Lattice Graphs for High-Scale Interconnection Topologies

postprint version

C. Camarero, C. Martínez and R. Beivide, “Lattice Graphs for High-Scale Interconnection Topologies,” in IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 9, pp. 2506-2519, 1 Sept. 2015.

doi: 10.1109/TPDS.2014.2355827

Cristóbal Camarero, Carmen Martínez and Ramón Beivide
Computer Science and Electronics Department

Abstract

Torus networks of moderate degree have been widely used in the supercomputer industry. Tori are superb when used for executing applications that require near-neighbor communications. Nevertheless, they are not so good when dealing with global communications. Hence, typical 3D implementations have evolved to 5D networks, among other reasons, to reduce network distances. Most of these big systems are mixed-radix tori, which are not the best option for minimizing distances and efficiently using network resources. This paper is focused on improving the topological properties of this kind of networks.

By using integral matrices to deal with Cayley graphs over Abelian groups, we have been able to propose and analyze a family of high-dimensional mesh-based interconnection networks. As they are built over n-dimensional grids that induce a regular tiling of space, these topologies have been denoted *lattice graphs*. Higher dimensional networks can be composed over these graphs by means of a lift operation, which is also introduced in the paper. Easy network partitioning and minimal routing algorithm are also provided for these topologies based on this new network operation. Later we focus on cubic crystal lattices for modeling symmetric 3D networks and to show how lattice graphs can help in the design of twisted interconnection networks. In all cases, the networks obtained are better, in topological terms, than their standard tori counterparts. Finally, some practical issues such as implementability and preliminary performance evaluations have been addressed at the end of this work.

1 Introduction and Related Work

Interconnection networks are critical subsystems in modern supercomputers. Powerful supercomputers such as Cray XK7, IBM BluGene/Q and K computers use moderate degree networks. The Cray employs a 3D torus whereas BlueGene uses a 5D one [14, 12]. The K computer employs small 3D meshes (that can also be seen as 4×3 tori) connected by a bigger 3D torus [1]. All these topologies are mixed-radix tori, as they have dimensions of different sizes. For example, a configuration for a Cray Jaguar can be $25 \times 32 \times 16$ and a BlueGene configuration $16 \times 16 \times 16 \times 12 \times 2$. The $88,128$-node K computer installed at Riken, is compatible with a $17 \times 18 \times 24$ torus connecting 3D meshes of 12 nodes. Mixed-radix tori are not edge-symmetric, which can lead to unbalanced use of their network links. However, these big systems are typically divided into smaller partitions which enables them to be used by multiple users. Hence, providing symmetry, at least, in typical network partitions is an advisable design goal.

Tori are not well suited to support global and remote communications. Their relatively long paths among nodes, especially their diameter and average distance, incur high latencies and limited throughput. Thus, reducing topological distances in the network should be pursued. The way to achieve network distance reductions is by changing the topology. Topological changes depend on the router degree. If the router degree must be kept within moderate values, that is about 12, it would be interesting to preserve the good topological properties of tori such as grid locality, easy partitioning and simple routing. Hence, practicable topological changes should not be radical. A typical technique employed to this end has been twisting the wrap-around links of tori [3, 28, 4, 22]. Interestingly, this twisting also allows for edge-symmetric networks of sizes for which their corresponding tori are asymmetric [7, 9]. Twisting 2D tori is nearly as old as the history of supercomputers. The Illiac IV developed in 1971 already employed a twisted network. Many works dealing with twisted 2D tori have been published since then. However, when scaling dimensions, the problem of finding a good twisting scheme becomes harder. Very few solutions are known for 3D, with the one presented in [7] being a practicable example. Exploring the effect of twists in higher dimensions remains, to our knowledge, an unexplored domain. A target of this paper is to improve current topologies for moderate degree interconnection networks. By twisting tori links, distance properties are improved and graph symmetry can be enforced. Both topological parameters have impact on performance, as demonstrated in the Section VI. If the router degree can be increased, a radically different solution for reducing network diameter can be used in high-degree hierarchical networks [17]. These direct networks employing high-degree routers are beyond the scope of this paper.

It has been recognized for a long time that Cayley graphs are well suited to interconnection networks. Actually, the widely used rings and tori are Cayley graphs. Nowadays, rings are common in on-chip networks [26] and, as stated previously, tori dominate high-end supercomputing. In [18], Fiol introduced multidimensional circulant graphs as a new algebraic representation for Cayley graphs over Abelian groups. This representation has proved its suitability for studying and characterizing 2D mesh-based networks in [9]. In this paper, *lattice networks* are introduced as multidimensional circulants with orthonormal adjacencies, that is, multidimensional meshes plus additional wrap-around links that complete their regular adjacency. Therefore, this work is devoted to the study of high dimensional twisted tori topologies by means of lattice graphs. Special emphasis on the study of network upgrading and sub-network decompositions is done. Later, special attention will be devoted to symmetric 3D networks, which were completely characterized in [10]. Thus, the main contributions of this paper are:

- The proposal of lattice graphs as good models for interconnection twisted-tori-like networks.
- The introduction of the projecting operation of lattice networks to study embedded subnetworks.
- The definition of the lift and the common lift operations for building lattice networks of higher degree that embed other ones.
- A generic routing algorithm for lattice networks that comes from the projecting operation.
- A complete study of a subfamily of 3D symmetric networks, i.e. cubic crystal graphs.
- A first approach to practical issues such as implementability and a preliminary performance evaluation of these networks, which includes both topological models and empirical simulations.

The remainder of this paper is organized as follows. Section 2 defines lattice graphs. In Section 3 the concepts of graph lift, common lift and projection are given and some network examples are provided. Section 4 presents minimal routing algorithms for lattice networks. Section 5 focuses on 3D networks, describes symmetric lattice graphs and considers the particular family of the cubic crystal graphs. Then, the two previous methods for scaling lattice networks to higher dimensions are applied to crystal networks and some examples are presented. Section 6 discusses implementability and performance issues of cubic crystal networks. Finally, Section 7 concludes the paper summarizing its main findings.

2 Lattice Graphs

In this section we introduce *lattice graphs*, which will be used to model interconnection networks of any finite dimension. The lattice graph is not a new concept; in fact, it has different uses. In its most common use, which is also considered in this paper, is a graph built over an n-dimensional grid that induces a regular tiling of the space. In [18], multidimensional circulants were defined as lattice graphs but for any set of adjacencies (not only the orthonormal adjacencies leading to the grids considered in this work), which *a priori* seemed to be a wider family of graphs. However, it can be proved that any multidimensional circulant can be transformed into a lattice graph. Hence, the study presented in this section is devoted, in fact, to the family of Cayley graphs over finite Abelian groups [10].

Lattice graphs are defined over the integer lattice $\mathbb{Z}^n$. Hence, their nodes are labelled by means of n-dimensional (column) integral vectors. A lattice graph can be intuitively seen as a multidimensional grid with additional wrap-around links between opposite faces that complete its regular adjacency. Before proceeding with their formal definition, first we introduce some notation.

Notation 1. *The following notation will be used throughout the article:*

- *Lower case letters denote integers: a, b, ...*
- *Bold font denotes integer column vectors:* $\mathbf{v}$, $\mathbf{w}$, ...
- *Capitals correspond to integral matrices: M, P, ...*
- $\mathbf{e}_i$ *denotes the vector with a 1 in its i-th component and 0 otherwise.*

To define the finite set of nodes of these graphs and their wrap-around links, a modulo function using a square integer matrix will be used. Hence, congruences modulo matrices are introduced in the next definition.

Definition 2. *[18] Let $M \in \mathbb{Z}^{n\times n}$ be a non-singular square matrix of dimension n. Two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{Z}^n$ are congruent modulo M if and only if we have* $\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} \in \mathbb{Z}^n$ *such that:*

$$\mathbf{v} - \mathbf{w} = u_1\mathbf{m}_1 + u_2\mathbf{m}_2 + \cdots + u_n\mathbf{m}_n = M\mathbf{u}$$

where $\mathbf{m}_j$ denotes the j-th column of M. We will denote this congruence as $\mathbf{v} \equiv \mathbf{w} \pmod{M}$ and the congruence class of $\mathbf{v}$ by $(\mathbf{v} \pmod{M})$.

The set of nodes of a lattice graph will be the elements of the quotient group

$$\mathbb{Z}^n/M\mathbb{Z}^n = \{\mathbf{v} \pmod{M} \mid \mathbf{v} \in \mathbb{Z}^n\}$$

generated by the equivalence relation induced by M. As was proved in [18], $\mathbb{Z}^n/M\mathbb{Z}^n$ has $|\det(M)|$ elements. Now, we can proceed with a formal definition of a lattice graph.

Definition 3. *Given a square non-singular integral matrix $M \in \mathbb{Z}^{n\times n}$, we define the lattice graph generated by M as $\mathcal{G}(M)$, where:*

i) The vertex set is $\mathbb{Z}^n/M\mathbb{Z}^n = \{\mathbf{v} \pmod{M} \mid \mathbf{v} \in \mathbb{Z}^n\}$.

ii) Two nodes $\mathbf{v}$ and $\mathbf{w}$ are adjacent if and only if $\mathbf{v} - \mathbf{w} \equiv \pm\mathbf{e}_i \pmod{M}$ for some $i = 1, \ldots, n$.

From here onwards, all matrices will be considered to be non-singular, unless the contrary is stated. Note that, since $\mathbb{Z}^n/M\mathbb{Z}^n$ has $|\det(M)|$ elements, this will be the number of nodes of $\mathcal{G}(M)$. Moreover, since any vertex $\mathbf{v}$ is adjacent to $\mathbf{v} \pm \mathbf{e}_i \pmod{M}$, the lattice graph $\mathcal{G}(M)$ is, in general[1], regular of degree $2n$,

[1]Unless $\mathbf{e}_i \equiv \pm\mathbf{e}_j \pmod{M}$ or $2\mathbf{e}_i \equiv 0 \pmod{M}$ for some $i, j \in \{1, \ldots, n\}$.

that is, any node has $2n$ different neighbours. Next we show that the family of lattice graphs contains tori of any dimension.

