. In these, a packet located in any node can travel in one hop to any of its eight neighbours just like the king on a chessboard. For this reason, these topologies are denoted *king networks*. Hence, these networks include the typical four orthogonal links of meshes and tori but also four new diagonal links. Such networks will be explored in this paper in two versions, without and with wrap-around links, referring to them as *king meshes* and *king tori*. In this way, a more conservative evolution towards higher radix networks is presented, that tries to exploit their advantages while avoiding (or minimizing) the use of long wires. The simplicity and topological properties of these networks offer tantalizing features for future high-performance computers: higher throughput, smaller latency, easily partitioned in smaller networks, good scalability and high fault-tolerance.

The proposal of topologies using diagonal links has been considered in the past in the fields of microprocessor design [13], FPGAs [14] and interconnection networks [15]. Also mesh and toroidal topologies with added diagonals have been proposed. With degree six [16] proposes a torus network with hexagonal shape. These contrast with the square shape of the diagonal networks presented here. Also meshes with diagonals in both directions were described in [17]. As a means to avoid deadlocks the latter used two vertical links, meaning that they are of degree ten, rather than eight like the king meshes presented in this work. King lattices have been previously studied in several papers in the field of Information Theory [18]. In that context the distance among nodes in a king network is denoted as the Tchebychev distance.

The goal of this paper is to assess the suitability of king topologies to constitute the communication substrate of forthcoming parallel systems. With this idea in mind, it presents the foundations of king networks and a first attempt to unleash their potential. The main contributions of our research are the following:

i) A discussion of the problem of increasing the de-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2409865, IEEE Transactions on Parallel and Distributed Systems

gree of planar networks.
ii) An in-depth analysis of the topological characteristics of king meshes and king tori comparing them with diagonal networks and 2D meshes and tori.
iii) The evaluation of king tori, not considered previously in the technical literature.
iv) A folding scheme that ensures king tori scalability.
v) Adaptive and deadlock-free routing algorithms for diagonal and king topologies.
vi) A performance evaluation of king networks based on synthetic traffic and trace-driven simulation.

The remainder of this paper is organized as follows. Section 2 is devoted to define the network topologies considered in this paper. The most relevant distance parameters, the bisection bandwidth and path diversity are computed for each network and a folding method is considered for the topologies with wrap-around links. Next, Section 3 presents a general view of the router architecture considered in this paper, followed by a detailed description of the routing algorithms used for each topology. In the case of king networks, several algorithms are considered, each one improving on the previous, to finally reach an acceptable solution. Section 4 contains the performance evaluation of the networks with the different routing algorithms, both with synthetic traffic and trace-driven simulations. Last, Section 5 concludes by highlighting the most important findings.

2 KING NETWORKS

As usual, networks are modeled by graphs, where graph vertices represent routers and edges represent the communication links among them. In this paper only square networks will be considered, as sometimes networks with sides of different length exhibit an unbalanced link usage in each dimension [19]. The adjective "square" will be assumed for the rest of the paper. As a consequence, for any of the networks considered the number of nodes will be $N = s^2$, for any integer $s > 1$, which represents their side.

The common mesh of side s will be denoted as M_s. This is a very well-known topology which has been deeply studied. In a mesh, all the nodes which are not in the periphery have four different neighbours, that is, they have degree four. Similarly, most nodes in the periphery have degree three and the ones at the corners have two. Since mesh nodes have degrees four, three and two, it is said that a mesh has maximum degree four. Next, different approaches are considered to increase the maximum degree of the mesh.

Traditionally, increasing the degree of meshes has lead to 3D meshes. These are built by stacking several 2D meshes and adding connections to allow each node to communicate with the nearest nodes above and below. While this option is reasonable for system networks, limitations on the size of the Z dimension in stacked technologies condition its use for on-chip networks. Currently, 3D stacking technology is being used to pile a few memory layers over the computing substrate. Hence, the computing nodes still remain in a single layer and it makes sense to have a planar interconnection network. Therefore, this article is focused on networks whose nodes can be naturally laid out on a plane, such as meshes and tori. Thus, the way of increasing the degree will be by adding diagonal links. Figure 1 shows examples of the networks considered in this work.

Initially diagonal links will be added to a mesh in one direction, which will named *diagonal mesh*. Note that there are two possible diagonal meshes, one with north-east links and another with north-west links. Both are denoted as DM_s, since both graphs are isomorphic. Straightforwardly, DM_s have maximum degree six.

Then, mesh based networks of maximum degree eight can be obtained by adding diagonal links in both directions, north-east and north-west. These will be denoted as *king meshes* or KM_s. In other words, a KM_s is obtained by adding both sets of diagonal links to a M_s. Next, these topologies are formally defined.

Definition 1: Let $V = \{(x, y) \in \mathbb{Z} \times \mathbb{Z} \mid x, y = 0, \ldots, s - 1\}$ be the set of nodes of the three topologies. Now, the definition of the network is completed with the rule to determine the links between two nodes. Therefore, any node (x, y) will be connected to the nodes:

- $(x, y) \pm (1, 0), \pm(0, 1)$ in the M_s.
- $(x, y) \pm (1, 0), \pm(0, 1), \pm(1, 1)$ (respectively $\pm(-1, 1)$) in the DM_s.
- $(x, y) \pm (1, 0), \pm(0, 1), \pm(1, 1) \pm (-1, 1)$ in the KM_s.

All these connections are made when the resulting node is in fact in the set of nodes (peripheral nodes lack of some of the previously defined links).

Note that all these mesh based networks are neither regular nor vertex-symmetric. The way to turn these into regular and vertex-symmetric networks is to add wrap-around links. These links allow every node to have the same number of neighbours. The common 2D torus of side s will be denoted as T_s. The torus is obviously the four degree regular counterpart of the mesh. Then, DT_s will denote the *diagonal torus* of side s, that is, the degree six graph obtained by adding wrap-around links to the corresponding diagonal mesh. Such diagonal tori can be seen as particular cases of Eisenstein-Jacobi graphs, which were introduced in [20]. Finally, KT_s will denote the *king torus* network, that is, a king mesh with the wrap-around links required to obtain a regular network with degree eight. Another way to see this network is as a torus with extra diagonal links that turn the degree four torus into a degree eight one. In the next definition a formal description of the topologies is given.

Definition 2: Let $V = \{(x, y) \in \mathbb{Z} \times \mathbb{Z} \mid x, y = 0, \ldots, s - 1\}$ be the set of nodes of the three topologies. Now, the definition of the topologies is completed with the rule to determine the links between two nodes. Therefore, any node (x, y) will be connected to the nodes:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2409865, IEEE Transactions on Parallel and Distributed Systems

- $(x, y) \pm (1, 0), \pm(0, 1) \pmod{(s, s)}$ in the T_s [1].
- $(x, y) \pm (1, 0), \pm(0, 1), \pm(1, 1) \pmod{(s, s)}$ (respectively $\pm(-1, 1)$) in the DT_s.
- $(x, y) \pm (1, 0), \pm(0, 1), \pm(1, 1) \pm (-1, 1) \pmod{(s, s)}$ in the KM_s.

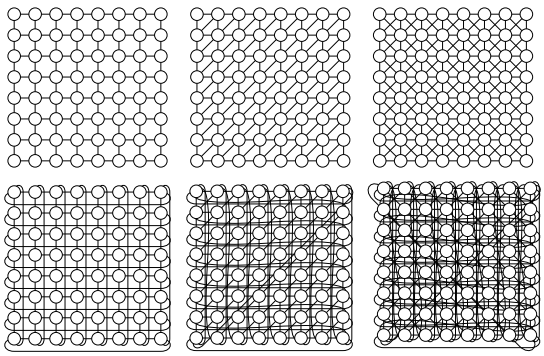


Fig. 1. Examples of Mesh, Diagonal Mesh, King Mesh, Torus, Diagonal Torus and King Torus Networks.

Minimizing packet latency is a target for any interconnection network. Base latency depends on the number of hops traversed, plus the spooling time in multiphit packets. Hence, in a first approach, transmission delays in networks can be inferred from their topological properties. The maximum packet delay is given by the diameter. Which is the length of the minimum-distance path connecting the most distant nodes. The average delay is proportional to the average distance, which is computed by averaging the length of all minimum paths connecting every pair of nodes. Table 1 records these parameters for the networks considered. For the mesh and torus, these are well-known values [21]. The distance properties of king torus were presented in [22]. Also, the distance distribution of diagonal tori, from which the diameter and average distance were derived, was presented in [23].

