Hamiltonian Decompositions of Regular Topology Networks with Convergence Routing

Bülent Yener, [*†] Terry Boult, [‡] Yoram Ofek, [§]

Columbia University
Computer Science Department
Technical Report CUCS-011094

Abstract

This paper introduces new methods to construct multiple *virtual rings* for loss-free routing of non-reserved bursty data in high-speed environments such as ATM LANs. The routing algorithm on multiple virtual rings is *convergence routing* which combines the actual routing decision with the internal flow control state.

Multiple virtual rings are obtained on the hypercube and the circulant networks such that each virtual ring is hamiltonian, and are mutually edge-disjoint. It is shown that multiple virtual rings improve (i) the bound on the length of routing, and (ii) the fault tolerance.

On the circulant graphs, necessary and sufficient conditions for hamiltonian decomposition is established. On the hypercube, three algorithms are designed for an N-*node* hypercube with even dimension: (i) an $O(N)$ time algorithm to find two edge-disjoint hamiltonian circuits, (ii) an $O(N \log N)$ time algorithm to find $\frac{\log N}{2}$ hamiltonian circuits with only $\epsilon \leq 0.1$ common edges, and (iii) a recursive algorithm for the hamiltonian decomposition of the hypercube with dimension power of two.

It is shown analytically, and verified by simulations on the circulants that with the d virtual ring embeddings, a bound of $O(N/d)$ is established on the maximum length of routing.

[*]College of Computer Science Northeastern University, Boston MA, yener@ccs.neu.edu

[†]Department of Computer Science, Columbia University, New York, NY, yener@cs.columbia.edu

[‡]Lehigh University Department of Electrical Engineering and Computer Science, Bethlehem, PA, tboult@eecs.Lehigh.edu.

[§]IBM T.J.Watson Research Center, P.O.Box 704, Yorktown Heights, NY 10598, ofek@watson.ibm.com

1 Introduction

Supporting bursty traffic sources on ATM LANs are the focus of growing research and development activities. ATM was initially conceived as a network architecture for connection oriented traffic, over fixed routes, and with reserved bandwidth per connection. Thus, the initial congestion control scheme for ATM was based on an "open-loop" mechanism, called "Leaky-Bucket" [23]. The Leaky-Bucket scheme enables only limited traffic burstiness, which should be negotiated with the network in advance. However, the "open-loop" approach may not be adequate for many types of applications, such as, distributed data processing with bursty sources.

In this paper we consider embeddings of multiple virtual rings into a network with bursty data sources to achieve congestion free (i.e., no-loss due to congestion) routing of the non-reserved bursty traffic. Each virtual ring operates like the MetaRing [6, 15] (i.e., buffer insertion ring with spatial bandwidth reuse).

The routing algorithm on multiple virtual rings is *convergence routing* [16, 17] which is a variant of *deflection routing* [4, 14]. Convergence routing ensures no-loss due to congestion and combines, in a dynamic fashion, the on-line routing decision with the traffic load inside network. However, unlike other deflection techniques, convergence routing guarantees that packets will reach (or converge) to their destinations. Thus it is suitable for ATM LANs with bursty traffic characteristics.

Routing of the non-reserved bursty traffic is realized on the virtual rings, embedded into the network topology. In this work we introduce new methods for embedding of multiple virtual rings such that (i) rings are pairwise edge-disjoint (for congestion free-ness), and (ii) each is hamiltonian (i.e., includes each node exactly once). Multiple virtual rings provide a *global sense of direction* and allow switching of the packets from one ring to another for fast convergence to their destinations. Convergence routing on the multiple virtual rings not only ensures a deterministic worst case bound on the length of routing, but also improves the fault tolerance.

In this paper we consider two types of regular networks to show how to construct multiple virtual rings: the *circulants* and the *hypercubes*. These regular topologies has several properties that are useful in network design and analysis. Furthermore, as the high-speed networks operate like multi-computers for parallel processing.

First we consider the circulant networks with N nodes, each with degree d and present an algebraic construction of d global virtual rings. The circulant networks are suggested for reliable communication networks [9], and for efficient routing [19, 20, 7]. These works have similar bound on the routing length, but it is achieved by compromising congestion-free routing and failures can result in deadlocks.

Second we consider the *N-node* hypercube (which is a hamiltonian network) and design new algorithms to find edge-disjoint hamiltonian circuits [1]. This problem is called the hamiltonian decomposition of the hypercube if every edge is included into some hamiltonian circuit. Decomposition of the hypercube is possible for any dimension d [1]. For even dimension, the decomposition results $\frac{d}{2}$ hamiltonian cycles, whereas for odd dimension the result is $\lfloor \frac{d}{2} \rfloor$ hamiltonian cycles and one perfect matching. However, there are no simple algorithms for obtaining such a decomposition, except for some special cases.

In this work we consider this problem and present an $O(N)$ algorithm to construct two edge-disjoint hamiltonian circuits of an *N-node* hypercube with even degree. In addition we present another algorithm with running time $O(N \log N)$ to construct $\frac{\log N}{2}$ hamiltonian circuits with a fraction $\epsilon \leq 0.10$ of common edges. We call the result of this algorithm a hamiltonian ϵ-Decomposition. Since this fast algorithm yields the maximum number of virtual rings, we note that pseudo-decomposition is a useful tool.