Definition 4. *The n-dimensional torus graph of sides $a_1, \dots, a_n$, denoted by $T(a_1, \dots, a_n)$ is defined as a graph with vertices $\mathbf{x} \in \mathbb{Z}^n$ such that $0 \le x_i < a_i$. Two vertices $\mathbf{x}$ and $\mathbf{y}$ are adjacent if and only if they differ in exactly one coordinate, let us say i, for which $x_i \equiv y_i \pm 1 \pmod{a_i}$.*

Theorem 5. *The torus graph $T(a_1, \dots, a_n)$ is isomorphic to the lattice graph $\mathcal{G}(\operatorname{diag}(a_1, \dots, a_n))$, where $\operatorname{diag}(a_1, \dots, a_n)$ denotes the square diagonal matrix with diagonal equal to $a_1, \dots, a_n$.*

Proof. Clearly the vertex space of both graphs is the same. In the following we check that adjacencies are preserved. If $\mathbf{x}$ is connected to $\mathbf{y}$ in the torus then it holds that $\mathbf{y} - \mathbf{x} = (y_i - x_i)\mathbf{e}_i$ Then for some integer k, $\mathbf{y} - \mathbf{x} = (\pm 1 + k a_i)\mathbf{e}_i = \pm \mathbf{e}_i + k a_i \mathbf{e}_i = \pm \mathbf{e}_i + \operatorname{diag}(a_1, \dots, a_n) k \mathbf{e}_i$. Hence $\mathbf{y} - \mathbf{x} \equiv \pm \mathbf{e}_i \pmod{\operatorname{diag}(a_1, \dots, a_n)}$. □

Example 6. *Let us consider the circulant graph $C_{17}(1, 3, 7)$. This graph has as set of nodes the group $\mathbb{Z}_{17}$, and every node n is adjacent to the other six nodes $n \pm 1, \pm 3, \pm 7 \pmod{17}$. As it was shown in [18] every circulant graph is a multidimensional graph. Hence, this graph is the lattice graph generated by matrix*

$$\begin{pmatrix} 17 & 3 & 7 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

3 Projections and Lifts of Lattice Graphs

In this section, the concepts of *projection* and *lift* of a lattice graph will be stated. Projecting a lattice graph allows the study of the different lattice graphs of smaller dimensions that are embedded on it, while lifting a lattice graph will be used for increasing its dimension. In this aim, we next recall some known results from [18] about right-equivalent matrices.

Definition 7. *M_1 is right equivalent to M_2, which is denoted by $M_1 \cong M_2$, if and only if there exists a unitary matrix $P \in \mathbb{Z}^{n \times n}$ such that: $M_1 = M_2 P$.*

As was proved in [18], if $M_1 \cong M_2$ then the graphs $\mathcal{G}(M_1)$ and $\mathcal{G}(M_2)$ are isomorphic. Moreover, swapping two rows and changing the sign of one row also preserve isomorphism.

Now, performing Gaussian elimination by columns in a matrix is a right-equivalent operation. Therefore, after one step of Gaussian elimination in the generating matrix of a lattice graph gives isomorphic graphs. The resulting matrix would be:

$$M \cong \begin{pmatrix} B & \mathbf{c} \\ 0 & a \end{pmatrix}$$

where $B \in \mathbb{Z}^{n-1 \times n-1}$ is a matrix of smaller dimension, $\mathbf{c} \in \mathbb{Z}^{n-1}$ is a column vector and a is a positive integer. As a consequence, we obtain that $|\det(M)| = |\det(B)|a$, that is, the number of nodes of $\mathcal{G}(M)$ can be expressed in terms of $\mathcal{G}(B)$ and the integer a. Moreover, the lattice graph $\mathcal{G}(B)$ is isomorphic to the subgraph of $\mathcal{G}(M)$ generated by $\{\pm \mathbf{e}_1, \pm \mathbf{e}_2, \dots, \pm \mathbf{e}_{n-1}\}$, which allows us to state the following definition.

Definition 8. *Let $M \in \mathbb{Z}^{n \times n}$ be non-singular and $\mathcal{G}(M)$ be its lattice graph. Let us consider $M \cong \begin{pmatrix} B & \mathbf{c} \\ 0 & a \end{pmatrix}$ such that a is a positive integer. Then, we will say that a is the side of $\mathcal{G}(M)$ and $\mathcal{G}(B)$ its projection over $\mathbf{e}_n$. Moreover, we will call $\mathcal{G}(M)$ a lift of $\mathcal{G}(B)$.*

In particular, any lattice graph can be considered to be generated by its unique Hermite matrix, which may be convenient as Examples 10 and 11 attempt to show. Before stating the examples, we recall the Hermite normal form of a matrix.

Definition 9. *A matrix H is said to be in* Hermite normal form *if it is upper triangular, has positive diagonal and each $H_{i,j}$ with $j > i$ lies in a complete set of residues modulo $H_{i,i}$.*

Definitions 8 and 9 allow us to consider a helpful graphical visualization of any lattice graph that will also be used for routing in Section 4. First, lattice graphs and their subgraphs can be seen as n-dimensional spaces whose dimensions are sized by the elements in the principal diagonal of M. Each column vector in M represents a graph dimension, signaling the point in the space at which a new copy of the tile induced by M is located; this is important as column vectors dictate the pattern of the wrap-around connections of each dimension.

Moreover, from the cardinal equality $|\mathcal{G}(M)| = |\mathcal{G}(B)|a$, the lattice graph $\mathcal{G}(M)$ can be seen as composed of a disjoint copies of its projection $\mathcal{G}(B)$. One or several parallel cycles connect these disjoint copies completing the adjacency pattern. The length of these cycles can be computed as $ord(\mathbf{e}_n)$, which is the order of the element $\mathbf{e}_n$ in the group $\mathbb{Z}^n/M\mathbb{Z}^n$. According to [18], the order of any element $\mathbf{x}$ can be computed as

$$ord(\mathbf{x}) = \frac{\det(M)}{\gcd(\det(M), \gcd(\det(M)M^{-1}\mathbf{x}))}.$$

Note that the second gcd (greatest common divisor) in the fraction corresponds to the gcd of the elements of a vector. The number of vertices of each cycle lying in each copy of $\mathcal{G}(B)$ can be calculated as the length of the cycle over the side of the graph, that is $\frac{ord(\mathbf{e}_n)}{a}$.

Example 10. *Let us consider the rectangular twisted torus of size $2a \times a$ and twist a, denoted as $RTT(a)$ in [7]. A graphical representation of $RTT(4)$ can be seen in Figure 1. This graph is generated by the matrix $H = \begin{pmatrix} 2a & a \\ 0 & a \end{pmatrix}$. Using H, the graph can be seen as a mesh of $2a \times a$ ($h_{1,1} \times h_{2,2}$). In the previous representation, wrap-around links in $\mathbf{e}_1$ (first) dimension conserve their horizontality since $h_{2,1} = 0$; wrap-around links in $\mathbf{e}_2$ (second) dimension do not conserve their verticality but suffer a twist of a columns since $h_{1,2} = a$. According to Definition 8, the projection over $\mathbf{e_2}$ of $RTT(a)$ is a cycle of $2a$ nodes. As the side of $RTT(a)$ is a, it will have a disjoint cycles of $2a$ nodes. As $ord(\mathbf{e}_2)$ (the element representing a jump in $\mathbf{e}_2$ dimension) is $2a$, the graph will have a parallel cycles of length $2a$ in that dimension. Each of these a cycles contains two vertices of each projection.*

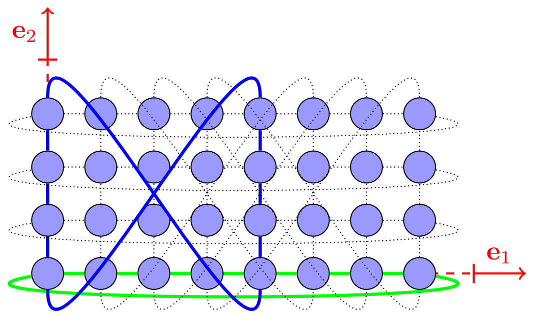

Figure 1: Two perpendicular cycles of length 8 in the $RTT(4)$.

Example 11. *Let us now consider the lattice graph $\mathcal{G}(M)$ with $M = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 4 & 2 \\ 0 & 0 & 4 \end{pmatrix}$. Note that M is in Hermite form. $\mathcal{G}(M)$ can be seen as a $4 \times 4 \times 4$ cubic grid. Three sets of wrap-around links, each one connecting opposite faces, have to be added to the grid-based cube. Wrap-around links in $\mathbf{e}_1$ always remain horizontal by construction, as imposed by the $n-1$ zeros in the first column vector of any Hermite matrix. Wrap-around links in the $\mathbf{e}_2$ dimension remain vertical in this graph because $m_{1,2} = 0$ but, in general, they can undergo only a twist over the $\mathbf{e}_1$ dimension of $m_{1,2}$ units. Finally, wrap-around links in the $\mathbf{e}_3$ dimension can undergo twists over both $\mathbf{e}_1$ and $\mathbf{e}_2$ dimensions. In the graph of this example, no twist is applied in $\mathbf{e}_3$ over $\mathbf{e}_1$ because $m_{1,3} = 0$ and a twist of 2 units is applied over the $\mathbf{e}_2$ dimension as $m_{2,3} = 2$. As can be seen in Figure 2, the projection of $\mathcal{G}(M)$ is $\mathcal{G}\begin{pmatrix} 4 & 0 \\ 0 & 4 \end{pmatrix}$, a 2D torus $T(4,4)$. Thus, the graph is composed of 4 disjoint copies of its projection, each of them connected by a cycle of length 8, as represented in the figure. Note that for every vertex in the graph there will be a similar cycle with the same pattern as the one represented in the figure. The cycle intersects in two vertices with each copy of the projection. For the sake of the clarity, only one cycle between copies $\mathcal{G}\begin{pmatrix} 4 & 0 \\ 0 & 4 \end{pmatrix}$ has been represented.*

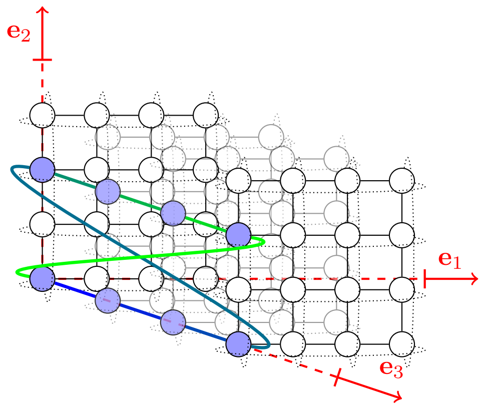

Figure 2: The cycle $\langle \mathbf{e}_3 \rangle$ joining the disjoint copies of the projection.

Note that we can project over any $\mathbf{e}_i$, simply by swapping rows i and n (which gives an isomorphic graph) and then, project over $\mathbf{e}_n$. Moreover, as we will see later, symmetries will make irrelevant over which dimension we project, so we will consider $\mathbf{e}_n$ by default. The resulting projection can again be projected over another vector, which results in a projection over a plane of the lattice graph. Clearly, projecting over a pair of vectors $\{\mathbf{e}_i, \mathbf{e}_j\}$ can be done in any order, since projecting first over $\mathbf{e}_i$ and then over $\mathbf{e}_j$ results in the same graph as projecting first over $\mathbf{e}_j$ and then over $\mathbf{e}_i$. Following the same idea, we can project over several dimensions iteratively. Therefore, we will call the result of projecting iteratively over the vectors in the set $\{\mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_r}\}$ the *projection* of $\mathcal{G}(M)$ *over the set*. In this case we will call it an r-dimensional projection which turns into a lattice graph generated by a $(n-r) \times (n-r)$ matrix.