An especially important metric of interconnection networks is the maximum data rate the network can deliver, also known as maximum throughput. For uniform traffic, where nodes send packets to random destinations with uniform probability, the throughput is bounded by the network bisection bandwidth. According to [21], in networks with homogeneous channel bandwidth, as the ones considered here, the bisection bandwidth is proportional to the channel count across the smallest cut that divides the network into two equal halves. This value represents an upper bound for the maximum throughput under uniform traffic. In Table 1, bisection values for mesh and torus are shown [21].

The bisection bandwidth for diagonal meshes and tori can be straightforwardly obtained by counting the number of links that need to be cut to split the network in two halves. For example, a vertical cut in a diagonal mesh of side s, affects s horizontal links and s diagonals. As these are bidirectional, the bisection bandwidth is $4s$. For diagonal tori, the bandwidth is duplicated because a torus requires two cuts to disconnect it. The same reasoning can be applied to king meshes and tori. It is noteworthy that a king network doubles the number of links of its degree four counterpart but has three times the bisection bandwidth.

Network	M_s	DM_s	KM_s	T_s	DT_s	KT_s
Diameter	$2s$	$2s$	s	s	s	$\lfloor \frac{s}{2} \rfloor$
Avg. Distance	$\approx \frac{2}{3}s$	$\approx \frac{17}{30}s$	$\approx \frac{7}{15}s$	$\approx \frac{s}{2}$	$\approx \frac{7}{18}s$	$\approx \frac{s}{3}$
Bisection Bw.	$2s$	$4s$	$6s$	$4s$	$8s$	$12s$

TABLE 1
Topological Parameters

An interesting property of square king networks is that they are *denser* than their standard counterparts. A graph is said to be *dense* when the number of reachable nodes for a given diameter is maximum. Dense graphs have the following advantage: in the same amount of time, they can deliver a packet to more nodes than those which are not dense. For example, a torus of side (and diameter) k, with k even, can arrange just k^2 nodes. Notwithstanding, in an infinite torus, $4d$ nodes can be found at distance d from any given node. By adding the reachable nodes at distances 1 through $\frac{k}{2}$ the expression $2k^2 + 2k + 1$ is obtained, which gives the number of nodes in the dense mesh-based graph of diameter k. Note that it is more than twice the number nodes in a common 2D torus, meaning that it is far from being dense. Dense tori were deeply studied in [24]. The same conclusion can be reached when considering the degree six, diagonal networks. One of side s will have diameter s and s^2 nodes. Which is less than the densest network of degree six and diameter s, that would have $3s^2 + 3s + 1$ nodes.

However, in the case of king topologies, all the square networks with odd side s are dense. Note that, in these topologies, at each distance d, $8d$ different nodes can be found. Therefore, for diameter $k = \lfloor \frac{s}{2} \rfloor$, a king torus has the following number of nodes:

$$1 + \sum_{d=1}^{k} 8d = 4k^2 + 4k + 1 = (2k + 1)^2 = s^2$$

In a king mesh, this number of nodes can be arranged with diameter s. Hence, in the case of square king topologies, with odd side, the maximum achievable number of nodes for a given diameter is always attained.

It is important to note that such node density is preserved across network partitions for king networks of odd side. For example, consider a king mesh of odd side $s = 15$. It will have $s^2 = 225$ nodes, and its diameter will be $k = s = 15$. Such network can be decomposed into 25 (5×5) equal dense sub-networks with diameter 1, each one composed of 9 nodes. Conversely, the network

1. The notation $\pmod{(s, s)}$ means to take modulo in each component, that is, $(4, 16) \pmod{(8, 8)} = (4 \pmod 8, 16 \pmod 8) = (4, 0)$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2409865, IEEE Transactions on Parallel and Distributed Systems

can also be partitioned into 9 (3×3) equal dense sub-networks with diameter 3, each one composed of 25 nodes. The case of such odd networks is especially interesting as the network, as well as any partition, has a center node that can be considered the center of a ball with radius the sub-graph diameter. Furthermore, finding these center nodes is straightforward, they are evenly distributed across the network and every node is closest to only one of the center nodes. The graphical representation of a partitioned king mesh where the center nodes are highlighted can be seen in Figure 2. This property is of great value for hierarchical designs in which the center nodes represent a singular device (memory controller, complex core, network bridge, etc). Current examples of such designs are the Runnemede project from Intel [12] in which the central nodes would be complex cores and other CMP designs in which central nodes would be memory controllers, [25].

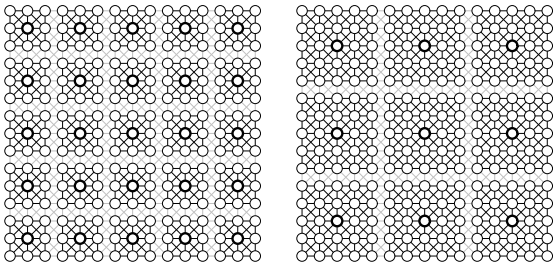


Fig. 2. Depiction of KM_{15} partitioned into 25 KM_3 or 9 KM_5 with highlighted center nodes.

Another graph property with relevance in interconnection networks is the *path diversity*. This is defined as the number of minimal paths between a given pair of nodes a, b and conditions the capability of packets to avoid congested areas without increasing the path length. For mesh and tori it will be denoted as $|R_{ab}|$:

$$|R_{ab}| = \binom{|\Delta_x| + |\Delta_y|}{|\Delta_x|}.$$

Similarly, in diagonal mesh and tori $|R_{ab}|$ is:

$$|RD_{ab}| = \binom{|\Delta_x|}{|\Delta_x| - |\Delta_y|} = \binom{|\Delta_x|}{|\Delta_y|}.$$

Finally, in king mesh and tori the path diversity is:

$$|RK_{ab}| = \binom{|\Delta_x|}{|\Delta_y|}_2$$

This expression uses the trinomial coefficient.[2]

Considering technological restrictions, physical implementation of regular (not hierarchic) interconnection networks usually requires that the length of all the links to be equal, or at least similar. In the context of on-chip networks, mesh implementation is fairly straightforward. Diagonal and regular meshes can be laid out with a single metal layer. Due to the crossing diagonal links, the king mesh would require two metal layers.

A standard torus could be implemented with two metal layers. However, tori have wrap-around links whose length depend on the size of the network. This seriously restricts the scalability of tori. To equalize the link length, a well known technique is graph folding [21]. The proposed approach to folding king tori is based on that of the common torus, but because of the diagonal links four metal layers are required. As a consequence of this folding process, the length of the links is between 2 and $\sqrt{8}$. Figure 3 shows a 8×8 folded king torus. Clearly, the same approach can be considered for tori with just a diagonal, which leads to a folded networks in which the maximal length of the links is also upper bounded by $\sqrt{8}$.

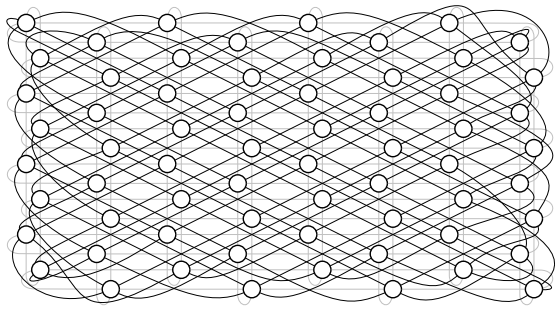


Fig. 3. Folding of King Torus Network. For the sake of clarity, the orthogonal links are shown in gray.

This Section summarised the possible advantages that king networks present. The noticeable improvements on basic network parameters of king networks over their standard counterparts invite to a more detailed study that will be carried out in the remainder of the paper. In addition, specifically comparing king meshes against standard tori shows that the cost of doubling the number of links of the mesh gives great returns. Bisection bandwidth is 50% larger, average distance is almost 5% less and diameter remains the same. It should be noted that implementing a king mesh on a planar substrate is simpler, as it does not need to be folded and fits in two metal layers just like a folded torus. A potential drawback is that, in contrast to meshes, tori are regular and vertex symmetric networks.

3 Network Routing

The performance of a given topology is heavily conditioned by the routing strategy used to guide the packets. Not making the right choices will certainly reduce the network's performance, but can also render it unusable due to the occurrence of deadlocks or other anomalies. An analysis of routing schemes is capital and this Section delves into such a study for the described topologies.