Convergence routing is simulated on the multiple virtual rings to verify the analytical bounds on the length of routing. The performance measure considered in this paper is the average number of hops assuming: (i) uniform destination distribution, and (ii) heavy load traffic conditions, which means that inside the network packets are sent over the longest possible paths (i.e., worst case analysis). We show analytically and verify computationally that a bound of $O(N/d)$ can be established on the worst case length of routing.

This paper is organized as follows. In the next section we explain the network model and explain virtual ring embedding. In Section 3, we discuss the conditions for circulant networks to be hamiltonian and hamiltonian decomposition of the hypercube. In Section 4, routing and flow control on the multiple virtual rings, is explained. In Section 5, analytic bounds on the length of routing is derived. In Section 6, experimental results are presented and compared to the analytical bounds. The work is concluded in Section 7.

[1] In this paper we consider simple cycles and use cycle and circuit interchangeably.

2 The Network Model

In this work we consider the circulant and the hypercube networks in which each node has a unique ID, denoted by a capital letter A, B, C, etc., and the links are full duplex (i.e., bidirectional).

Both the circulant networks and the hypercube are node-symmetric graphs (i.e., by simply relabeling the nodes, any node can be mapped onto any other node) with small diameter. This property enables us to analyze the network from an arbitrary node's point of view. Furthermore, since edge connectivity of a connected point-symmetric graph is equal to the minimum degree of this graph, we ensure that the network with degree d is a d-edge connected network.

2.1 Virtual Ring Embeddings

In this paper virtual ring embeddings are based on the hamiltonian cycles of the network topology. Each undirected hamiltonian embedded ring (HER) can be used to obtain two directed global rings, thus each unidirectional virtual ring has a **dual** in the opposite direction. For example, consider the network in Figure 1 which has four virtual rings (see Figure 2) such that ring HER_0 has a dual HER_1 and ring HER_2 has a dual HER_3. Each time a virtual ring visits a node, a *virtual node* is induced. Thus each node U has exactly one virtual node on each virtual ring, and a node with degree d has d virtual nodes.

Each virtual ring is a buffer insertion ring with spatial bandwidth reuse and operates according to the principles of the MetaNet [16]. Virtual embeddings of rings into the networks linearize them to ensure (1) no packet loss due to congestion (buffer contention) inside the network, (2) fair and deadlock free access, and (3) self-routing and simple broadcast/multicast with a single input buffer per link.

2.2 Global Assignment of Virtual Node Labels

The virtual node labels should be assigned in a systematic and unique manner. For a network of N nodes and degree d, each virtual node is assigned two integers: **r** - to indicate the ring number that it resides in, 0, 1, ..., $(d-1)$, and **l** - to indicate its offset which is the distance from a predefined origin on the ring, 0, 1, ..., $(N-1)$. The computation of the virtual node label, VN_j, is simply: $r \times N + l$. For example, consider the virtual rings in Figure 2 where node C has virtual address: $\{VN_2, VN_{16}, VN_{26}, VN_{28}\}$ and the global ID of its virtual nodes are computed as $2 = 2 + (0 \times 9)$, $16 = 7 + (1 \times 9)$, $26 = 8 + (2 \times 9)$, $28 = 1 + (3 \times 9)$, one on each virtual ring. Since each node has exactly one virtual node on each virtual ring, virtual address of a node can