Now, we consider a new way of lifting lattice graphs. In this new operation, given two lattice graphs we will look for another one which has them as projections but minimizing the resulting degree.

Definition 12. *The lattice graph $\mathcal{G}(M)$ is a common lift of $\mathcal{G}(M_1)$ and $\mathcal{G}(M_2)$ if both can be obtained as projections of $\mathcal{G}(M)$.*

Remark 13. *There are several ways of obtaining different common lifts of two given lattice graphs. A straightforward one is to consider the lattice graph $\mathcal{G}(M_1 \oplus M_2)$ generated by the direct sum of the matrices. As we state next, this option leads to the Cartesian product of the two given lattice graphs.*

Lemma 14. *$\mathcal{G}(M_1 \oplus M_2)$ is a common lift of $\mathcal{G}(M_1)$ and $\mathcal{G}(M_2)$ and $\mathcal{G}(M_1 \oplus M_2) \cong \mathcal{G}(M_1) \times \mathcal{G}(M_2)$, which denotes the Cartesian product of $\mathcal{G}(M_1)$ and $\mathcal{G}(M_2)$.*

As we will see next, there exist other common lifts that obtain $\mathcal{G}(M_1)$ and $\mathcal{G}(M_2)$ as projections but generating a lattice graph of smaller dimension. Note that this would be beneficial for cost aspects, such as minimizing the degree of the network routers, and to provide a good relation between the size of the graph and its projections.

Theorem 15. *Given two lattice graphs $\mathcal{G}(M_1)$ and $\mathcal{G}(M_2)$, we consider the lattice graph $\mathcal{G}(M_1 \boxplus M_2)$ that is obtained as follows: Let $M_1 \cong H_1$ and $M_2 \cong H_2$ with H_1 and H_2 in Hermite normal form. Let C be the submatrix with the first common columns of H_1 and H_2. Then $H_1 = \begin{pmatrix} C & R_A \\ 0 & A \end{pmatrix}$ and $H_2 = \begin{pmatrix} C & R_B \\ 0 & B \end{pmatrix}$, where A and B are square matrices. Then*

$$M_1 \boxplus M_2 = \begin{pmatrix} C & R_A & R_B \\ 0 & A & 0 \\ 0 & 0 & B \end{pmatrix}$$

It is obtained that:

i) *$\mathcal{G}(M_1 \boxplus M_2)$ is a common lift of $\mathcal{G}(M_1)$ and $\mathcal{G}(M_2)$*

ii) $\max(dim(\mathcal{G}(M_1)), dim(\mathcal{G}(M_2))) \leq dim(\mathcal{G}(M_1 \boxplus M_2)) \leq dim(\mathcal{G}(M_1 \oplus M_2))$

Proof. The first item is obtained by construction. For the second one, consider $\max(dim(\mathcal{G}(M_1)), dim(\mathcal{G}(M_2))) \leq \dim(\mathcal{G}(M_1 \boxplus M_2)) = dim(\mathcal{G}(M_1)) + dim(\mathcal{G}(M_2)) - dim(\mathcal{G}(C)) \leq dim(\mathcal{G}(M_1)) + dim(\mathcal{G}(M_2)) = dim(\mathcal{G}(M_1 \oplus M_2))$ □

Note that when the matrices M_1 and M_2 have no common columns, both $\mathcal{G}(M_1 \boxplus M_2)$ and $\mathcal{G}(M_1 \oplus M_2)$ coincide. Moreover, by construction, the operation $\mathcal{G}(M_1 \boxplus M_2)$ provides a lift that minimizes its dimension. Although in upcoming sections several examples will be considered, the next one tries to clarify this definition.

Example 16. *Let us consider* $M_1 = \begin{pmatrix} 2a & 0 \\ 0 & 2a \end{pmatrix}$ *and* $M_2 = \begin{pmatrix} 2a & a \\ 0 & a \end{pmatrix}$. *Clearly,* $\mathcal{G}(M_1)$ *is the 2D-torus of side* $2a$ *and* $\mathcal{G}(M_2)$ *the RTT. Then if we consider* $M_1 \boxplus M_2 = \begin{pmatrix} 2a & 0 & a \\ 0 & 2a & 0 \\ 0 & 0 & a \end{pmatrix}$, *the resulting is a lattice graph of degree 6 having both graphs as its projections.*

4 Routing in Lattice Graphs

Most interconnection networks use routing tables but their size can compromise system scalability. In this section routing algorithms for lattice graphs are presented. In this way, algorithmic routing can be used to avoid the need of tables. If tables are to be used, the algorithms presented can be employed to fill them.

Routing in circulant graphs was first related to the Closest Vector Problem (CVP) in [6] for the l_1-norm. Later, this fact was used to optimize a routing algorithm for circulants of degree four in [20]. Following the same ideas, similar complexity for the CVP can be inferred for routing in lattice graphs. As proved in [15] and [16], CVP can be solved with asymptotical complexity $2^{O(n)}$. However, algorithms for particular graphs can be improved. In this subsection we consider how to appropriately choose the projection of the lattice graph in order to obtain the best routing algorithm among all the possibilities.

Our routing algorithm is based on the hierarchy induced by the projecting operation. Routing in a lattice graph can be done by routing in its projection and in the ring defined by its side. First, we state the node labelling adopted and present a hierarchical routing. Then, we establish the general routing algorithm. Finally, complexity and implementation aspects are considered.

For solving the routing problem over lattice graphs we need first to state which labelling set will to be applied. A labelling set is the set that contains the labels for the vertices of the graph. There are many choices for the labelling set. In the 2D case, several approaches to the routing problem have been made in [19, 27, 9]. In those articles, several labellings such as the one given by the fundamental parallelogram of the lattice, the set of integers modulo N or the set of minimum norm residues have been considered. Anyway, for labelling a lattice graph of dimension n, a subset of $\mathbb{Z}^n$ will be needed. In particular, we define it as follows.

Definition 17. *Given a lattice graph* $\mathcal{G}(M)$ *of dimension* n *a labelling set of the graph is* $\mathcal{L} \subset \mathbb{Z}^n$ *such that* $|\mathcal{L}| = |\det(M)|$ *and for every pair* $\mathbf{l}_1, \mathbf{l}_2 \in \mathcal{L}$ *we have* $\mathbf{l}_1 \not\equiv \mathbf{l}_2 \pmod{M}$.

If $\mathbf{v}_s, \mathbf{v}_d \in \mathcal{L}$, where $\mathbf{v}_s$ labels the source node and $\mathbf{v}_d$ labels the destination node, we will call any vector $\mathbf{r} \in \mathbb{Z}^n$ a **routing record** when

$$\mathbf{v}_d - \mathbf{v}_s \equiv \mathbf{r} \pmod{M}$$

with $\mathbf{v}_d - \mathbf{v}_s \in \mathbb{Z}^n$ such that:

$$\mathbf{v}_d - \mathbf{v}_s \in \mathcal{L} - \mathcal{L} = \{\mathbf{x} - \mathbf{y} \mid \mathbf{x}, \mathbf{y} \in \mathcal{L}\}.$$

From a design perspective, it is convenient to label the graph nodes according to their positive coordinates. Hence, we will consider the labelling given by the Hermite normal form of the generating matrix. Therefore, let us assume that H is the Hermite normal form of M and

$$\mathcal{L} = \{\mathbf{x} \in \mathbb{Z}^n \mid 0 \le x_i < H_{i,i}\}.$$

The differences set that will be the input for any of the considered routing algorithms will be:

$$\mathcal{L} - \mathcal{L} = \{\mathbf{x} \mid -H_{i,i} < x_i < H_{i,i}\}.$$

Each component of a routing record indicates the number of hops in the corresponding dimension and its sign, the direction of the hops. The length of a path associated with a routing record is given by its l_1-*norm*:

$$|\mathbf{r}| = \sum_i |r_i|$$

As minimal routing requires shortest paths, minimum norm routing records should be obtained. Hence, the **routing problem** over $\mathcal{G}(M)$ can be stated as follows:

Algorithm 1: Hierarchical Routing in Lattice Graphs

```
Input: $\mathbf{v}_s$ source, $\mathbf{v}_d$ destination
Output: $\mathbf{r}$ minimum routing record from $\mathbf{v}_s$ to $\mathbf{v}_d$
Let $y$ be the last component of $\mathbf{v}_d$;
$\mathbf{v}_s + \mathcal{C}$ is the cycle translated to $\mathbf{v}_s$;
foreach vertex $\mathbf{c}_i$ of the cycle in the copy $[\mathcal{G}(B)]_y$ do
    $r_i^{\mathcal{C}}$: Route in the cycle from $\mathbf{v}_s$ to vertex $\mathbf{c}_i$;
    $\mathbf{r}_i^{\mathcal{G}(B)}$: Route in $[\mathcal{G}(B)]_y$ from $\mathbf{c}_i$ to $\mathbf{v}_d$;
end
Return the routing record that minimizes the weight of $\begin{pmatrix}\mathbf{r}_i^{\mathcal{G}(B)}\\ r_i^{\mathcal{C}}\end{pmatrix}$;
```

$$\begin{array}{rl}\textbf{input}: & \mathbf{v}{:=}\mathbf{v}_d - \mathbf{v}_s \in \mathcal{L} - \mathcal{L}\\ \textbf{output}: & \underset{\mathbf{r}\equiv\mathbf{v} \pmod M}{\operatorname{argmin}} \ (|\mathbf{r}|)\end{array}$$

where argmin states for the element in the set $\{\mathbf{r} \in \mathbb{Z}^n \mid \mathbf{r} \equiv \mathbf{v} \pmod M\}$ minimizing $|\mathbf{r}|$.