3.1 Router Model

Once the topology has been established, the most important component of the interconnection network is the

2. Trinomial coeficient: $\binom{n}{k}_2 = \sum_{j=0}^{n} (-1)^j \binom{n}{j} \binom{2n-2j}{n-k-j}$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2409865, IEEE Transactions on Parallel and Distributed Systems

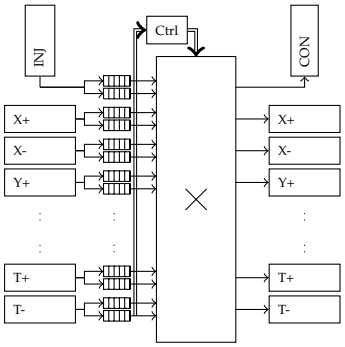


Fig. 4. Basic router organization.

router. It will accept data packets from the computing nodes, send them through the network and deliver them to the destination nodes. It is in charge of making appropriate path selection and resource sharing to maximise the performance while avoiding congestion or deadlock. Thus, knowing its internal structure is important to adequately design the routing strategy. From the outside, the router shows a set of ports through which it will be connected to the computing nodes and neighbouring routers, as established by the topology. In the case of a king torus, the router has eight full-duplex ports connecting it to the eight nearest neighbours plus one to the computing node. The internal structure of the router is covered extensively in the literature [21]. Figure 4 describes the basic router data-path showing the usual hardware modules: ports, queues and crossbar, as well as a router control block responsible of arbitration and routing logic, which is the main concern of this Section.

When a router receives a packet from the computing element, it must decide how it should traverse the network. Some networks use *source routing*, where the source router alone decides on how the packet reaches its destination. This might be done by consulting a routing table and explicitly writing on the packet header the list of nodes it must visit to successfully reach its destination. However in 2D networks, routing is typically done by means of a *routing record*. It is a vector specifying the number of hops the packet must make in each dimension. And, instead of looking it up in a table, it can be calculated with simple arithmetic operations. This vector is written on the packet header and is modified with each hop, so that it always indicates the remaining hop counts. A routing record with all counts set to zero means the packet has reached its destination.

It is known that letting packets traverse the network freely will soon cause deadlock. The use of *Dimension Ordered Routing (DOR)* prevents this situation in mesh-like networks by forcing packets to exhaust the routing record in one dimension before advancing in the next. However this comes at a cost. The packets are forced to follow a single path and can run into local congestion that could be avoided if they could change direction anywhere, adapting to the traffic situation. Using *virtual channels* can improve this drawback. These are implemented by properly dividing the buffer space assigned to a communication link so that several flows (depending on the number of virtual channels) of packets can share the physical link. Employing different routing policies on different virtual channels can provide adaptivity and avoid deadlock, [26]. In its simplest form, this mechanism works with just two virtual channels: one without the DOR restriction allows packets to avoid congestion and when any of them is in risk of causing deadlock, it changes to the other channel that operates with DOR and ensures packet delivery.

The above only applies to mesh networks, as the dependency loops susceptible of becoming deadlocked span the two dimensions of the mesh and are broken by the DOR policy. But mesh networks have the problem of being irregular and this usually causes an unbalanced use of their links. As a consequence, the observed performance is lower than predicted by a simple topological analysis. With a similar amount of resources, tori have better topological characteristics, and as they are regular, the appearance of central congestion is less likely. However, in general, using just two virtual channels is not sufficient to achieve deadlock freedom. This is due to the fact that tori are composed of set of rings in each dimension, which are clearly deadlock prone. There are several ways of dealing with this shortcoming. Dateline, a technique which uses two virtual channels to break deadlocks on rings together with DOR has been used in several implementations, [21]; adaptive routing on a 2D torus using dateline requires three virtual channels. Nevertheless, the experiments shown in this article have been carried out using *Adaptive Bubble Routing (ABR)* [27] [28], a development of the work presented in [29] and [30], applied to *Virtual Cut-Through (VCT)* [31]. With ABR, two virtual channels are enough to provide adaptive deadlock-free routing. This routing/flow-control mechanism is the one used by IBM BlueGene supercomputers [32].

3.2 Routing in Diagonal networks

To devise a routing scheme for diagonal networks, it is important to notice differences and similarities with the well known 2D networks. Like the latter, diagonal networks have their nodes arranged in a square and each node can be addressed by its Cartesian coordinates. On the other hand, 2D networks have vertical and horizontal links corresponding to the two dimensional plane in which they are contained. This contrasts with diagonal networks, which are also contained in a plane, but have the extra diagonal links. Then there are three link directions but only two spatial dimensions. This forces the routing record to have an extra component, one for each direction. Otherwise the packets would not use the diagonal links.

Now, a mechanism must be found to derive the routing record from the source and the destination addresses of incoming packets. For 2D meshes the routing record

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2409865, IEEE Transactions on Parallel and Distributed Systems

(Δ_x, Δ_y) is found by subtracting the source address (x_0, y_0) from the destination (x_1, y_1):

$$(\Delta_x, \Delta_y) = (x_1 - x_0, y_1 - y_0)$$

This routing record can be used as a basis for determining the three component routing record $(\delta_x, \delta_y, \delta_z)$ required for diagonal networks. Considering that a packet is headed to a node in the first or third quadrants, that is Δ_x and Δ_y have the same sign, then each pair of hops in x and y directions can be swapped for a hop in the z direction as follows:

$$(\delta_x, \delta_y, \delta_z) = \begin{cases} (\Delta_x - \Delta_y, 0, \Delta_y) & |\Delta_x| \geq |\Delta_y| \\ (0, \Delta_y - \Delta_x, \Delta_x) & |\Delta_x| < |\Delta_y| \end{cases}$$

In any other case, there is no benefit in using the diagonal links. Therefore, when the destination is found in the second and fourth quadrants,

$$(\delta_x, \delta_y, \delta_z) = (\Delta_x, \Delta_y, 0)$$

An interesting fact of this routing approach is that there is no packet that can have three non-zero components in its routing record. This is a consequence of the z direction being a linear combination of x and y.

Although this basic algorithm can be useful for diagonal tori, it can not be directly applied. The reason behind this is better understood when considering a ring of nodes. Any pair of nodes is connected by two segments on a ring. In general one will be shorter than the other and this will be the one chosen to send packets. However there are cases in which both segments have the same length, then packets must be equally distributed between both segments. Otherwise the traffic on the ring becomes unbalanced, reducing performance. If, on a ring, packets might need to choose between two paths, on tori there can be up to four, one for each quadrant. With this in mind, a proper routing mechanism for diagonal tori is shown in Algorithm 1. It computes the length of the four candidate paths connecting source and destination. Then finds which one of them is the shortest. And because there can be several, it selects one at random. Once the path is chosen, it is just a question of applying the diagonal mesh algorithm. With this, experimentation shows that the use of all directions is balanced. That is, under uniform traffic the load on all links is the same. As in the mesh algorithm, it is impossible for any routing record to have more than two non-zero components.

3.3 Routing in King networks

Initially, routing on king networks might seem trivial. But king networks have a peculiarity that has a profound influence on the routing algorithm design. King networks are not locally planar. That is, a king mesh can not be laid out on a plane without edges crossing. This fact will lead to a set of different solutions, each with improvements over the previous one, as will be presented next.

Input: $f = (x_0, y_0)$: source, $t = (x_1, y_1)$: destination, s: side
Output: $r = (r_X, r_Y, r_Z)$: routing record
begin
- $(x, y) \leftarrow t - f$
- **if** $x < 0$ **then** $x \leftarrow x + s$
- **if** $y < 0$ **then** $y \leftarrow y + s$
- $p_0 \leftarrow max(x, y)$
- $p_1 \leftarrow max(s - x, s - y)$
- $p_2 \leftarrow s - x + y$
- $p_3 \leftarrow s - y + x$
- `// Find index of minimum paths`
- $I \leftarrow \{i \in 0..3 / p_i = min(p_0, p_1, p_2, p_3)\}$
- `// Randomly select minimum path`
- $s \leftarrow randomelement(I)$
- **if** $s = 0$ **then**
  - **if** $x > y$ **then**
    - $r_Z \leftarrow y$
    - $r_X \leftarrow p_0 - y$
  - **else**
    - $r_Z \leftarrow x$
    - $r_Y \leftarrow p_0 - x$
- **if** $s = 1$ **then**
  - **if** $x > y$ **then**
    - $r_Z \leftarrow x - s$
    - $r_Y \leftarrow s - x - p_1$
  - **else**
    - $r_Z \leftarrow y - s$
    - $r_X \leftarrow s - y - p_1$
- **if** $s = 2$ **then**
  - $r_X \leftarrow x - s$
  - $r_Y \leftarrow y$
- **if** $s = 3$ **then**
  - $r_X \leftarrow x$
  - $r_Y \leftarrow y - s$

end

Algorithm 1: Minimal routing algorithm for a diagonal torus.