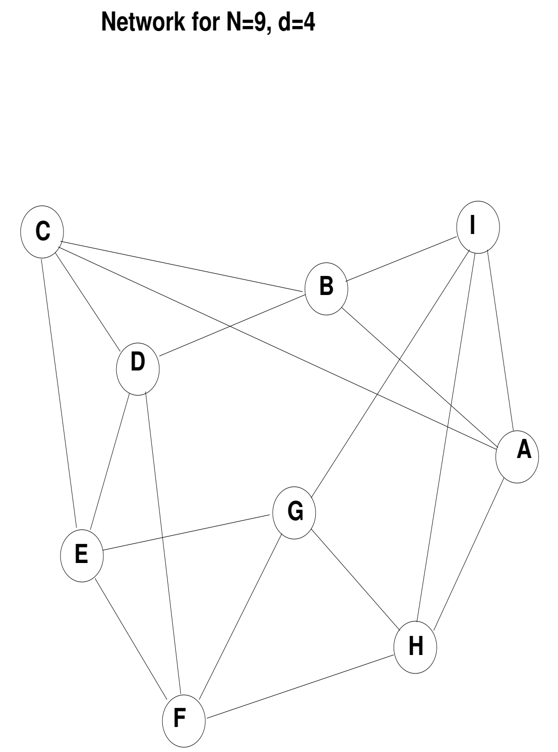

Figure 1: The Circulant Network

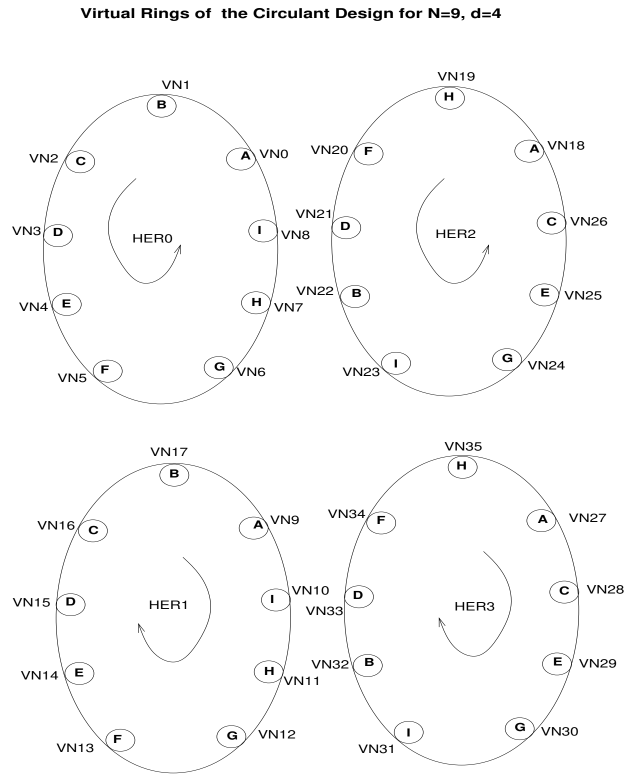

Figure 2: Hamiltonian Virtual Ring Embeddings (HER)

be represented as a *d-field* array.

Having explained the virtual ring embeddings, next we consider how to construct multiple virtual rings in the circulants and the hypercube.

3 Hamiltonian Virtual Ring Embeddings

In this section we show how to obtain the multiple virtual rings in two regular topologies. Although it is possible to construct *partial* rings (i.e., each ring visits only a subset of the nodes), we limit this paper to the hamiltonian virtual rings. We assume an assignment of a unique positive integer j, between 0 and N-1, to each node in order to express the construction of the rings algebraically (i.e., A=0, B=1,...).

3.1 Hamiltonian Circulant Networks

A circulant network with N nodes is constructed by connecting the node j, to the nodes $(j \pm n_i)$ *mod* N, where n_i is called a *jump size* and $j = 0, 1, \cdots, N-1$ [8]. For example, suppose

$N = 9$ and $n_i = 7$, then node 7 will be connected to node $(7 + 7)$ mod $9 = 5$ and to node 0. Similarly node 5 will be connected to $(5 + 7)$ mod $9 = 3$ and node 7. Construction of circulant graphs is quite simple and can be done in linear time simply by determining the jump sequence. To meet the constraints that d virtual rings are (i) hamiltonian, and (ii) mutually edge-disjoint, one must choose carefully $d/2$ jumps to construct the circulant.

First note that the edge-disjoint virtual rings are a natural outcome of the circulant based design. To prove that let j be the integer associated with a node and $n_i < n_{i+1}$ be any two jumps. Consider the edges that can be generated by these jumps from the node j. Since $(j \pm n_i) mod\ N \neq (j \pm n_{i+1}) mod\ N$, the other end point of each one of these edges is unique.

However, it is not always true that the circulant graph has hamiltonian virtual rings. In order to ensure that each cycle is hamiltonian, we state the following sufficiency condition and use a basic theorem from number theory to prove it. Denote by $gcd(x, y)$ the greatest common divisor of the integers x, y.

Claim 1 *Let n_i be a jump, if $gcd(n_i, N) = 1$, then the edges constructed by this jump induces a hamiltonian cycle.*

Proof: Note that the unique numbers $j = 0, \cdots, N-1$ constitute a complete system of residues modulo N. Then $(0 \times n_i) + b, (1 \times n_i) + b, \cdots, (N - 1 \times n_i) + b$ is a complete system of modulo N for any integer b, since $gcd(n_i, N) = 1$ (see also theorem 3.6. in [22]).□

Following we present an example to summarize the construction process. Suppose $N = 9$ and $d = 4$, and consider Figure 3, in which two jumps are chosen as $n_1 = 1$ and $n_2 = 7$ (both are relatively prime to $N = 9$, in order to ensure the hamiltonian property). Each hamiltonian circuit is generated by a distinct jump n_i, by constructing an edge (j, j'), between the node which is associated with integer j, and the node with $j' = (j + n_i)$ mod N. For example in Figure 3, the first ring is generated by the jump size $n_1 = 1$, thus there is an edge between A and B which is obtained by a jump from 0 to $1 = (0 + 1)$ mod 9. The topology for the circulant network—obtained from the union of these rings—is shown in Figure 1 and the virtual rings are shown in Figure 2.