Our routing approach takes advantage of the hierarchical nature of lattice graphs. The idea is that routing in a lifted graph can be done by routing in its projection and in the cycle that joins the disjoint projections. Remember that the lattice graph $\mathcal{G}(M)$ with $M \cong \begin{pmatrix} B & \mathbf{c}\\ 0 & a\end{pmatrix}$ has a disjoint copies of its projection $\mathcal{G}(B)$ embedded, which are connected by $\frac{|\det(M)|}{ord(\mathbf{e}_n)}$ parallel cycles. The cycles have length $ord(\mathbf{e}_n)$. The number of vertices belonging to a cycle that lies in the same copy of $\mathcal{G}(B)$ is $\frac{ord(\mathbf{e}_n)}{a}$. Hence, we can separately consider the elements of the routing record in the following way:

Proposition 18. *Let $M \cong \begin{pmatrix} B & \mathbf{c}\\ 0 & a\end{pmatrix}$. Then, if $\mathcal{L}_M$ denotes the labelling set $\mathcal{G}(M)$ and $\mathcal{L}_B$ denotes the labelling set of its projection $\mathcal{G}(B)$ we deduce that*

$$\mathcal{L}_M = \left\{ \begin{pmatrix}\mathbf{x}\\ y\end{pmatrix} \mid \mathbf{x} \in \mathcal{L}_B, 0 \le y < a \right\}.$$

Now, we can state the following main result:

Theorem 19. *If $[\mathcal{G}(B)]_y$ is the projection $\mathcal{G}(B)$ of $\mathcal{G}(M)$ that contains $y\mathbf{e}_n$, $\mathcal{C}$ denotes the cycle generated by $\mathbf{e}_n$ and, given a vertex $\mathbf{v} \in \mathbb{Z}^n$, $\mathbf{v} + \mathcal{C}$ denotes the translation of the cycle to this vertex. Algorithm 1 gives minimum routing records in any lattice graph.*

Proof. Since the algorithm composes routing records from two subgraphs, then the result is indeed a routing record. We need to see that a minimum one is found.

Let $\mathbf{r}^{\min}$ be one of the routing records with minimum norm. Since $\mathbf{v}_s + \mathbf{r}_n^{\min}$ is in the cycle mentioned in the algorithm, then there is an index i such that $\mathbf{r}_n^{\min}$ is the minimum route in the cycle from $\mathbf{v}_s$ to $\mathbf{c}_i$. As $\mathbf{r}^{\min}$ is minimal, we find that the minimal routing from $\mathbf{c}_i$ to $\mathbf{v}_d$ does not use the n dimension. Thus, routing in $[\mathcal{G}(B)]_y$ gives the minimum. By composing both, the algorithm finds the minimum routing $\mathbf{r}^{\min}$ and returns it or another one with same norm. □

Remark 20. *In the last step of Algorithm 1 there can sometimes be several routing records with the same weight. In this case it is advisable to choose one of them at random, thus balancing the use of the paths.*

Remark 21. *Let us assume the lattice graph $\mathcal{G}(M)$ with $M \cong \begin{pmatrix} B & \mathbf{c}\\ 0 & a\end{pmatrix}$. Clearly, the complexity of Algorithm 1 is $O(C\frac{ord(\mathbf{e_n})}{a})$, where C denotes the complexity of routing in $[\mathcal{G}(B)]_y$. If routing is done with the same algorithm by means of recursive calls, the final complexity would be $O(\prod_{i=1}^n \frac{ord(\mathbf{e_i}, M_i)}{a_i})$, where M_i are the successive projections, a_i denotes the side of $\mathcal{G}(M_i)$ and $ord(\mathbf{e_i}, M_i)$ the order of $\mathbf{e_i}$ in $\mathcal{G}(M_i)$. In the worst case, this complexity would attain $O(\det(M)^n)$. However, in some families the order of $\mathbf{e_i}$ is upper bounded, thus obtaining good complexities for this recursive version of the routing algorithm, as it will be seen in the following section.*

5 Symmetric Lattice Graphs

Symmetry is a desirable property for any network as it impacts on performance and routing efficiency. Many interconnection networks have been based on vertex-symmetric graphs, but less attention has been devoted to edge-symmetric networks. Square and cubic tori have been the networks of choice for many designs as they are symmetric (vertex and edge symmetric). For this reason, symmetric lattice graphs will be considered in this section. Hence, we next introduce the concept of a symmetric graph.

A graph $G = (V, E)$ is *vertex-symmetric* (or *vertex-transitive*) if for each pair of vertices $(x, y) \in V$, there is an automorphism ϕ of G such that $\phi(x) = y$. Also, G is *edge-symmetric* (or *edge-transitive*) if for each pair of edges $(\{x_1, x_2\}, \{y_1, y_2\}) \in E$, there is an automorphism ϕ of G such that $\phi(\{x_1, x_2\}) = \{\phi(x_1), \phi(x_2)\} = \{y_1, y_2\}$. Finally, G is said to be *symmetric* when it is both vertex-symmetric and edge-symmetric. Since every Cayley graph is vertex-symmetric [2], we will focus on edge-symmetry. Clearly, vertex-symmetry implies that all vertices have the same degree. As was shown in [8], the consideration of non-linear automorphisms in the edge-symmetry characterization leads to marginal families of graphs that do no exemplify the general behaviour. Hence, in this paper we will refer only to automorphisms that are linear applications, that is, $\phi(x + y) = \phi(x) + \phi(y)$. Therefore, in an abuse of notation, symmetric graphs will refer to those in which there exist a linear automorphism fulfilling the previous definition. The group of automorphisms is denoted by $Aut(\mathcal{G}(M))$. Next, an interesting result about symmetric graphs is stated.

Theorem 22. *The projections of a symmetric lattice graph are all isomorphic.*

Proof. Let us denote $proj_i(\mathcal{G}(M))$ to be the projection of $\mathcal{G}(M)$ over $\mathbf{e}_i$ and $\mathcal{B}_n$ the n-dimensional orthonormal basis. We know $proj_i(\mathcal{G}(M))$ is isomorphic to the subgraph of $\mathcal{G}(M)$ generated by $\mathcal{B}_n \setminus \{\mathbf{e}_i\}$. As $\mathcal{G}(M)$ is symmetric we know $\phi \in Aut(\mathcal{G}(M))$ such that $\phi(\mathbf{e}_i) = \pm\mathbf{e}_j$. As $\mathbf{e}_i$ is the only generator not in $proj_i(\mathcal{G}(M))$, $\mathbf{e}_j$ is the only generator not in $\phi(proj_i(\mathcal{G}(M)))$. Hence, as ϕ is an automorphism, we deduce that $proj_i(\mathcal{G}(M)) \cong proj_j(\mathcal{G}(M))$. □

Now, we concentrate on 3D symmetric graphs. In [10] it is proved that the only symmetric 3D lattice graphs are the ones given by the matrices described in the next result.

Theorem 23. *Let $M \in \mathbb{Z}^{3\times 3}$. Then, the lattice graph $\mathcal{G}(M)$ is symmetric if and only if it is isomorphic to $\mathcal{G}(M')$ where:*

$$M' \in \left\{ \begin{pmatrix} a & c & b \\ b & a & c \\ c & b & a \end{pmatrix}, \begin{pmatrix} a & b & c \\ a & c & -b-c \\ a & -b-c & b \end{pmatrix} \right\}.$$

The previous characterization gives us a broad family of symmetric graphs. Note that the side of matrix $\begin{pmatrix} a & c & b \\ b & a & c \\ c & b & a \end{pmatrix}$ is $\gcd(a, b, c)$. Thus, maximizing the side implies $b, c \in \{0, a\}$. This is exactly the case of cubic crystal lattices [21], which are:

- **Primitive Cubic Lattice:** $\begin{pmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{pmatrix}$.
- **Face-centered Cubic Lattice:** $\begin{pmatrix} a & a & 0 \\ a & 0 & a \\ 0 & a & a \end{pmatrix}$.
- **Body-centered Cubic Lattice:** $\begin{pmatrix} -a & a & a \\ a & -a & a \\ a & a & -a \end{pmatrix}$.

In the rest of the paper we will concentrate on cubic crystal lattice graphs for three major reasons. As we will see next, these lattice graphs have all their projections also symmetric. Moreover, the routing algorithm in these cases is optimal since the side is maximized, thus minimizing the recursive call in the hierarchical algorithm. We also select this family of 3D symmetric lattice graphs to exemplify how the previously introduced graph operations can be applied to construct a wide variety of new topologies for interconnection networks.

In the following two subsections we will consider the lattice graphs defined by cubic crystal lattices, their isomorphisms with previously studied network topologies and a comparison among them in terms of their distance properties. Once we have characterized 3D symmetric topologies and detailed the special case of the cubic crystal graphs we will consider their upgrading process. As we have asserted before, symmetry could help when the application runs on the whole network. However, in big systems the user typically only has a partition of the complete machine assigned. Therefore, looking for symmetry in higher dimensions cannot be prioritized. Nevertheless, reducing the distance properties of the whole network would be still beneficial since applications and system software sometimes run over the entire network. Consequently, what we look for are higher dimensional networks embedding the previous crystal cubic lattice graphs. Therefore, two more subsections are included in which we explore two different methods for upgrading cubic crystal lattice graphs. The first one is to consider the lifting of crystal graphs, which results in 4D topologies. Whenever possible, the lift is done in such a way that the resulting eight-degree topology preserves symmetry. We will introduce a tree that represents the process of network upgrading, preserving symmetry. The second one presents the common lifts of lattice graphs and the resulting *hybrid graphs*, since several lattice graphs of different nature (symmetric or non-symmetric) and degrees are embedded on them. The section finishes with a routing discussion subsection.

5.1 Cubic Crystal Lattice Graphs

We define the **Primitive Cubic Lattice Graph** $PC(a)$ as the lattice graph generated by the matrix associated with the primitive cubic lattice, that is:

$$\begin{pmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{pmatrix}.$$

Clearly, the number of nodes of the graph is a^3, which is the determinant of the diagonal matrix. According to Theorem 5, $PC(a)$ is isomorphic to the 3D torus of side a, or equivalently, the a-ary 3-cube.

Lemma 24. *The projection of $PC(a)$ is the 2D torus graph of side a or $\mathcal{G}(\begin{pmatrix} a & 0 \\ 0 & a \end{pmatrix})$.*

The **Face-centered Cubic lattice graph** $FCC(a)$ of side a can be defined as the lattice graph generated by the matrix associated with the face-centered cubic crystal lattice, that is:

$$\begin{pmatrix} a & a & 0 \\ a & 0 & a \\ 0 & a & a \end{pmatrix} \cong \begin{pmatrix} 2a & a & a \\ 0 & a & 0 \\ 0 & 0 & a \end{pmatrix}.$$

The number of nodes of the graph is $|\det(M)| = 2|a|^3$.

Lemma 25. *The projection of $FCC(a)$ is the rectangular twisted torus graph of side a, $RTT(a)$.*

Proof. After performing Gaussian elimination, on the right of the previous expression we obtained the Hermite form of the matrix. It is easy to see that its projection is generated by $\begin{pmatrix} 2a & a \\ 0 & a \end{pmatrix}$. As we have seen before and was proved in [9], this graph is isomorphic to the rectangular twisted torus $RTT(a)$ of side a or the *Gaussian graph* generated by $a + ai$ [24]. □

A $FCC(a)$ is isomorphic to the *prismatic doubly twisted torus* of side a ($PDTT(a)$), introduced in [7], as the next Proposition proves.