3.3.1 Knaive

The first approach is heavily based on the diagonal mesh routing algorithm. Because of the regularity of king tori, there is no need to have a substantially different algorithm than for king meshes. Both of them start by obtaining the routing record for the corresponding 2D counterpart (Δ_x, Δ_y). From then on, both algorithms are identical. As king networks have higher degree than the diagonal networks, another extension of the routing record is necessary. Because they have links oriented in four different directions, the routing record will have four components $(\delta_x, \delta_y, \delta_z, \delta_t)$. The computation of the different components is similar to the method proposed for diagonal meshes. If the destination is in the first or third quadrants, where Δ_x and Δ_y have equal sign, an attempt will be made to use direction z as when possible:

$$(\delta_x, \delta_y, \delta_z, \delta_t) = \begin{cases} (\Delta_x - \Delta_y, 0, \Delta_y, 0) & |\Delta_x| \geq |\Delta_y| \\ (0, \Delta_y - \Delta_x, \Delta_x, 0) & |\Delta_x| < |\Delta_y| \end{cases}$$

Whereas if the destination is located in the second or fourth quadrants, Δ_x and Δ_y have different sign, direction z is useless and it is t that has to be exploited:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2409865, IEEE Transactions on Parallel and Distributed Systems

$$(\delta_x, \delta_y, \delta_z, \delta_t) = \begin{cases} (\Delta_x + \Delta_y, 0, 0, \Delta_y) & |\Delta_x| \geq |\Delta_y| \\ (0, \Delta_x + \Delta_y, 0, -\Delta_x) & |\Delta_x| < |\Delta_y| \end{cases}$$

Due to its elegant simplicity and straightforward implementation, this algorithm is denoted *Knaive*. Its main advantage is that, as experimentally reported later on the paper, it perfectly balances the use of all directions in uniform traffic. Moreover, empirical results revealed a correlation with the topological properties of Table 1.

Unfortunately, network traffic is not always uniform. There are other traffic patterns, modeled as shuffle, complement or tornado and others in [21], that cause packets to congest small areas of the network while leaving others unused. These adverse situations cause a diminished performance, even when using adaptive routing. In general, these are named *adverse traffic patterns*, in contrast to *benign traffic patterns*, where packets are naturally spread around the network.

As will be shown later, Knaive performs poorly in adverse traffic patterns. The reason being that, although king networks have four routing record components, Knaive can not have more than two greater than zero. The path diversity expressed by such a routing record is always smaller than that of the king network. Then Knaive allows packets to traverse the network only in a subset of the minimal paths available on the underlying topology. In order to visualise this fact, observe Figure 5(a). It shows the path followed by packets traveling from a node (0,0) to another (3,0). Computing the routing record is trivial (3,0,0,0) and as can be seen, it represents a single path. The expression of the path diversity of the king topologies shows that the number of minimal paths between these two nodes is 7. So Knaive is ignoring most of the minimal paths connecting these two nodes. The same conclusion can be drawn from Figure 5(d), that shows a node (0,0) and the paths leading to node (5,2). Again the Knaive routing record (3,0,2,0) only gives a subset of the minimal paths available in the topology. In conclusion, Knaive would give good performance in uniform traffic, due to the balanced use of directions, yet it would show poor performance in adverse traffic, because it does not use all the path diversity.

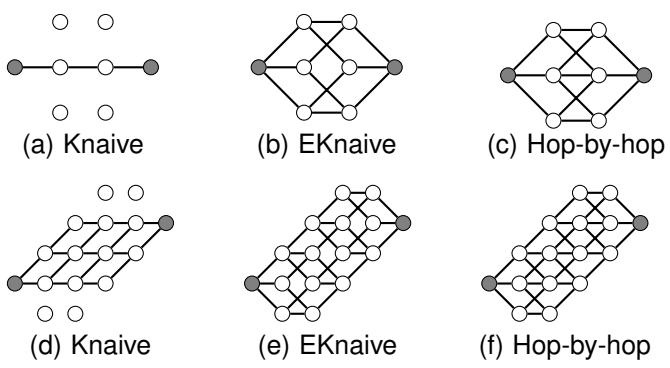


Fig. 5. Links used by the different paths connecting pairs of nodes, shown in gray, with different routing algorithms.

3.3.2 EKnaive

After considering the shortcomings of Knaive, it is necessary to devise a better routing algorithm. It should make use of more, if not all, the path diversity and show a better behavior in adverse traffic. This proposal is a refinement of Knaive and therefore named Enhanced Knaive, or *EKnaive*.

Observing Figures 5(a) and 5(d), it becomes apparent that Knaive is restricting the paths to a rhomboid defined by the source and destination nodes, with two sides in an orthogonal direction and the other two in a diagonal direction. Notwithstanding, several minimal paths lie outside of this rhomboid. In fact, all minimal paths are contained by a rectangle rotated 45 degrees with opposite vertices on the source and destination nodes. Thus EKnaive needs to make use of both diagonals in order to increase the path diversity. Note that a pair of orthogonal hops in the same direction can be converted to two diagonal hops in different directions. Using this rule, the orthogonal portion of a Knaive routing record can be converted to diagonal hops. As a consequence, the routing record will have at most three non-zero components, one orthogonal and two diagonals. The question remains as to how much of the orthogonal path is transformed to diagonals. It was shown in [33] that the best results were achieved when converting two thirds of the orthogonal component.

Of course, EKnaive gives more path diversity, as can be seen in Figures 5(b) and 5(e), but it also tends to use the diagonal links more than the orthogonal. This breaks the balance of Knaive and justifies the reduced performance on uniform traffic shown later, as forcing an increased use of diagonal links causes them to become the network bottlenecks. In addition, the full path diversity predicted in the topological analysis is not reached. Observe that there are unused orthogonal links in Figures 5(b) and 5(e). This means that a routing algorithm with better path diversity is still to be found.

But coming up with a routing record that covers all the path diversity of the king network is impossible. As two directions are linear combinations of the other two, the routing record can not express the variety of paths connecting two nodes. Any routing record can be transformed, using the above rule, into others with the same destination and length. But all of them will have at most three non-zero components and none will cover the whole path diversity. Remember that two orthogonal hops can be converted to diagonals, but the converse is not true. Two diagonals in the same direction can not be substituted by any number orthogonals resulting in a path with equal distance. Hence, using routing records with four non-zero components means that miss-routing is allowed. Some efforts were made in this direction by relaxing the minimum-distance constraint in [33].

3.3.3 Hop-by-hop routing

The algorithms presented above are all based on routing records computed at injection time, similar (but not

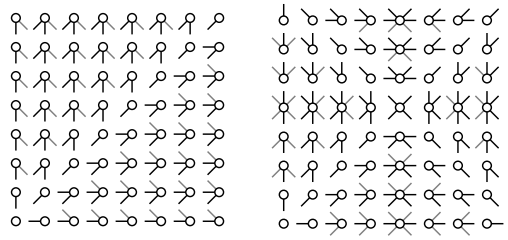
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2409865, IEEE Transactions on Parallel and Distributed Systems

the same) to a source routing mechanism. That is, the decision on which set of paths a packet can follow is predetermined at the source node. This set of paths has been expressed as a routing record, which is a vector holding the number of hops remaining in each direction to reach the destination. But it has also been shown that, because of the linear dependence of the different directions in the king networks, the routing record can not express the full variety of paths connecting any two nodes. Then a new approach is going to be adopted in order to exploit the topological richness of these networks.

This new approach abandons the concept of computing routing records at source and gives freedom to every router on the network upon how to direct the packets they receive. This is similar to the way packets are routed across the Internet. When a packet arrives at a router, the it reads the destination node from the packet header. And it decides to which neighbour it is sent or if it should be consumed. In every hop, the packet always gets closer to its destination, thus keeping the routing minimal.

In TCP/IP networks each router constructs a table to help it decide on the way to send packets. The composition of these tables is a resource consuming task, but because the topology of king networks is regular, a simple algorithm can be derived to calculate the profitable directions for a packet at each node, knowing the destination. This algorithm returns a vector with eight integers $(x_+, x_-, y_+, y_-, z_+, z_-, t_+, t_-)$ which are set to one if the corresponding port nears the packet to its destination. With this information the node can apply the Adaptive Bubble Routing mechanism easily. If the packet is going to be adaptively routed, a port is selected at random from those that are set to one in the vector. Whereas if the packet needs to use the escape channels then the first port set to one is selected. This ensures that DOR still governs the escape network.

Figure 6 shows the profitable port of each router of a king mesh, or king torus, when they receive a packet bound for node 0,0.



(a) King mesh (b) King torus

Fig. 6. Representation of the hop-by-hop vector at every node when destination is node (0,0) on the lower left corner.

In the routing algorithms based on routing records, the number of non-zero components of the routing record has been a concern, as it made a distinction between the naturally balanced Knaive and the increased path diversity of EKnaive. To make a similar analysis with the hop-by-hop algorithm, the vector integers are grouped in pairs, one pair for each direction. Then it can be seen that, like EKnaive it has at most three non-zero pairs. But as the vector is calculated in each node, it does not exclude any path and the full path diversity is attained. Observe that in Figures 5(c) and 5(f), all the links leading to the destination are drawn.