3.2 Hamiltonian Decomposition of the Hypercube

In this section we present algorithms for embeddings of multiple hamiltonian virtual rings into even dimensional hypercube. Denoted by HQ^d, the N-*node* hypercube is a *d_regular* graph with degree $d = \log N$, 2^d nodes and $M = d2^{d-1}$ edges. Each node is represented with a *d_bit*

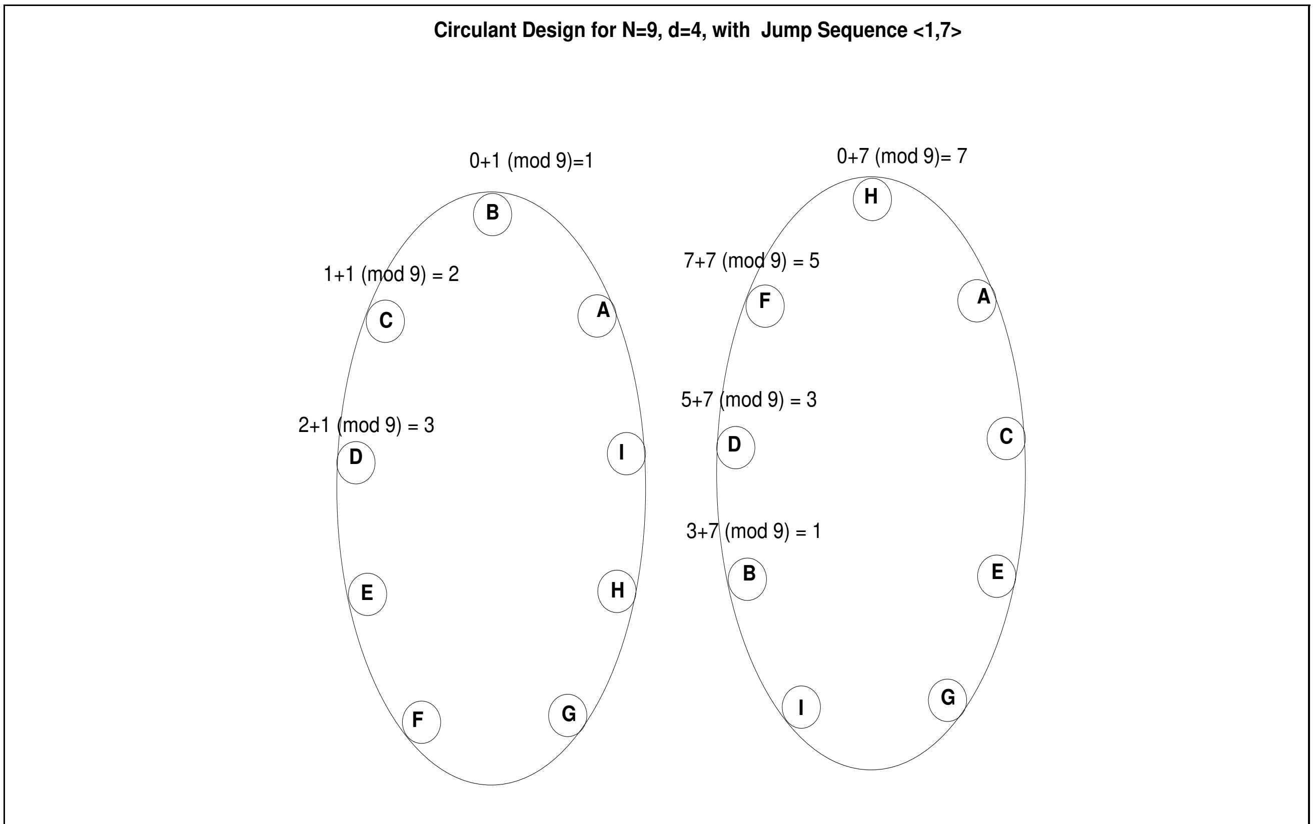

Figure 3: Construction of Hamiltonian Cycles for $n_1 = 1, n_2 = 7$

binary string and two nodes are adjacent if and only if their binary string differ in exactly one bit position. The hypercube is a hamiltonian graph (i.e., it contains a hamiltonian cycle) and sequence of nodes traversed by a hamiltonian cycle generates a *Gray code* [13]. There are three main previous results that ensure the hamiltonian decomposition of a hypercube. The first is from Kotzig [12] and shows how to decompose the cartesian product of two cycles into two hamiltonian circuits. The second is from Foregger [10] who showed that cartesian product of three cycles is also hamiltonian decomposable. Finally Aubert and Schneider [3] showed that the cartesian product of a cycle and a 4-regular graph, which has two hamiltonian cycles, is decomposable into three hamiltonian cycles.