Proposition 26. *$FCC(a)$ is isomorphic to the prismatic doubly twisted torus of side a, $PDTT(a)$.*

Proof. The $PDTT(a)$ was defined in [7] as a graph in which the connectivity of each plane is a $RTT(a)$, hence the isomorphism is immediate once we have proved that all the projections of $FCC(a)$ are isomorphic to $RTT(a)$. Note that this fact can be inferred from Lemma 25 and Theorem 22. □

The **Body-centered Cubic lattice graph** $BCC(a)$ of side a can be defined as the lattice graph generated by the matrix:

$$\begin{pmatrix} -a & a & a \\ a & -a & a \\ a & a & -a \end{pmatrix} \cong \begin{pmatrix} 2a & 0 & a \\ 0 & 2a & a \\ 0 & 0 & a \end{pmatrix}.$$

The number of nodes of the graph is $4a^3$. As far as we know, this graph has not previously been considered for interconnection networks. However, as we will see later, the graph not only meets the symmetry requirements but also has a better nodes/diameter

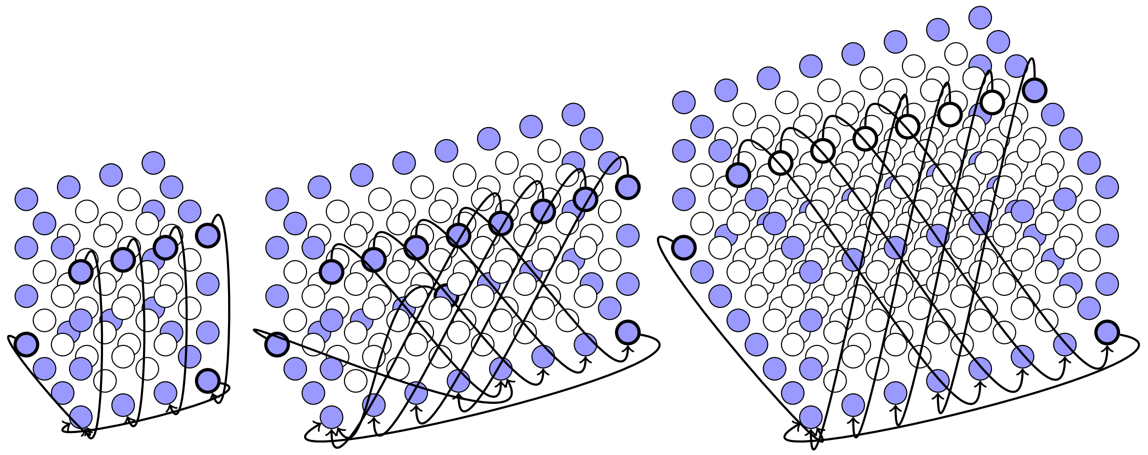

Figure 3: The three Cubic Crystal Graphs: *PC*, *FCC* and *BCC*.

ratio than PC and FCC, as it will be explained later. Moreover, it embeds 2D symmetric tori as is proved in:

Lemma 27. *The projection of $BCC(a)$ is the 2D torus graph $T(2a, 2a)$*

Proof. It can be verified that after performing Gaussian elimination over the original matrix, it is easy to see that its projection is generated by $\begin{pmatrix} 2a & 0 \\ 0 & 2a \end{pmatrix}$, which is the 2D torus of side $2a$. □

A graphical representation of the three topologies introduced in this subsection is presented in Figure 3.

5.2 Cubic Crystal Lattice Graph Comparison

Among the three different 3D symmetric topologies based on cubic crystal lattices, two of them –the 3D torus or *PC* and the PDTT or *FCC*–, were previously known, and the last one, that is the *BCC*, is a new proposal introduced in this paper. In this subsection, our aim is to consider their distance properties and to perform a first comparison in terms of diameter, average distance and projections.

First of all, we would like to highlight that a cubic crystal lattice graph exists for any number of nodes that is a power of two. This is important because we can gracefully upgrade a network in three steps while conserving symmetry. If t is a positive integer, then:

- There exists a primitive cubic lattice graph with 2^{3t} nodes.
- There exists a face-centered cubic lattice graph with 2^{3t+1} nodes.
- There exists a body-centered cubic lattice graph with 2^{3t+2} nodes.

Although this fact provides practical versatility, it complicates the comparison among networks.

In Table 1 the distance properties for the three graphs are summarized. For an easier comparison, note that average distance values are given as approximations. Mixed-radix torus graphs that have the same number of nodes of the *FCC* and *BCC* crystals have been also added in the table for comparison. Clearly, crystals have better distance properties than their corresponding torus networks. Moreover, *BCC* is more dense than the other two cubic crystals since, for the same diameter, it attains a greater number of nodes. Finally, as we have seen in previous subsections, while *FCC* has the twisted torus as

Topology	Nodes	Diameter	Average Distance
$PC(a)$	a^3	$3\left\lfloor \frac{a}{2} \right\rfloor$	$\approx \frac{3}{4}a = 0.75a$
$T(2a,a,a)$	$2a^3$	$a + 2\left\lfloor \frac{a}{2} \right\rfloor$	$\approx a$
$FCC(a)$	$2a^3$	$\left\lfloor \frac{3}{2}a \right\rfloor$	$\approx \frac{7}{8}a = 0.875a$
$T(2a,2a,a)$	$4a^3$	$\left\lfloor \frac{5}{2}a \right\rfloor$	$\approx \frac{5}{4}a = 1.25a$
$BCC(a)$	$4a^3$	$\left\lfloor \frac{3}{2}a \right\rfloor$	$\approx \frac{35}{32}a = 1.09375a$

Table 1: Distance properties of cubic crystal lattice graphs

its projection, both PC and BCC are lifts of a 2D symmetric torus graph.

Having considered distance-related parameters for comparing crystals, let us also take into account other topological parameters to complete the study. In networking literature, the *bisection bandwidth* (BB) is used to obtain an upper bound for the network load under uniform random traffic. However, it was shown in [7] that in rectangular twisted tori some minimal routes between pairs of vertices in opposite network partitions could traverse the bisection twice. Hence, this work proved that BB is not a tight bound for network throughput in twisted topologies. Indeed, the same happens with any non-torus lattice graph.

There is another way to accurately bound network throughput under uniform traffic under ideal conditions. Throughput is inversely proportional to average distance, $\bar{k}$, in symmetric networks. As, under uniform traffic at rate l, l phits are injected into each node each cycle, we have a total of $lN\bar{k}$ links being used each cycle. Any link can only transfer 2 phits (one in each way) each cycle, which implies $lN\bar{k} \leq 2|E| = \Delta N$, where Δ denotes the graph degree and N and E denote the number of nodes and the edge set, respectively. Thus, it must be highlighted that network throughput is bounded by $\dfrac{\Delta}{\bar{k}}$. For lattice graphs, $\Delta = 2n$ where n is the number of dimensions. Hence, in $FCC(a)$ maximum throughput will be bounded by $\dfrac{48}{7a}$ and in $BCC(a)$ by $\dfrac{192}{35a}$. Nevertheless, the previous count cannot be applied to edge-asymmetric networks such as mixed-radix tori. In that case, it can be seen that throughput is inversely proportional to the maximum average distance per dimension, namely $\dfrac{\Delta}{n\bar{k}_{max}}$, as inferred from [7]. Network throughput for both $T(2a,a,a)$ and $T(2a,2a,a)$ is bounded by $\dfrac{12}{3a} = \dfrac{4}{a}$ as $\bar{k}_{max} \approx \dfrac{a}{2}$, given that their longest dimensions are $2a$-node rings. This leads to an improvement in maximum throughput under uniform traffic of 71% when comparing $FCC(a)$ to $T(2a,a,a)$ and 37% for $BCC(a)$ versus $T(2a,2a,a)$.

Being symmetric has more positive impact when the number of nodes is $2a^3$. In $T(2a,a,a)$, when the links in the longest dimension are fully utilized, links in the other two shortest dimensions are used at 50%. This is because, on average, the length of the paths in the longest dimension doubles the length of the shortest ones. When the number of nodes is $4a^3$, $T(2a,2a,a)$ uses its resources better as only links in one dimension operate at half rate.

5.3 Symmetric Lifts of Cubic Crystal Graphs

First, we consider the PC. There is a straightforward way of lifting a $PC(a)$ to 4D, which is the Cartesian product of the PC by one cycle of length a, thus obtaining the generator matrix:

$$\begin{pmatrix} a & 0 & 0 & 0 \\ 0 & a & 0 & 0 \\ 0 & 0 & a & 0 \\ 0 & 0 & 0 & a \end{pmatrix}.$$

The 4D torus generated by the previous matrix is completely symmetric. However, the lifting technique can be used to embed the completely symmetric 3D torus in a different lattice graph. We will denote the *body centered hypercube lattice graph* as *4D-BCC*, that is, the lattice graph generated by matrix:

$$\begin{pmatrix} 2a & 0 & 0 & a \\ 0 & 2a & 0 & a \\ 0 & 0 & 2a & a \\ 0 & 0 & 0 & a \end{pmatrix}.$$

Proposition 28. *$4D$-$BCC(a)$ is a symmetric lattice graph of side a and projection $PC(2a)$.*

Proof. Let ϕ be defined by $\phi(\mathbf{e}_i) = \mathbf{e}_{i+1 \pmod n}$. ϕ has an associated matrix $P = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$. As $Q = M^{-1}PM = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 2 & 1 \end{pmatrix}$ is an integer matrix we conclude that ϕ is an automorphism of $4D$-BCC [8]. In the group generated by ϕ there are enough automorphisms to provide the edge-symmetry. It should be note that the projection is straightforward as the matrix is triangular superior. □

Now, if we want to lift the FCC, there are two ways of doing so which make the lifted graph symmetric. The first one will be denoted as $4D$-FCC (*4-dimensional face-centered cubic lattice graph*), that is, the lattice graph generated by matrix:

$$\begin{pmatrix} 2a & a & a & a \\ 0 & a & 0 & 0 \\ 0 & 0 & a & 0 \\ 0 & 0 & 0 & a \end{pmatrix}.$$

Proposition 29. *$4D$-$FCC(a)$ is a symmetric lattice graph of side a whose projection is a $FCC(a)$.*

Proof. Exactly like the proof of Proposition 28, the matrix $Q = M^{-1}PM$ is different but still with integer entries. □

The second way to lift a FCC is introduced below.