Notwithstanding, the benefit of using two non-zero components was that it could give the best results for uniform traffic as it perfectly balances network resources. But this is forsaken in the hop-by-hop algorithm, and shows a diminished performance under such traffic. However, if the way ports are selected could be prioritised, it could be possible to get the best of both worlds. This new scheme requires that the integers in the vector can take three values. Ports can be assigned zero if they are not profitable, one if they are profitable and are in a direction the Knaive would have chosen, lastly, they can be assigned two otherwise. A visualisation of this is shown in Figure 6 noting that the black lines would represent ones and the gray twos. Observe that only the diagonals can have the value two.

Now, the behaviour of the router is slightly different. When a packet arrives, the router checks the adaptive ports that have one in the vector. If all are full, it looks into the adaptive ports marked with two. Should these be also full, the escape port would be used, being the first one in the vector marked with one. In a way, this algorithm makes the decision about which adaptive channel is used in two steps. Therefore it is denoted 2S hop-by-hop. With this strategy packets tend to be routed with Knaive unless local congestion occurs, point at which it falls back to full path diversity routing. Thus this algorithm can give the best results, both in benign and adverse traffic patterns.

3.3.4 Broadcast Routing

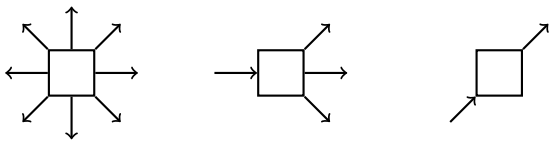
Many parallel applications, both in distributed and shared memory computers, rely on efficiently sending broadcast and multicast messages. In fact, the research for improving the performance of collective communications has received much attention in the last years [34]. Hence, a broadcast algorithm is presented. It relies on routers being able to forward packets through more than one port in one cycle. As the packets are sent under DOR discipline and ABR restrictions, the algorithm is deadlock safe.

To perform a broadcast, the source node sends the message to all its neighbours (Figure 7(a)). The message is marked with a broadcast flag and a time-to-live (TTL) integer. Then all nodes behave in the same fashion. If a broadcast packet is received, the TTL is decremented and consumed. If TTL is greater than 0, the packet is also sent through a number of other ports. Namely, if the broadcast packet was received from an orthogonal port, it is forwarded through the next neighbour in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2409865, IEEE Transactions on Parallel and Distributed Systems

the same direction plus two in the diagonal directions (Figure 7(b)). If the packet is received on a diagonal port, the packet is only forwarded to the next neighbours in the same direction (Figure 7(c)). Thus the TTL value defines the maximum area that the broadcast will reach. Obviously it must be set to a number that covers the whole network. In the case of king meshes, the TTL can be calculated as $max(x, s-x, y, s-y)$, where s is the side of the mesh and x, y are the coordinates of the source node. For king tori the TTL is the diameter.

A multicast could be constructed around the same idea. The source node would make a broadcast message including a list of the destination nodes. This list should be organised in the same way as the branches of the broadcast pattern. Then as the broadcast packet spreads and different copies of it take different paths, each copy would have the section of the destination list corresponding to its path. Destination nodes that receive the packet should consume it if they are on the destination list. And they should forward it, like the broadcast above, only if the list contains more addresses after removing its own from it. Thus the multicast would not send unnecessary packets to parts of the network where there are no destination nodes. Note that this scheme would not allow adaptive routing.



(a) Source node (b) Ortho. node (c) Diagonal node

Fig. 7. Behavior of nodes during broadcast.

4 EVALUATION

The previous Sections outline the topological properties of 2D networks with diagonals and propose routing algorithms to take advantage of them. For king networks in particular, there are several algorithms with different strengths and weaknesses. The following Section describes a set of experiments that corroborate the hypotheses of Section 2. Furthermore, it gives an experimental validation of the arguments used in Section 3 to develop the different routing algorithms. Lastly, it shows the performance of the different topologies and algorithms running real applications.

4.1 Experimental Setup

All the experiments carried out in this research have been done with the FSIN functional network simulator [35]. The router model was shown in Section 3.1. To be fair when comparing networks of different degree, a constant buffer space was assigned to each router and divided among all individual buffers. An initial evaluation has been performed with synthetic workloads using typical traffic patterns on 8×8 and 16×16 networks. These were applied on the networks by injecting packets of 16 phits at a constant rate, or load, measured in number of phits per cycle per node. To ensure the stability of the results, measurements were taken only when the network reached a steady state and averaged by simulating with five different random seeds. The increased degree of some topologies theoretically allows the throughput to rise above one phit per cycle per node. In order to take advantage of this, the simulations have been run with enough injectors per router to fully saturate the bisection bandwidth of the network. Some relevant simulation parameters is shown in Table 2.

Simulated cycles	50000
Network size	16×16, 8×8
Bidirectional links	yes
Virtual channels	2
Buffer space	32 phits per router
Injection queue	8 phits per queue
Flow control	Virtual cut-through + Bubble
Injection mode	shortest
Parallel injection	yes
Request mode	random
Arbiter	random
Packet length	8 phits

TABLE 2
Simulation settings.

FSIN is also capable of simulating networks using real application traces. These are injected preserving causal dependencies between messages and modeling computation time. Then, a second evaluation of the different topologies has been performed using traces of selected applications from the *NAS Parallel Benchmark(NPB)* suite [36]. This is typically employed in large parallel systems and is representative of general HPC applications running in those systems. To obtain the traces, the Extrae MPI tracing tool [37] was used while executing the suite with problem size A on 64 IBM JS21 blades. Due to the large length of the trace files, these were processed and trimmed to remove initialization and finalization, and leaving only the *region of interest*. This is the portion of the program's execution that contains all the communication events that are interesting to evaluate interconnection networks.

4.2 Experiments with synthetic traffic

The first experiment tries to validate the simulation infrastructure. This is done by comparing the experimental results with the theoretical expressions presented in Section 1. To this aim, Table 3 presents various performance metrics corresponding to toroidal topologies of 16×16 under uniform traffic. First, the theoretical average distance and the experimental minimum latency. The latter has been obtained by computing the average packet transmission times when the packets have one phit at minimum load. Under these conditions the latency and the average distance should be the same and, in fact,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2409865, IEEE Transactions on Parallel and Distributed Systems

the Table shows this similarity. On the other hand, the Table also presents the maximum throughput, both computed from the theoretical expressions and measured experimentally. To reach the throughput bound of each topology with multi-phit packets, the effect of the *Head-of-Line(HoL)* blocking must be minimised. Therefore a sufficient amount of virtual channels and injectors was used. It can be seen that the experimental results are close to the theoretical limits, however they are not reached due to the effect of the router architecture.

Network	T_{16}	DT_{16}	KT_{16}
Average distance	8	6.22	5.33
Minimum latency	8.13	6.34	5.48
Max. Throughput (Th)	0.5	1.0	1.5
Max. Throughput (Ex)	0.45	0.96	1.49

TABLE 3
Minimum latency and maximum throughput of toroidal networks under uniform traffic.

Section 3 called attention on the way the different routing algorithms use the channels in different directions under uniform traffic. It pointed out that a balanced use could show better performance. The next experiment shows the throughput of different algorithms running on king tori in Figure 8(a) and the average use of the different directions in Figure 8(b); a 16×16 torus is shown but results are similar with other sizes and topologies. It is noticeable that the two algorithms that have an equal use of all channels give better results than the others. Therefore, for the remainder of this article, king networks will be used only with Knaive and 2S Hop-by-hop algorithms.

Comparing the performance of different topologies is best done by observing the saturation throughput, and the evolution of the average latency as input load is increased from minimum to medium traffic load. Figures 9 and 10 show the average saturation throughput of the different topologies, network sizes and routing methods.

As it has been shown in Section 3, the Knaive algorithm perfectly balances the use of all directions but does not use the full path diversity available on the network. Notwithstanding, it obtains the best results because the combination of traffic in different directions makes an even use of all the network's links. On the other hand, the 2S hop-by-hop algorithm can use the full path diversity, but in a way that is not detrimental to the direction balance, if the traffic pattern is uniform. Therefore its performance is equivalent to that of Knaive.

In general, it can be seen that increasing the number of diagonals gives better results than the traditional 2D networks. The best overall performance is given by the 2S Hop-by-hop routing algorithm that is able to exploit the full path diversity of the king networks as can be seen in highly adverse traffic patterns like tornado or shuffle. It is noteworthy that the performance of networks with only one diagonal depends strongly on the mapping of the application. As when the main direction of the traffic matches the added diagonal, a better performance is observed. A good example of this behaviour is the transpose traffic on mesh networks or the tornado traffic. In these situations there is one diagonal networks that reaches the same performance as a king network, while the other hardly improves on the 2D network result.