Although these results are constructive, driving an algorithm for decomposition of the hypercube requires considering many cases and will not to be included into the scope of this paper (the general case for hamiltonian decomposition of the hypercube is studied in [24]). Therefore, we consider the hypercube with even dimension and note that the *N-node* hypercube can be decomposed into d global virtual rings since:

Theorem 1 *[5, 11] Let $N = 2^d$. Then for any even d, the* N-node *hypercube can be decomposed*

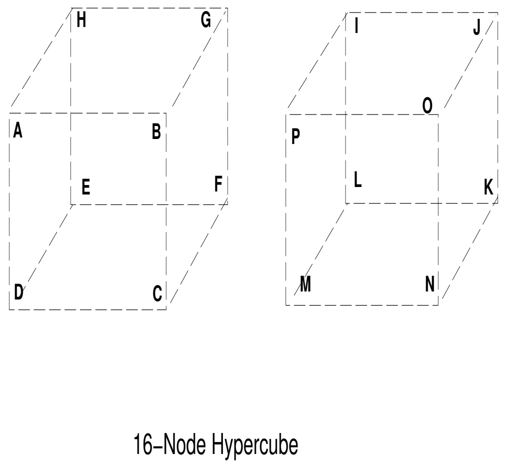

Figure 4: The hypercube N=16, d=4

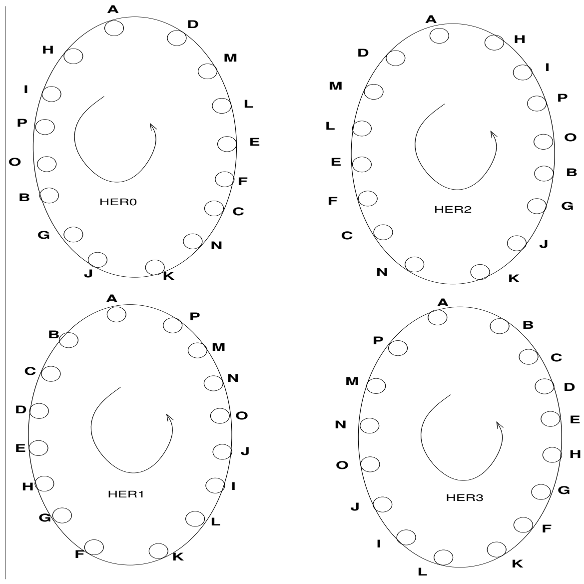

Figure 5: 4 edge-disjoint directed HC

into $\frac{d}{2}$ edge-disjoint hamiltonian cycles.$\square$

For example, in Figure 5 a decomposition of the 16_node hypercube (shown Figure 4), into four directional rings is illustrated. Each such a ring can be used for a HER.

In what follows we present three algorithms for finding hamiltonian circuits of an even dimensional hypercube. The first algorithm is solves the special case where the dimension is a power of two. The second algorithm finds not $d/2$ but exactly two edge-disjoint hamiltonian cycles. The last one is a fast approximation to the $d/2$ edge-disjoint hamiltonian cycles, since it finds $d/2$ hamiltonian cycles which share a small faction of the total edges.

3.2.1 The Degree is a power of two

In this section we consider the special case where the dimension of the hypercube is a power of two [21], and present a simple algorithm for its hamiltonian decomposition.

Algorithm has two phases, and its correctness is based on two theorems proven by Kotzig [12]. In the first phase (*cycle phase*), two hamiltonian cycles of size $2^{d/2}$ is used to obtain a torus of size 2^d. In the second phase (*torus phase*), the torus is decomposed into two hamiltonian cycles of size 2^d. The algorithm switches back and forth between these two phases, until the dimension of the target hypercube is reached.

Let C_i^d be the $i'th$ hamiltonian cycle in the d dimensional hypercube HQ^d ($d = 2^p$ where p is a non negative integer). Let G_i be the $N \times N$ torus obtained by $C_i^d \Box C_i^d$ (where $\Box$ denotes the cartesian product).

Claim 2 *The torus G_i is decomposable into two hamiltonian circuits.*

Proof: The cartesian product of two arbitrary cycles is decomposable into two hamiltonian cycles Kotzig [12]. We proceed with an example to show the hamiltonian decomposition of the hypercube with $N = 2^{16}$ nodes. Consider Figure 6, in which 8 cycles (each is a 2×2 cycle) are used construct a torus of size $2^2 \times 2^2$ from the cartesian product of a distinct pair of these cycles. The hamiltonian decomposition of each of these toruses yields to two hamiltonian cycles (each of size 2^4). Let h_{ij} be the $j'th$ cycle in the $i'th$ torus. Then we can express the hypercube as

$$(h_{11} \bigcup h_{12}) \Box (h_{21} \bigcup h_{22}) \Box (h_{31} \bigcup h_{32}) \Box (h_{41} \bigcup h_{42})$$

$$(h_{11} \Box h_{21} \Box h_{31} \Box h_{41}) \bigcup (h_{12} \Box h_{22} \Box h_{32} \Box h_{42})$$

Note that the pair $h_{11} \Box h_{21}$ results a torus of size 2^8 with a hamiltonian decomposition of two cycles. Let H_{ij} be the $j'th$ cycle in the $i'th$ torus. Then we can express the hypercube as

$$(H_{11} \bigcup H_{12}) \Box (H_{21} \bigcup H_{22}) \bigcup (H_{31} \bigcup H_{32}) \Box (H_{41} \bigcup H_{42})$$

$$(H_{11} \Box H_{21}) \bigcup (H_{12} \Box H_{22}) \bigcup (H_{31} \Box H_{41}) \bigcup (H_{32} \Box H_{42})$$