Proposition 30. *The lattice graph generated by the matrix $\begin{pmatrix} a & -a & -a & -a \\ a & a & -a & a \\ a & a & a & -a \\ a & -a & a & a \end{pmatrix}$ is a symmetric lifting of the $FCC(2a)$.*

Proof. First, the following two matrices are right-equivalent:

$$\begin{pmatrix} a & -a & -a & -a \\ a & a & -a & a \\ a & a & a & -a \\ a & -a & a & a \end{pmatrix} \cong \begin{pmatrix} 2a & -2a & 0 & -a \\ 0 & 2a & -2a & a \\ 2a & 0 & 2a & -a \\ 0 & 0 & 0 & a \end{pmatrix}$$

Hence, the corresponding lattice graphs are isomorphic. Note that the $(4,4)$-minor corresponds with the generating matrix of $FCC(2a)$. Finally, for symmetry, the procedure described in the proof of Proposition 28 is repeated. □

This second lifting relates the graphs obtained to the family of Lipschitz graphs and quaternion algebras, introduced in [23], for obtaining perfect codes over 4D spaces. This graph will be denoted as $Lip(a)$.

Finally, there are several ways of lifting the BCC, although none of them preserves symmetry as proved in the next theorem.

Theorem 31. *Any lift of BCC yields a non-edge-symmetric graph.*

Proof. Let $M = \begin{pmatrix} 2a & 0 & a \\ 0 & 2a & a \\ 0 & 0 & a \end{pmatrix}$, $BCC(a) \simeq \mathcal{G}(M)$. Assume that exists a symmetric lift $\mathcal{G}(L)$ of $BCC(a)$.

$$L = \begin{pmatrix} 2a & 0 & a & x \\ 0 & 2a & a & y \\ 0 & 0 & a & z \\ 0 & 0 & 0 & t \end{pmatrix}.$$

In Hermite form we have $0 \le x, y < 2a$ and $0 \le z < a$. For symmetry, the gcd of every row must be the same (map $\mathbf{e}_i$ into $\mathbf{e}_n$ and Gauss-reduce), hence t divides all the other entries of L and without loss of generality we assume $t = 1$. By [8] we know that automorphisms are matrices P satisfying the condition that $L^{-1}PL$ is an integer matrix where P is unitary and has only ± 1 entries. Both, the sets of these matrices that would give edge-transitivity, and the possible lifts, are finite. Hence we can run a computation that gives the negative result. □

As we have concluded before, there is no decisive interest in obtaining a symmetric graph in 4D such

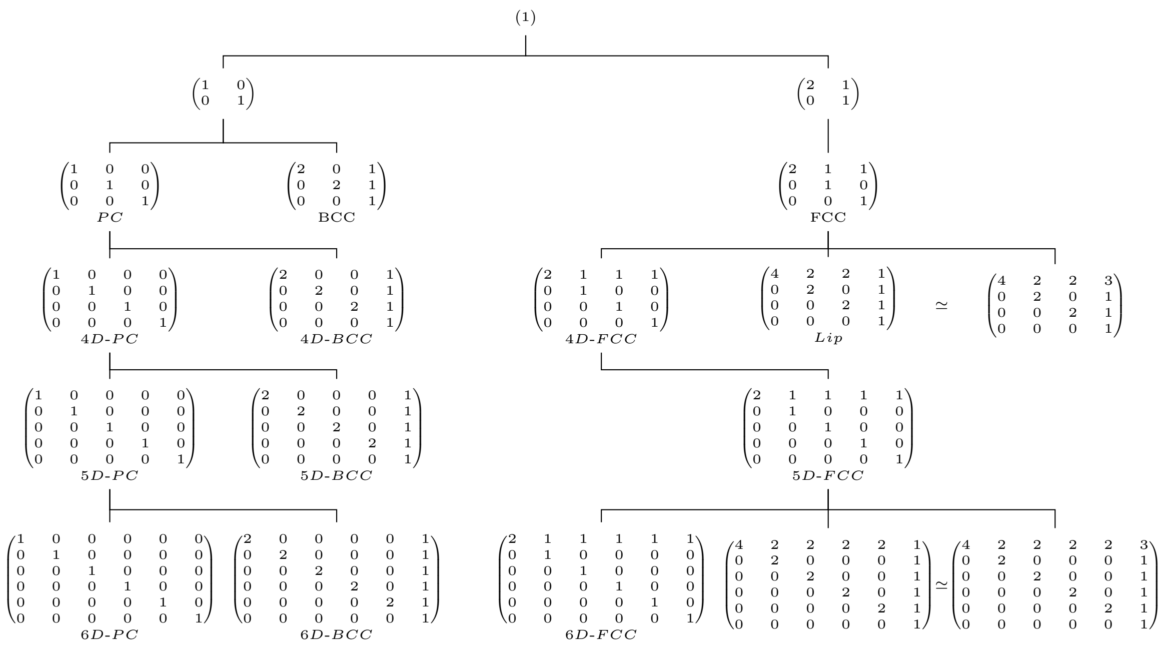

Figure 4: Tree showing lifts and projections of cubic crystal graphs up to dimension 6.

that its 3D partitions remain themselves symmetric. Therefore, we could explore which of the lattice graphs whose projection is a BCC would be the most interesting.

Figure 4 summarizes how the previous constructions can be generalized to any number of dimensions. The procedure is represented in a tree. In this tree, nodes are the matrices of the lattice graphs. Note that, for an easier visualization, matrices have been normalized by multiplying by $\frac{1}{a}$. Hence, each child is a lift of its parent. Moreover, we have restricted lifts to those whose side is greater or equal to the half of the side of its projection, otherwise many more graphs would appear.

The root of the tree is the matrix associated with a cycle. The lifts of the cycle conserving symmetry, and fulfilling the restrictions mentioned above, are the torus and the twisted torus introduced in Section 2. Then, as we have seen in Section 5, the cubic crystal lattice graphs are lifts of these two. The two branches show that only two families are obtained. The left branch consists of the infinite family of symmetric tori or n-dimensional PCs; and each nD-PC has a nD-BCC sibling that is a leaf, without any further symmetric lift. The right branch is the family of the n-dimensional FCCs; the nD-FCC always has the $(n+1)D$-FCC as a symmetric lift. Moreover, there are some dimensions (4 and 6 in the figure) in which a different lift exists. Interestingly, two non right-equivalent matrices generate isomorphic graphs (denoted with $\simeq$). The two branches in the tree are really different and, as we show next, they can be used to obtain new hybrid lattice graphs.

5.4 Hybrid Graphs: Common Lift of Crystal Graphs

In this subsection a different approach for embedding crystal graphs is considered, that is, to create common lifts that do not necessarily combine symmetric graphs. As shown in the next example, to handle graphs using the $\boxplus$ operator that belong to the same branch of the tree in Figure 4 has some advantages.

Example 32. *The first one is the hybrid graph obtained as a common lift of the $PC(2a)$ and $BCC(a)$. The calculation described in the Theorem 15 leads to the matrix*

$$\begin{pmatrix} 2a & 0 & 0 \\ 0 & 2a & 0 \\ 0 & 0 & 2a \end{pmatrix} \boxplus \begin{pmatrix} 2a & 0 & a \\ 0 & 2a & a \\ 0 & 0 & a \end{pmatrix} = \begin{pmatrix} 2a & 0 & 0 & a \\ 0 & 2a & 0 & a \\ 0 & 0 & 2a & 0 \\ 0 & 0 & 0 & a \end{pmatrix},$$

which corresponds to a 4D lattice graph. On the other hand, if we make the common lift of $PC(2a)$ and $FCC(a)$:

$$\begin{pmatrix} 2a & 0 & 0 \\ 0 & 2a & 0 \\ 0 & 0 & 2a \end{pmatrix} \boxplus \begin{pmatrix} 2a & a & a \\ 0 & a & 0 \\ 0 & 0 & a \end{pmatrix} = \begin{pmatrix} 2a & 0 & 0 & a & a \\ 0 & 2a & 0 & 0 & 0 \\ 0 & 0 & 2a & 0 & 0 \\ 0 & 0 & 0 & a & 0 \\ 0 & 0 & 0 & 0 & a \end{pmatrix}$$

which generates a 5D lattice graph. In this case, the common lift has one extra dimension since the graphs considered belong to different branches of the tree. The same happens with the mix $FCC(a)$ and $BCC(a)$, as shown next:

$$\begin{pmatrix} 2a & a & a \\ 0 & a & 0 \\ 0 & 0 & a \end{pmatrix} \boxplus \begin{pmatrix} 2a & 0 & a \\ 0 & 2a & a \\ 0 & 0 & a \end{pmatrix} = \begin{pmatrix} 2a & a & a & 0 & a \\ 0 & a & 0 & 0 & 0 \\ 0 & 0 & a & 0 & 0 \\ 0 & 0 & 0 & 2a & a \\ 0 & 0 & 0 & 0 & a \end{pmatrix}$$

Finally, we present in Table 2 a selection of lattice graphs composed following the guidelines presented in this section. The table also includes their main topological characteristics. Depending on the focus some of them outperform the others. For example, if we are looking for a 4-dimensional topology that embeds tori networks, $4D$-$BCC(a)$ and $PC(2a) \boxplus BCC(a)$ must be considered. Both topologies equal their number of nodes, so if we want to minimize distance properties, $4D$-$BCC(a)$ should be a good candidate. On the other hand, if we are interested on 5 dimensions and a great number of different embedded topologies, $PC(2a) \boxplus FCC(a)$ would be a good choice having a good nodes/distance ratio. Therefore, what we want to show with these examples is the wide range of possibilities that provides the previous $\boxplus$-operation.

Topology	Dimension	Nodes	Projection	Diameter	Average Distance
$T(2a,2a) \boxplus RTT(a)$	3	$4a^3$	vary	$2a$	$\approx 1.14877a$
$4D\text{-}FCC(a)$	4	$2a^4$	$FCC(a)$	$2a$	$\approx 1.10396a$
$4D\text{-}BCC(a)$	4	$8a^4$	$T(2a,2a,2a)$	$2a$	$\approx 1.5379a$
$Lip(a)$	4	$16a^4$	$FCC(2a)$	$3a$	$\approx 1.815a$
$PC(2a) \boxplus BCC(a)$	4	$8a^4$	vary	$2.5a$	$\approx 1.59715a$
$PC(2a) \boxplus FCC(a)$	5	$8a^5$	vary	$3.5a$	$\approx 1.87856a$
$BCC(a) \boxplus FCC(a)$	5	$4a^5$	vary	$2.5a$	$\approx 1.52522a$

Table 2: Distance properties of several lattice graphs

5.5 Routing Discussion

First, note that following the ideas in Section 4, we can infer the impact of routing complexity for the different lifts of crystal lattice graphs. As we have seen, $\frac{ord(\mathbf{e}_n)}{a}$ determines the number of intersections of the cycle with the destination projection, which dictates the number of nested routing calls. First, $\frac{ord(\mathbf{e}_n)}{a} = 1$ in *nD-PC*. Second, $\frac{ord(\mathbf{e}_n)}{a} = 2$ in *nD-BCC* and *nD-FCC*. Clearly, these are constant values, which imply just one or two calls to the routing of dimension $n-1$ in Algorithm 1. Therefore, if the algorithm is used in a recursive form, we have that:

- The routing in *nD-PC* can be done immediately with n comparisons in parallel.