On the other hand, an analysis of the average delay of packets can reveal interesting facts about the networks presented. Figure 11 shows these results for 8×8 meshes and tori. Those for 16×16 lead to the same conclusions and therefore have been omitted.

Observing the results for uniform traffic, the base latencies for small loads are slightly above the average distance of the topology plus the spooling latency, due to the packet length. This validates the expressions presented in Section 2 for the average distance. Then, adding diagonals significantly reduces the distance between nodes, and consequently the packet delay.

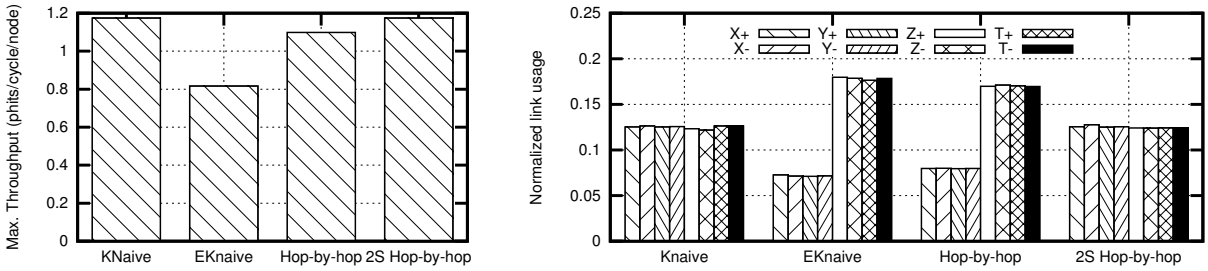
Looking at other traffic patterns, there are two extreme behaviours represented by the transpose and tornado patterns. The first obtained similar throughput in all topologies, this is a consequence of all having the same number of links crossing the diagonal of the network, therefore presenting the same bottleneck. But at low loads, as the different topologies have different average distances this fact is reflected in the base latency. The diagonal networks resemble the results of the 2D or king networks depending on the added diagonal, thus the aforementioned importance of the application mapping in these topologies.

The case of the tornado traffic is the opposite, the packets move mainly along orthogonal directions and the added diagonals do not reduce the distance among nodes. However they do add path diversity, so at high loads the traffic should spread throughout more paths and improve the maximum throughput. However this effect is only noticeable in the 2S Hop-by-hop algorithm, as it was shown in Section 3 that Knaive was not able to exploit the full path diversity.

4.3 Experiments with trace-driven simulations

The use of synthetic traffic allows early evaluation of network performance by continually stressing the network with well-known traffic patterns. However, real parallel applications typically combine communication phases, in which the network is used, with computation phases, where no network traffic is observed [38]. Therefore an analysis in a more realistic scenario is important to round up an evaluation of the topologies presented in this paper. The number of nodes used for this evaluation suggests tens or hundreds of processing units. Some experimental prototypes with large number of cores do not support cache coherence [39], [40]. Therefore, a set of experiments with traces from the NPB suite, that uses the message passing paradigm, have been carried out. The results presented in this paper only cover those benchmarks which have a significant network traffic, as these

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2409865, IEEE Transactions on Parallel and Distributed Systems



(a) Throughput (b) Direction usage

Fig. 8. Benefits of balanced use of different directions on 16×16 king tori.

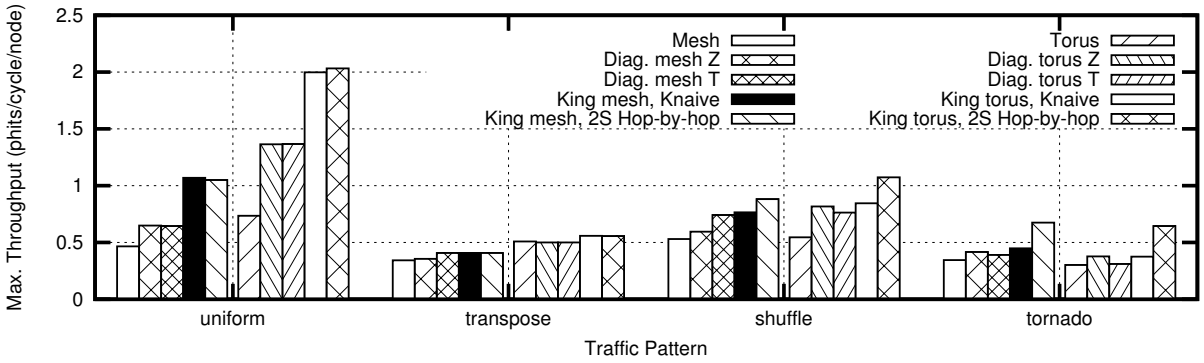


Fig. 9. Maximum throughput comparison for 8×8 networks.

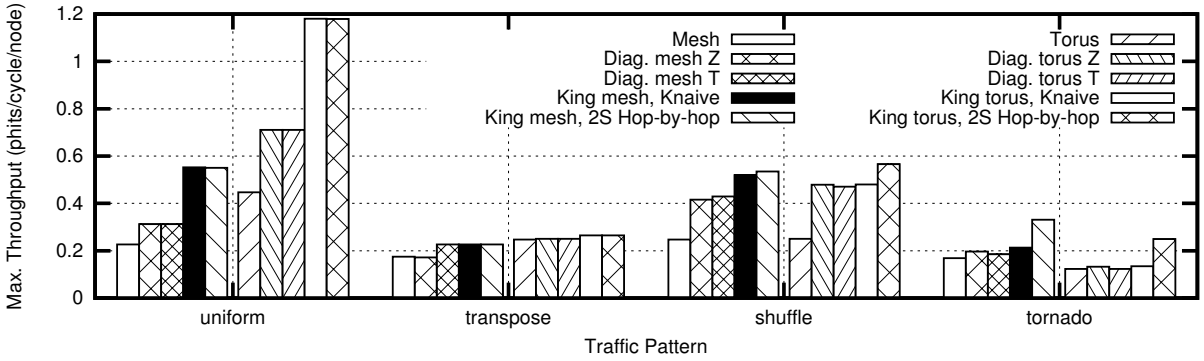


Fig. 10. Maximum throughput comparison for side 16×16 networks.

allow the study of the advantages and disadvantages of the different network topologies.

To present these results in an objective manner, and to emphasize the effect of the network, the simulation time for each topology is compared to the benchmark's ideal best time. Each trace was simulated with an ideal network that has infinite throughput and zero latency. This simulation time expresses the lower bound that can be aspired to when only the interconnection network is optimized. The graph in Figure 12 shows the slow-down exhibited by each topology and routing, computed as the ratio of the ideal time to the simulation time.

A first glance at the results shows that the best performance is given by the king topologies with the 2S hop-by-hop routing algorithm. This can be understood when considering the traffic generated by real applications. Instead of injecting packets at a constant rate with a given pattern, they show communication bursts with irregular use of network resources. As was seen with synthetic traffic the 2S hop-by-hop algorithm, which uses all the minimum distance paths between any pair of nodes, spreads the traffic among more network resources. Then, this kind of routing improves the traffic balance on the network. A direct consequence is that there is less congestion and thus running the trace needs less time.

Looking at the results in more detail, a series of behavioral patterns are observed. First, when running real applications, adding one diagonal to the mesh or torus gives some improvement. But which diagonal is added does not affect the performance. For instance

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2409865, IEEE Transactions on Parallel and Distributed Systems