Note that the pair $H_{11} \Box H_{21}$ results a torus of size 2^{16}. Let $\mathcal{H}_{ij}$ be the $j'th$ cycle in the $i'th$ torus. Then we obtain the decomposition of the $N = 2^{16}$ node hypercube to 8 hamiltonian cycles:

$$\mathcal{H}_{11} \bigcup \mathcal{H}_{12} \bigcup \mathcal{H}_{21} \bigcup \mathcal{H}_{22} \bigcup \mathcal{H}_{32} \bigcup \mathcal{H}_{33} \bigcup \mathcal{H}_{41} \bigcup \mathcal{H}_{42}$$

To conclude the correctness of the algorithm, let $G_1, G_2, \cdots, G_p$ be the toruses obtained as explained above. Note that the *d_dimensional* hypercube HQ^d is a $2 \Box 2 \Box \cdots \Box 2$ array [13], then it follows that $G_1 \Box G_2 \Box \cdots \Box G_p = HQ^d$, where $d = 4p$.

Claim 3 *The hypercube HQ^d is decomposable into $\frac{d}{2}$ hamiltonian circuits.*

Proof: ¿From the previous claim we know that each G_i is decomposable two hamiltonian circuits. Thus, due to Kotzig [12] the graph $G_1 \Box G_2 \Box \cdots \Box G_p$ has a decomposition into $2p$ hamiltonian cycles. Since $d = 4p$, claim holds.

What remains to be shown is to find the hamiltonian cycles in each torus. In Figure 7, we demonstrate how to find the two hamiltonian cycles on a 12×12 torus (only one of the hamiltonian cycles is marked for the clarity). However, the same method can be used in hamiltonian decomposition of any $N \times N$ torus [12, 5].