- The hierarchical routing in *nD-BCC* requires 2 calls to the routing algorithm for $(n-1)D\text{-}PC$.

- The hierarchical routing in *nD-FCC* requires 2 calls to the $(n-1)D\text{-}FCC$, which accumulates into 2^{n-2} calls to $2D\text{-}FCC$ or RTT. These last routing calls will be performed by Algorithm 2, found in the appendix.

Specialized routing algorithms for FCC and BCC can be found in the appendix. As is can be noted, all the routing algorithms presented there have constant complexity.

Finally, let us consider the case of hybrid graphs. As we have seen in Section 3, hybrid graphs are obtained as common lifts of different lattice graphs. Therefore, given a hybrid graph $\mathcal{G}(M)$ there would be several possible lattice graphs that could be considered as its projection. Since the heaviest computation part in Algorithm 1 corresponds to the routing calls in the projection, that projection should be carefully chosen. For example, let $\mathcal{G}(M)$ be given by

$$M = \begin{pmatrix} 2a & 0 & 0 & a \\ 0 & 2a & 0 & a \\ 0 & 0 & 2a & 0 \\ 0 & 0 & 0 & a \end{pmatrix}.$$

As we have previously seen, this graph is obtained as the common lift of $PC(2a)$ and $BCC(a)$. Clearly, taking $BCC(a)$ as the projection, would complicate the routing function. Hence, we should choose $PC(2a)$ as its projection, in which dependencies among dimensions do not exist and routing will be less laborious.

6 Some Practical Issues

This work has been conceived to study the fundamentals of twisting wrap-around links in multidimensional torus networks. Nevertheless, this research has been motivated by the widespread presence of moderate degree tori in the supercomputing market. Although Fujitsu has recently entered in this terrain with its K system, traditionally Cray and IBM are the two major companies standing out for years in the development of interconnection networks based on torus networks. Hence, this section will be devoted to discuss certain practical aspects. The first one is related to physical network deployment and the second consists of a preliminary performance evaluation.

6.1 Physical Organization

It is not difficult to conceive a package hierarchy and a 3D physical organization to deploy systems based on lattice graphs. For illustrating this organization, let us first consider the approach followed by manufacturers. Cray uses a straightforward structure. For example, an actual configuration [5], was a $T(25, 32, 16)$ packaged on a 200 rack system arranged as an 8×25 rectangle. We can see the system as:

- System of $25 \times 8 \times 1$ racks.
- Racks of $1 \times 4 \times 16$ nodes.

That is, the third dimension is completely inside the racks and the first dimension is formed entirely joining racks. However the second dimension is partially inside the rack and requires connecting rack columns by rings. Taking into account forthcoming improvements in integration and packaging technologies, it could be expected that a 4D torus would have two dimensions internal to the racks and the other 2 external to the racks. This idea generalizes to lattice graphs. If $\mathcal{G}(M)$ is a 4D lattice graph, its 2D projections would be built inside racks, which would be a torus or a twisted torus. Then it becomes a question of completing the lattice by adjusting the offsets of the cables connecting the racks. Moreover, folding techniques for 3D networks presented in [7] can also be of application in our case and easily generalized to higher dimensions.

IBM presents a more elaborated organization in the Blue Gene family [13]. Although the complete network is a torus, each midplane (half of a rack) has additional edge hardware that enables the midplane to disconnect from the remainder of the network and to be itself a small torus. By arranging several midplanes, this additional hardware enables a multitude of different tori shapes to be connected. With slight modifications to such hardware it is possible to allow each group to be a symmetric crystal (or another lattice if desired) instead of a mixed-radix torus. This hardware changes its configuration only between different application runs. Then, the potentially added functionality would not have any negative impact on the system.

6.2 Evaluation Compared to Currently Used Topologies

Most evaluations of big networks have relied on measuring their behavior when managing synthetic traffic loads. Typical experiments are based on simulation. Notwithstanding, the work presented in [11] evaluates different routing algorithms reporting maximum achievable loads on a real IBM BlueGene/Q system. They make runs on machines whose topologies are the tori $T(8, 8, 8, 4, 2)$ and $T(16, 8, 8, 8, 2)$. We shall ignore the last dimension of size 2 and treat them as four dimensional networks; the last small dimension comes from the inside of computing nodes, fixed by computer technology. We have simulated the same tori plus symmetric lattice graphs of the same sizes. We evaluate $4D\text{-}BCC(4)$ compared to $T(8, 8, 8, 4)$ and $4D\text{-}FCC(8)$ compared to $T(16, 8, 8, 8)$.

The torus $T(8, 8, 8, 4)$ contains 2048 vertices, has diameter 14 and an average distance of 7.0. On the other hand, $4D\text{-}BCC(4)$ contains has the same number of vertices, but it has diameter 8 and average distance 6.1. In addition the torus is not symmetric while the body-centered is, thus it is expected an increase of more than $7.0/6.1 = 1.15$ for uniform loads. For the large size, $T(16, 8, 8, 8)$ contains 8192 vertices with diameter 20 and average distance 10.0; while $4D\text{-}FCC(8)$ has diameter 16 and average distance 8.8. In this subsection simulation results show that better distance properties plus symmetry translate into a better performance of symmetric networks.

We have used the same synthetic traffic patterns as in [11]:

- **uniform** Each node generates packets to any other node, at random with a uniform probability distribution.
- **antipodal** Each node generate traffic to the

most distant one. Also known as *furthest-node pairing*.

- **centralsymmetric** Once a center of symmetry is fixed, each node has as its destination the symmetric one. It is the immediate generalization of *diagonal pairing*.
- **randompairings** The network is divided into pairs in a random uniform way, which then communicate for all the simulation.

Simulations have been conducted using INSEE (Interconnection Network Simulation and Evaluation Environment) [25]. Their basic units are the *cycle* for measuring time and the *phit* for measuring information. Each network link (edge of the graph) can send one or zero phits in each cycle. The network *load* is the amount of information injected per time unit. We measure the network load in phits/(cycle $\cdot$ node). Nodes (vertices of the graph) generate *packets* composed of an integral number of phits (typically constant) to be sent to other network nodes. For any provided traffic up to load l, a packet is injected each cycle in each node with probability $\frac{l}{s}$, where s is the size of a packet measured in phits. The accepted traffic or throughput is the total of phits received divided by the total simulation time and by the number of nodes N. Simulation parameters are shown in Table 3. We have simulated $100,000$ cycles for statistics, preceded by a network warmup. At least 5 simulations are averaged for each point. The BlueGene family of supercomputers implements a congestion control mechanism that prioritizes in-transit traffic over new injections, which is also implemented in our router model.

Figures 5 and 6 show results of accepted load in the four networks. Under uniform traffic, results exhibit gains of 27% in the small case (4*D-BCC*) and 49% in the large one (4*D-FCC*). In random pairings, the throughput is consistently higher, with gains of 15% and 2% respectively. The other two traffic patterns show congestion at high loads for all the networks considered. Nevertheless, the peak throughput for the antipodal traffic improves by 95% and 43% respectively. Under central symmetric traffic, gains

Injectors	6
Packet size	16 phits
Queues	4 packets
Deadlock avoidance	Bubble
Virtual Channels	3
flow control	Virtual Cut-through
Routing Mechanisms	DOR
Arbitration mechanism	random

Table 3: Simulation parameters

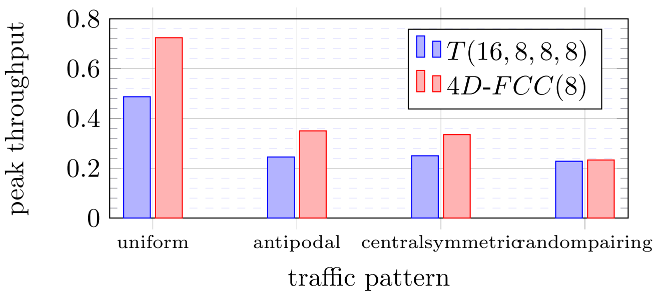

Figure 5: Throughput peak in $T(16,8,8,8)$ and $4D$-$FCC(8)$ under several synthetic traffics.

are 29% in the small case and 34% in the large one. Figures 7 and 8 show average packet latencies. The different curves demonstrate the superior behavior of lattice topologies. Gain values are coherent with the topological analysis presented in Subsection 5.2.

7 Conclusions

This research has been focused on the study and proposal of new multidimensional twisted torus interconnection networks. Due to their complex spatial characteristics, their analysis is far from being straightforward. Nevertheless, we have taken advantage of an algebraic tool based on integral square matrices presented in [18]. Such matrices define the graph and its topological characteristics. Adequate algebraic manipulations of the matrices enable a better understanding of different network properties. For example, when using the Hermite normal form, ma-

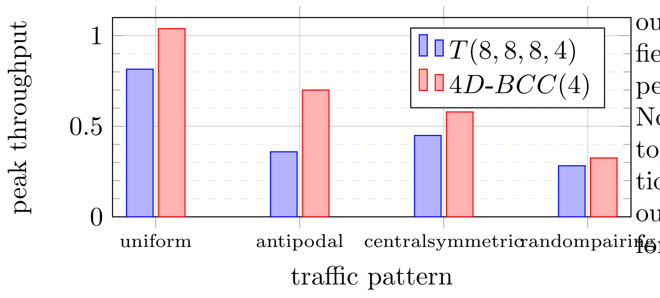

Figure 6: Throughput peak in $T(8,8,8,4)$ and $4D$-$BCC(4)$ under several synthetic traffics.

trices reveal the subgraphs naturally embedded in the network.

Using this tool, several networks have been proposed and analyzed in this paper. We have introduced two graph lifting methods that allow for higher dimensional networks that embed lattice subnetworks. Complementarily, the use of graph projections facilitates the conception of routing algorithms for these networks. Based on this graph operation, minimal routing schemes have been proposed for all the topologies. Then, we focus on 3D symmetric networks as alternatives to mixed-radix tori that are not edge-symmetric. Taking the matrices that define cubic crystallographic lattices, we were able to evaluate and compare their associated interconnection networks. If symmetry is desired, the best path when upgrading 3D systems clearly seems to be $PC(a) \rightarrow FCC(a) \rightarrow BCC(a) \rightarrow PC(2a)$, that is, duplicating the machine size on each step and maintaining most of the original connections. Although we have focused on typical network configurations derived from powers of two, our results remain valid for any other network size.