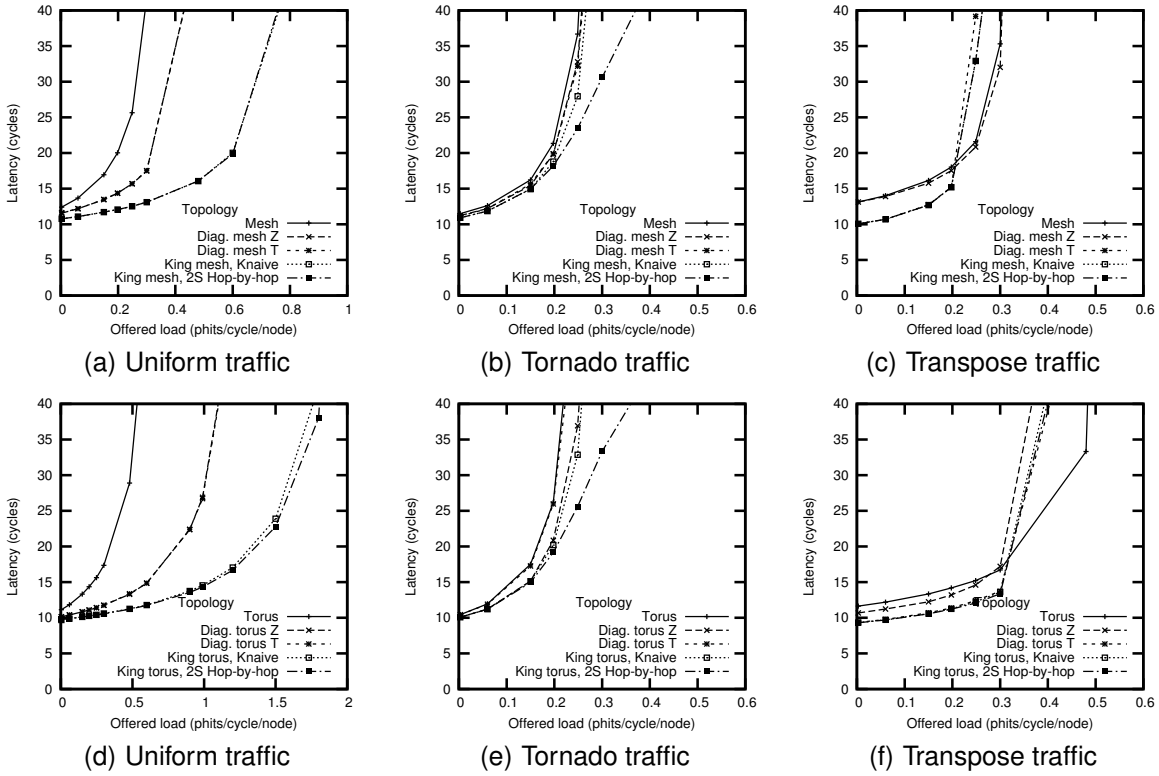


Fig. 11. Latency behavour in 8×8 meshes and tori.

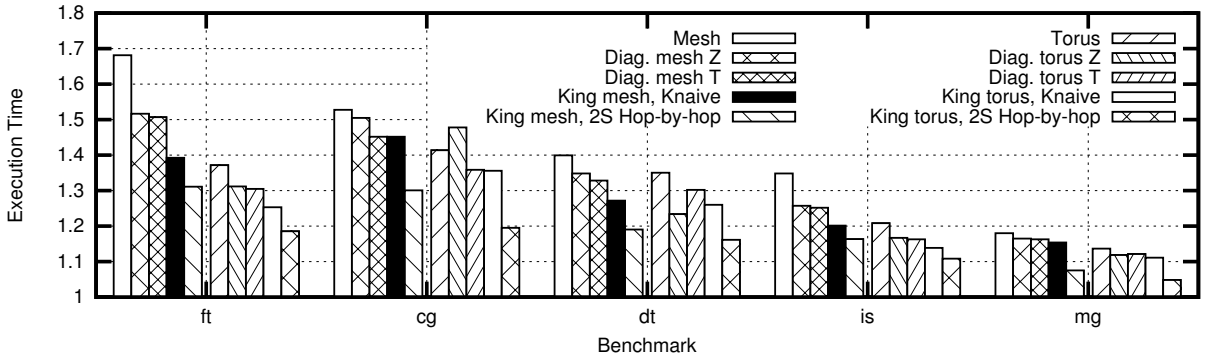


Fig. 12. Performance of 8×8 networks. Execution time is normalized to that of the ideal network.

benchmarks ft, is and mg both in meshes and tori. In all these cases adding a second diagonal, especially when using hop-by-hop routing, gives much better results.

Second, in some cases the election of the added diagonal is important, as was mentioned above. For instance, the performance of diagonal networks with benchmarks cg on meshes and dt is heavily dependent on which diagonal is chosen. One diagonal can give performance close to that of the king topologies with Knaive while the other is barely better than the 2D network. Therefore, when using diagonal networks the application mapping is particularly important. Focusing on king topologies, the performance difference between the Knaive and the 2S hop-by-hop routing points out the importance of the routing algorithm, as the use of a more sophisticated one compensates for the increased cost of these topologies.

Finally, the cg benchmark on tori shows that the addition of a diagonal can be detrimental to performance. This is due to the fact that cg has high communication between neighboring nodes on one diagonal. Then, as minimum routing is used, the number of paths connecting the nodes is reduced from two to one.

5 Conclusions

This paper presents an analysis of two topologies that, while having more ports per router than the traditional meshes or tori, still preserve most of their advantages, like their regularity or planar layout. First, a study of topologies that add diagonal links in one direction have been considered. But more importantly it proposes the novel king networks, that add diagonal links in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2409865, IEEE Transactions on Parallel and Distributed Systems

two directions. A topological analysis reveals that by duplicating the number of links, these networks not only double the bisection bandwidth but have three times more than their traditional 2D counterparts. The fact that the added links are diagonal, halves its diameter and reduces its average distance in 33%. Moreover, king topologies can be easily partitioned, that allows regular distribution of singular nodes, like specialized processing units or memory controllers. A folding scheme for the king torus has been also presented, which facilitates its implementation both as system or on-chip networks.

To benefit from all of the above, an appropriate routing algorithm must be used. Due to the regularity of the King topologies, finding a routing algorithm seems straightforward. In fact the first proposal of this paper is a simple, yet effective, arithmetic routing algorithm, named Knaive. However, as it does not exploit all the path diversity, it performs poorly under adverse traffic patterns. The paper proposes several solutions to solve this problem. The best is the 2S Hop-by-Hop where each node that receives a packet dynamically evaluates its profitable ports and decides which one is the best according to its destination and local traffic conditions.

The paper presents an extensive experimental evaluation of the different topologies and routing algorithms. Experimentation with synthetic load shows that, the performance of the topologies under uniform traffic practically reaches the theoretical limits. This happens when the effect of HoL blocking is reduced by increasing the number of virtual channels. The evaluation also reports a set of experiments with trace-driven simulations of various applications from the well known NPB suite. These experiments ultimately confirm that the king topologies perform significantly better than their standard counterparts. In the experiments the king mesh is always superior to the standard torus. Both networks employ two metal layers but king meshes could be preferable because they do not require folding and perfect partitioning is simple. Forthcoming work will include network evaluations under cache coherent traffic and energy consumption estimations.

REFERENCES

[1] D. Chen, N. A. Eisley, P. Heidelberger, R. M. Senger, Y. Sugawara, S. Kumar, V. Salapura, D. L. Satterfield, B. Steinmacher-Burow, and J. J. Parker, "The IBM Blue Gene/Q interconnection network and message unit," in *Int. Conf. for High Performance Computing, Networking, Storage and Analysis*. ACM, 2011, pp. 26–1.

[2] A. S. Bland, J. C. Wells, B. Messer, O. R. Hernandez, and J. H. Rogers, "Titan: Early experience with the Cray XK6 at Oak Ridge National Laboratory," *Cray User Group*, 2012.

[3] Y. Ajima, S. Sumimoto, and T. Shimizu, "Tofu: A 6D mesh/torus interconnect for exascale computers," *Computer*, vol. 42, no. 11, pp. 36–40, 2009.

[4] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-Driven, Highly-Scalable Dragonfly Topology," in *Int. Symp. on Computer Architecture*, 2008, p. 77.

[5] S. Scott, D. Abts, J. Kim, and W. J. Dally, "The BlackWidow High-Radix Clos Network," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 2, p. 16, 2006.

[6] G. Faanes, A. Bataineh, D. Roweth, E. Froese, B. Alverson, T. Johnson, J. Kopnick, M. Higgins, J. Reinhard *et al.*, "Cray cascade: a scalable HPC system based on a Dragonfly network," in *Int. Conf. on High Performance Computing, Networking, Storage and Analysis*, 2012, p. 103.

[7] S. Ramos and T. Hoefler, "Modeling Communication in Cache-coherent SMP Systems: A Case-study with Xeon Phi," in *Int. Symp. on High-performance Parallel and Distributed Computing*. New York, NY, USA: ACM, 2013, pp. 97–108.

[8] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C. C. Miao, J. F. B. III, and A. Agarwal, "On-Chip Interconnection Architecture of the Tile Processor," *IEEE Micro*, vol. 27, pp. 15–31, 2007.

[9] S. R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar, "An 80-Tile Sub-100-W TeraFLOPS Processor in 65-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, p. 29, 2008.

[10] N. E. Jerger, Z. Liu, and Z. Wang, "Leaving One Slot Empty: Flit Bubble Flow Control for Torus Cache-coherent NoCs," *IEEE Transactions on Computers*, vol. 99, no. PrePrints, p. 1, 2013.

[11] J. Kim, W. J. Dally, and D. Abts, "Flattened butterfly," in *Int. Symp. on Computer architecture*, 2007, p. 126.