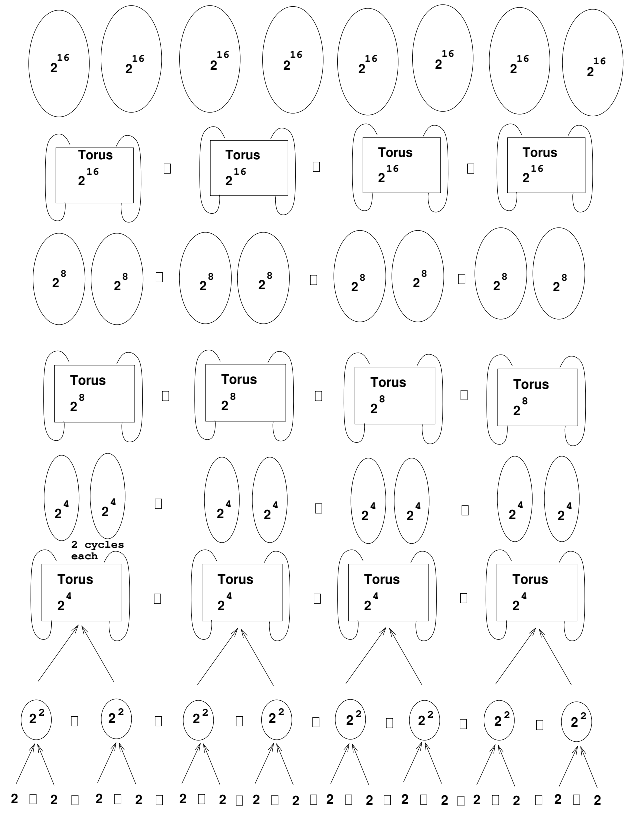

Figure 6: Hamiltonian decomposition of a $N = 2^{16}$-node hypercube

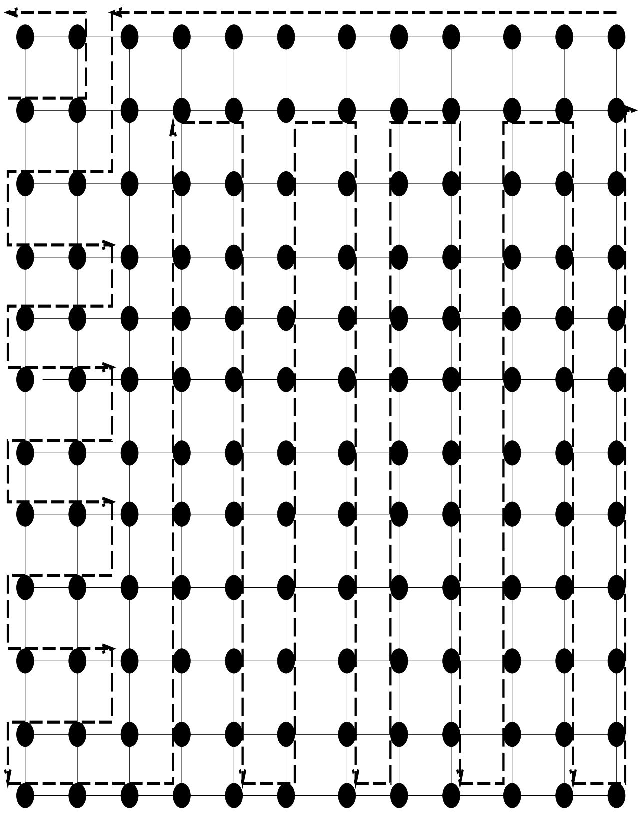

Figure 7: Hamiltonian decomposition of a 12×12 torus

3.2.2 2-Hamiltonian Decomposition

In this section, we introduce a linear time (i.e., $O(N)$) algorithm which constructs *exactly* 2 edge-disjoint hamiltonian cycles from a hypercube with *any* even degree. The algorithm is based on the fact that $(2N+2)$ node hypercube is isomorphic to the Cartesian product of two $(N+1)$. A multi-index—based on the gray code of the nodes—is defined for generating one hamiltonian cycle. Each multi-index consists of two subindices and takes the leftmost $\frac{d}{2}$ bits as the first index i_0 and the other half as the i_1. For example a node with binary string 00111110—in a 8-dimensional hypercube—is mapped to a multi-index (2,12). In our algorithm (i) each subindex is associated with a unique loop variable (i.e., j_0, j_1), (ii) each loop variable is initialized to the corresponding subindex (e.g., $i_0 = 2,\ i_1 = 12$) and incremented $k = 2^{\frac{\log N}{2}}$ times by modulo operation (e.g., $j_0 = (j_0+1) mod\ \ k$), thus each loop has a counter which is equal to k.

In our algorithm, an ordering of the subindices corresponds to two nested loops which together generate a cyclic code for all the possible values of a multi-index. Since each value of a multi-index corresponds to a unique node, the result is a hamiltonian cycle. Thus in the algorithm there are two set of nested loops defined by (i_0, i_1) and (i_1, i_0), to print the values of the two multi-indices, and the algorithm runs in time $O(N)$.

For example, in Figure 8 we demonstrate two edge-disjoint hamiltonian cycles for the 64-Node hypercube generated by our algorithm. The multi-index consists of two subindices (i_0, i_1). Each subindex is mapped to 3 bits of the 6 bit ID of a node and thus has decimal range between 0 and 7.

Hamiltonian Circuit 1 generated by the indices (i_0, i_1):
(0,0) (1,0) (2,0) (3,0) (4,0) (5,0) (6,0) (7,0) (7,1) (0,1) (1,1) (2,1) (3,1) (4,1) (5,1) (6,1) (6,2) (7,2) (0,2) (1,2) (2,2) (3,2) (4,2) (5,2) (5,3) (6,3) (7,3) (0,3) (1,3) (2,3) (3,3) (4,3) (4,4) (5,4) (6,4) (7,4) (0,4) (1,4) (2,4) (3,4) (3,5) (4,5) (5,5) (6,5) (7,5) (0,5) (1,5) (2,5) (2,6) (3,6) (4,6) (5,6) (6,6) (7,6) (0,6) (1,6) (1,7) (2,7) (3,7) (4,7) (5,7) (6,7) (7,7) (0,7)

Hamiltonian Circuit 2 generated by the indices (i_1, i_0):
(0,0) (0,1) (0,2) (0,3) (0,4) (0,5) (0,6) (0,7) (1,7) (1,0) (1,1) (1,2) (1,3) (1,4) (1,5) (1,6) (2,6) (2,7) (2,0) (2,1) (2,2) (2,3) (2,4) (2,5) (3,5) (3,6) (3,7) (3,0) (3,1) (3,2) (3,3) (3,4) (4,4) (4,5) (4,6) (4,7) (4,0) (4,1) (4,2) (4,3) (5,3) (5,4) (5,5) (5,6) (5,7) (5,0) (5,1) (5,2) (6,2) (6,3) (6,4) (6,5) (6,6) (6,7) (6,0) (6,1) (7,1) (7,2) (7,3) (7,4) (7,5) (7,6) (7,7) (7,0)

Figure 8: Two edge-disjoint hamiltonian circuits of 64-Node hypercube

3.2.3 Hamiltonian ϵ-Decomposition

In this section we introduce a fast algorithm to find $\frac{d}{2}$ HCs on a *d_dimensional* hypercube for any even d. The output of the algorithm is not a proper decomposition, since some number $(M' \subset M)$ of the total number of (M) edges will not be included into any HC. Let $\epsilon = \frac{M'}{M}$, and note that these edges are excluded since some other edges are included into more than one hamiltonian cycles. Consequently, the algorithm does not yield to mutually edge-disjoint hamiltonian circuits (HCs); however, it constructs $\frac{\log N}{2}$ HCs in time $O(N \log N)$ with a small ratio (i.e., $\epsilon \leq 0.1$) of common edges, thus called hamiltonian ϵ-decomposition of the hypercube. First, we present the hamiltonian ϵ-decomposition of the hypercube, and then compute the ratio ϵ empirically.

Our algorithm algorithm takes a 2×2 array as its *building block*, visits all the nodes on this block, and then jumps to a neighboring block. The algorithm has two preprocessing steps: (i) construction of decimal multi-indices, and (ii) construction of a Latin square [2].