The paper preliminarily addresses some practical issues. Physical packaging and system organization in racks have been taken into account, concluding that, for deploying networks based on lattice graphs, very few changes over typical tori would be necessary. In addition to the algebraic analysis carried out through the paper, an empirical evaluation of different interesting topologies has been carried out. Comparisons with current machines have certified that hyper-dimensional twisted tori clearly outperform their standard (not twisted) counterparts. Noticeable gains were exhibited by twisted lattice topologies for both configurations under consideration. These preliminary experiments motivate a thorough network evaluation that will be reported in a forthcoming work.

Acknowledgments

This work has been supported by the Spanish FPU grant AP2010-4900, the Spanish Science and Technology Commission (CICYT) under contracts TIN2010-21291-C02-02 and TIN2013-46957-C2-2-P, the European Union FP7 under Agreements ICT-288777 (Mont-Blanc) and ERC-321253 (RoMoL), the European HiPEAC Network of Excellence and the JSA no. 2013-119 as part of the IBM/BSC Technology Center for Supercomputing agreement.

References

[1] Yuichiro Ajima, Shinji Sumimoto, and Toshiyuki Shimizu. Tofu: A 6D mesh/torus interconnect for exascale computers. *Computer*, 42:36–40, 2009.

[2] S.B. Akers and B. Krishnamurthy. A group-theoretic model for symmetric interconnection networks. *IEEE Transactions on Computers*, 38:555–566, 1989.

[3] G.H. Barnes, R.M. Brown, M. Kato, D.J. Kuck, D.L. Slotnick, and R.A. Stokes. The Illiac IV computer. *IEEE Transactions on Computers*, C-17(8):746–757, aug. 1968.

[4] Ramón Beivide, Enrique Herrada, José L. Balcázar, and Agustin Arruabarrena. Optimal distance networks of low degree for parallel computers. *IEEE Trans. Comput.*, 40(10):1109–1124, 1991.

[5] Buddy Bland. Jaguar: Powering and cooling the beast. `http://www.cse.ohio-state.edu/`

`~panda/875/class_slides/cray-jaguar.pdf`, 2009.

[6] Jin-yi Cai, George Havas, Bernard Mans, Ajay Nerurkar, Jean-Pierre Seifert, and Igor Shparlinski. On routing in circulant graphs. In *COCOON*, pages 360–369, 1999.

[7] Jose M. Camara, Miquel Moreto, Enrique Vallejo, Ramon Beivide, Jose Miguel-Alonso, Carmen Martínez, and Javier Navaridas. Twisted torus topologies for enhanced interconnection networks. *IEEE Transactions on Parallel and Distributed Systems*, 21:1765–1778, 2010.

[8] Cristóbal Camarero, Carmen Martínez, and Ramón Beivide. Symmetric L-graphs. In *2010 International Workshop on Optimal Network Topologies*, 2010.

[9] Cristóbal Camarero, Carmen Martínez, and Ramón Beivide. L-networks: A topological model for regular two-dimensional interconnection networks. *Computers, IEEE Transactions on*, PP(99):1, 2012.

[10] Cristóbal Camarero, Carmen Martínez, and Ramón Beivide. Classes of symmetric cayley graphs over finite Abelian groups of degrees 4 and 6. 2014. arXiv:1403.5440.

[11] Dong Chen, Noel Eisley, Philip Heidelberger, Sameer Kumar, Amith Mamidala, Fabrizio Petrini, Robert Senger, Yutaka Sugawara, Robert Walkup, Burkhard Steinmacher-Burow, Anamitra Choudhury, Yogish Sabharwal, Swati Singhal, and Jeffrey J. Parker. Looking under the hood of the IBM Blue Gene/Q network. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, pages 69:1–69:12, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.

[12] Dong Chen, Noel A. Eisley, Philip Heidelberger, Robert M. Senger, Yutaka Sugawara, Sameer Kumar, Valentina Salapura, David L. Satterfield, Burkhard Steinmacher-Burow, and Jeffrey J. Parker. The IBM Blue Gene/Q interconnection network and message unit. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 26:1–26:10, New York, NY, USA, 2011. ACM.

[13] P. Coteus, H. R. Bickford, T. M. Cipolla, P. G. Crumley, A. Gara, S. A. Hall, G. V. Kopcsay, A. P. Lanzetta, L. S. Mok, R. Rand, R. Swetz, T. Takken, P. La Rocca, C. Marroquin, P. R. Germann, and M. J. Jeanson. Packaging the Blue Gene/L supercomputer. *IBM Journal of Research and Development*, 49(2.3):213 –248, march 2005.

[14] Cray XE6 brochure. `http://www.cray.com/Products/XE/Technology.aspx`.

[15] Daniel Dadush and Santosh Vempala. Deterministic construction of an approximate m-ellipsoid and its applications to derandomizing lattice algorithms. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 1445–1456. SIAM, 2012.

[16] Daniel Dadush and Santosh S. Vempala. Near-optimal deterministic algorithms for volume computation via m-ellipsoids. *Proceedings of the National Academy of Sciences*, 2013.

[17] Greg Faanes, Abdulla Bataineh, Duncan Roweth, Tom Court, Edwin Froese, Bob Alverson, Tim Johnson, Joe Kopnick, Mike Higgins, and James Reinhard. Cray cascade: a scalable HPC system based on a dragonfly network. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, pages 103:1–103:9, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.

[18] M.A. Fiol. On congruence in $\mathbb{Z}^n$ and the dimension of a multidimensional circulant. *Discrete Math*, 141:1–3, 1995.

[19] Mary Flahive and Bella Bose. The topology of Gaussian and Eisenstein-Jacobi interconnection networks. *IEEE Trans. Parallel Distrib. Syst.*, 21(8):1132–1142, 2010.

[20] Domingo Gómez, Jaime Gutierrez, Álvar Ibeas, Carmen Martínez, and Ramón Beivide. On finding a shortest path in circulant graphs with two jumps. In *COCOON*, pages 777–786, 2005.

[21] T. Janssen. *Crystallographic Groups.* American Elsevier, 1973.

[22] A. J. Martin. The torus: An exercise in constructing a processing surface. *Proceedings of the VLSI Conference*, 1981.

[23] C. Martínez, R. Beivide, and E.M. Gabidulin. Perfect codes from Cayley graphs over Lipschitz integers. *Information Theory, IEEE Transactions on*, 55(8):3552 –3562, aug. 2009.

[24] Carmen Martínez, Ramon Beivide, Esteban Stafford, Miquel Moreto, and Ernst M. Gabidulin. Modeling toroidal networks with the Gaussian integers. *IEEE Transactions on Computers*, 57:1046–1056, 2008.

[25] Javier Navaridas, Jose Miguel-Alonso, Jose A. Pascual, and Francisco J. Ridruejo. Simulating and evaluating interconnection networks with insee. *Simulation Modelling Practice and Theory*, 19(1):494 – 515, 2011. Modeling and Performance Analysis of Networking and Collaborative Systems.

[26] Cheolmin Park, R. Badeau, L. Biro, J. Chang, T. Singh, J. Vash, Bo Wang, and T. Wang. A 1.2 TB/s on-chip ring interconnect for 45nm 8-core enterprise Xeon® processor. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 180 –181, feb. 2010.

[27] Borut Robic. Optimal routing in 2-jump circulant networks. Technical report, University of Cambridge Computer Laboratory, TR397, 1996.

[28] Carlo H. Sequin. Doubly twisted torus networks for VLSI processor arrays. In *ISCA '81: Proceedings of the 8th annual symposium on Computer Architecture*, pages 471–480, Los Alamitos, CA, USA, 1981. IEEE Computer Society Press.

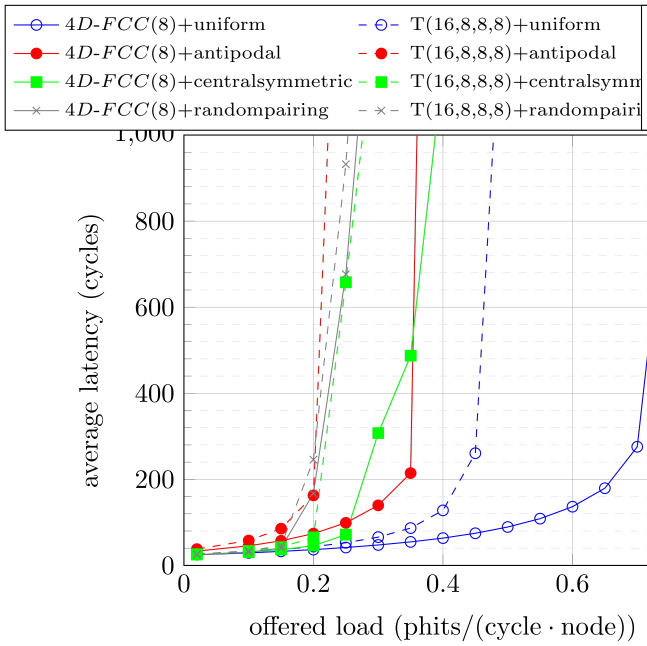

Figure 7: Packet delays in T(16,8,8,8) and 4*D*-*FCC*(8) under several synthetic traffics.

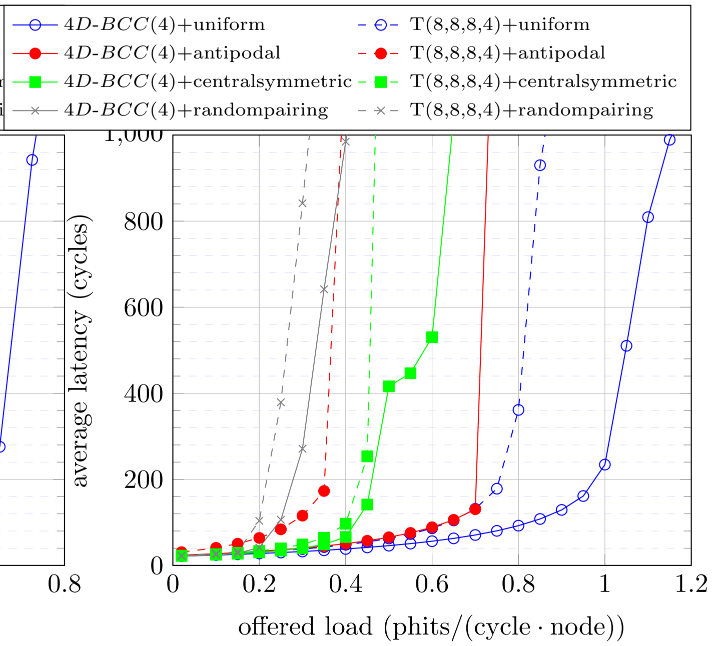

Figure 8: Packet delays in T(8,8,8,4) and 4*D*-*BCC*(4) under several synthetic traffics.