[12] N. P. Carter, A. Agrawal, S. Borkar, R. Cledat, H. David, D. Dunning, J. Fryman, I. Ganev, R. A. Golliver, R. Knauerhase *et al.*, "Runnemede: An architecture for ubiquitous high-performance computing," in *Int. Symp. on High Performance Computer Architecture*, 2013, pp. 198–209.

[13] M. Igarashi, T. Mitsuhashi, A. Le, S. Kazi, Y.-T. Lin, A. Fujimura, and S. Teig, "A diagonal-interconnect architecture and its application to RISC core design," in *IEEE Int. Solid-State Circuits Conf.*, 2002, p. 210.

[14] A. Marshall, T. Stansfield, I. Kostarnov, J. Vuillemin, and B. Hutchings, "A reconfigurable arithmetic array for multimedia applications," in *Int. Symp. on Field Programmable Gate Arrays*, 1999, p. 135.

[15] K. Tang and S. Padubidri, "Diagonal and toroidal mesh networks," *IEEE Transactions on Computers*, vol. 43, no. 7, p. 815, 1994.

[16] K. Shin and G. Dykema, "A distributed I/O architecture for HARTS," in *Int. Symp. on Computer Architecture*, 1990, p. 332.

[17] W.-H. Hu, S. E. Lee, and N. Bagherzadeh, "DMesh: a Diagonally-Linked Mesh Network-on-Chip Architecture," *Int. Workshop on NoC Architectures (MICRO-41)*, 2008.

[18] I. Honkala and T. Laihonen, "Codes for Identification in the King Lattice," *Graphs and Combinatorics*, vol. 19, no. 4, p. 505, 2003.

[19] J. M. Camara, M. Moreto, E. Vallejo, R. Beivide, J. Miguel-Alonso, C. Martinez, and J. Navaridas, "Twisted Torus Topologies for Enhanced Interconnection Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 12, p. 1765, 2010.

[20] C. Martínez, E. Stafford, R. Beivide, and E. M. Gabidulin, "Modeling hexagonal constellations with Eisenstein-Jacobi graphs," *Problems of Information Transmission*, vol. 44, no. 1, p. 1, 2008.

[21] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., 2003.

[22] C. Martinez, E. Stafford, R. Beivide, C. Camarero, F. Vallejo, and E. Gabidulin, "Graph-based metrics over QAM constellations," in *IEEE Int. Symp. on Information Theory*, 2008, p. 2494.

[23] M. Flahive and B. Bose, "The Topology of Gaussian and Eisenstein-Jacobi Interconnection Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 8, p. 1132, 2010.

[24] R. Beivide, C. Martínez, C. Izu, J. Gutierrez, J. Á. Gregorio, and J. Miguel Alonso, "Chordal topologies for interconnection networks," in *High Performance Computing*, 2003, pp. 385–392.

[25] D. Abts, N. D. Enright Jerger, J. Kim, D. Gibson, and M. H. Lipasti, "Achieving predictable performance through better memory controller placement in many-core CMPs," *ACM SIGARCH Computer Architecture News*, vol. 37, no. 3, p. 451, 2009.

[26] 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*, vol. 7, no. 8, p. 841, 1996.

[27] V. Puente, R. Beivide, J. Gregorio, J. Prellezo, J. Duato, and C. Izu, "Adaptive bubble router: a design to improve performance in torus networks," in *Int. Conf. on Parallel Processing*, 1999, p. 58.

[28] V. Puente, C. Izu, R. Beivide, J. Gregorio, F. Vallejo, and J. Prellezo,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2409865, IEEE Transactions on Parallel and Distributed Systems

"The Adaptive Bubble Router," *Journal of Parallel and Distributed Computing*, vol. 61, no. 9, p. 1180, 2001.

[29] A. W. e. a. Roscoe, "Routing messages through networks: an exercise in deadlock avoidance," in *Proceedings of 7th occam User Group technical meeting, IOS BV, Amsterdam*, 1987.

[30] I. Cidon and Y. Ofek, "Fairness Algorithm for Full-duplex Buffer Insertion Ring," IBM Research, Tech. Rep. RC 14961, September 1989.

[31] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique," *Computer Networks*, vol. 3, no. 4, p. 267, 1979.

[32] N. R. Adiga *et al.*, "An Overview of the BlueGene/L Supercomputer," in *ACM/IEEE Conf. on Supercomputing*, 2002, p. 60.

[33] E. Stafford, J. L. Bosque, C. Martínez, F. Vallejo, R. Beivide, and C. Camarero, "A first approach to king topologies for on-chip networks," in *Euro-Par: Parallel Processing*. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 428–439.

[34] R. Thakur, R. Rabenseifner, and W. Gropp, "Optimization of collective communication operations in MPICH," *International Journal of High Performance Computing Applications*, vol. 19, no. 1, pp. 49–66, 2005.

[35] J. Navaridas, J. Miguel-Alonso, J. A. Pascual, and F. J. Ridruejo, "Simulating and evaluating interconnection networks with INSEE," *Simulation Modelling Practice and Theory*, vol. 19, no. 1, p. 494, 2011.

[36] D. H. Bailey *et al.*, "The NAS parallel benchmarks—summary and preliminary results," in *ACM/IEEE Conf. on Supercomputing*, 1991, p. 158.

[37] B. S. Centre, "Extrae MPI profiling tool," 2010. [Online]. Available: http://www.bsc.es/ssl /apps/performanceTools

[38] L. G. Valiant, "A bridging model for parallel computation," *Communications of the ACM*, vol. 33, no. 8, p. 103, 1990.

[39] I. Labs, "Single-Chip Cloud Computer," http://www.intel.com/content/www/us/en/research/intel-labs-single-chip-cloud-computer.html, Tech. Rep., 2014.

[40] S. Subramanian, "Ordered Mesh Network Interconnect (OMNI): design and implementation of in-network coherence," Ph.D. dissertation, Massachusetts Institute of Technology, 2013.

Esteban Stafford received the M. Sc. degree in Telecomunication Engineering from the University of Cantabria in 2001. He is currently a part-time professor in the Dept. of Computer Engineering and Electronics of the University of Cantabria. His research interests include computer architecture, parallel computers and interconnection networks.

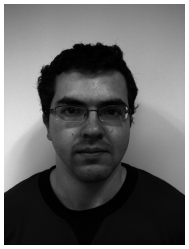
Jose Luis Bosque graduated in Computer Science and Engineering from Universidad Politécnica de Madrid in 1994. He received the PhD degree in Computer Science and Engineering in 2003 and the Extraordinary Ph.D Award from the same University. He has been an assistant professor at the Universidad de Alcalá, since 1995 to 1998 and Associated Professor at the Universidad Rey Juan Carlos, since 1998 to 2006. He joined the Universidad de Cantabria in 2006, where he is currently Associate Professor in the Department of Computer and Electronics. His research interests include high performance computing, heterogeneous systems and interconnection networks.

Carmen Martínez received the Ph.D. degree in Mathematic from the University of Cantabria (Spain) in 2007. Currently, she is an Associate Professor in the Dept. of Computer Engineering and Electronics of the University of Cantabria. Her interests include graph theory, coding theory and optimal topologies for interconnection networks.

Fernando Vallejo received his M.S. and Ph.D. degrees in physical science (electronics) from the University of Cantabria, Spain in 1985 and 1991, respectively. Since 1985, he has been with the Computer Engineering and Electronics Department, Universidad de Cantabria where he is currently the Department dean and Assistant Professor of computer architecture. His current research interests include parallel computers, interconnection systems, performance evaluation and new routing mechanism for fault-tolerance.

Ramón Beivide received the B.Sc. and M.Sc. degrees in Computer Science from the Universidad Autónoma de Barcelona (UAB) in 1981 and 1982, respectively, and the Ph.D. degree, also in Computer Science, from the Universidad Politécnica de Catalunya (UPC) in 1985.

He was an Associated Professor at UPC from 1987 to 1989 and at the Universidad del País Vasco from 1989 to 1991. He joined the Universidad de Cantabria in 1991, where he is currently a Professor in the Dept. of Computer Engineering and Electronics. His research interests include computer architecture, interconnection networks, coding theory and graph theory.



Cristóbal Camarero received the Master Degree in Computer Science (with distinction) from the University of Cantabria, Spain, in 2011. He is currently working towards the Ph.D. degree at the Electronics and Computers Department from the University of Cantabria. His research interests include graph theory with applications to interconnection networks and coding theory.

Emilio Castillo received the BS and MS degrees with honours in Computer Science from the University of Cantabria, Spain, where he is a PhD student. Previously he has worked as a Software Engineer at CERN in Switzerland and Hitachi Ltd. in Tokyo, Japan. He is interested in interconnection networks, FPGAs and multicore processors.