The multi-index—defined with respect to the *d_bit* binary string ID of each node— is used for the traversal, and constructed from the collection of $\frac{\log N}{2}$ subindices such that the subindex

i_k is obtained from b_n, b_{n+1} bits of the ID. For example, a node of the 8-dimensional hypercube with binary string 11010010 will be mapped to an multi-index $(i_0, i_1, i_2, i_3) = (2, 1, 0, 3)$ where i_k is a subindex.

In our algorithm (i) each subindex is associated with a unique loop variable (i.e., j_0, j_1, j_2, j_3), (ii) each loop variable is initialized to the corresponding subindex (e.g., $i_0 = 2$) and incremented four times by modulo operation (e.g., $j_0 = (j_0 + 1) mod \;\; 4$), thus each loop has a counter which is equal to four. An ordering of the subindices defines a nesting of these loops. For example, let i_3, i_1, i_0, i_2 an arbitrary ordering of the subindices, then there will be four nested loops (each with four iterations) such that the loop associated with subindex i_3 will be the innermost loop.

Consider an algorithm which takes an ordering of the subindices, and prints the values of all the loop variables (i.e., the value of the multi-index) at each iteration. First note that each output line of the algorithm corresponds to the ID of unique a node, and when the algorithm terminates the result will be a cyclic code, representing a hamiltonian cycle. Thus, by choosing $\frac{\log N}{2}$ different ordering of the subindices, it is possible to obtain $\frac{\log N}{2}$ hamiltonian circuits. The next question is how to choose such an ordering of the subindices to minimize number of common edges among these hamiltonian circuits. Our solution is based on construction of a Latin square (LSQ) [2] from the subindices. For example in Table 1 we show a LSQ for the subindices of 12-dimensional hypercube. Each column (row) in this Latin square gives a unique ordering of nested loops of these subindices and can be used to generate a distinct hamiltonian circuit. A subindex with a smaller row (column) number in the Latin square is nested into the index with a larger row (column) number, thus it is incremented faster. For example, the first loop in Table 1 induces a hamiltonian circuit by first incrementing i_1 then i_6 and so on.

H. Cycle 1	H. Cycle 2	H. Cycle 3	H. Cycle 4	H. Cycle 5	H. Cycle 6
i_1	i_2	i_3	i_4	i_5	i_6
i_6	i_1	i_2	i_3	i_4	i_5
i_5	i_6	i_1	i_2	i_3	i_4
i_4	i_5	i_6	i_1	i_2	i_3
i_3	i_4	i_5	i_6	i_1	i_2
i_2	i_3	i_4	i_5	i_6	i_1

Table 1: A Latin square for 12-dimensional hypercube

3.2.4 Computation of Congestion and the Ratio ϵ

An edge is *congested* if it is included into more than one hamiltonian cycles. Let $Z_{i,j}$ denote the congestion (i.e., multiple occurrences of an edge (i, j)), and Z_{max} be the maximum congestion over all edges. Note that if the $Z_{i,j} = 0$ for all edges (i, j) then a hamiltonian decomposition is achieved. However, if $Z_{i,j} = k$ for a positive integer k, the edge (i, j) replaces some edge on the k hamiltonian cycles. Thus, the value of congestion determines the value of ϵ. In what follows we consider the hypercubes up to $N = 2^{16}$ nodes, and show empirically that $\epsilon \leq 0.10$.

N: # of nodes	M: # of edges	M': # of excluded edges	$\epsilon = \frac{M'}{M}$	# of HCs	Z_{Max}
4	4	0	0%	1	0
16	32	0	0%	2	0
64	192	9	4.6%	3	1
256	1024	36	3.5%	4	1
1024	5120	252	4.9%	5	3
4096	24576	1047	4.2%	6	3
65536	524288	32760	6.2%	8	6

Table 2: Edges (M') that are not included into any HC

In our simulations, first we generate the hamiltonian cycles, by mapping the *d_bit* binary string ID of each node onto a *multi-index* and constructing a Latin square as explained before. Then a procedure is activated to count the number of common edges (*congested edges*) to get the ratio ϵ (shown in column 4) over the total number of edges. Evidently the maximum congestion is at most one less than the number of hamiltonian cycles (i.e., $\frac{\log N}{2} - 1$), and the congested edges are approximately 5-7% of the total edges. In Table 3 we show in detail the value of the congestion. The columns of this table indicate the congestion values and the number of edges with these values. For example, the second column in this table gives the number of the edges that occur in exactly one HCs while the last column shows the number of edges appearing in 7 different HCs.

Note that almost half of the congested edges have a congestion of two which is much less than the maximum congestion. The following claim presents the summary of the results:

Claim 4 *The algorithm Hamiltonian ϵ-Decomposition outputs $\frac{\log N}{2}$ hamiltonian cycles in time $O(N \log N)$ and the maximum congestion on any edge is at most $\frac{\log N}{2} - 1$.*

Proof: Since there are $\frac{\log N}{2}$ rows in the Latin square, the number of HCs holds. Each Latin square row gives a HC since the columns correspond to Gray coding. The number of indices

N: # of nodes	$Z = 0$	$Z = 1$	$Z = 2$	$Z = 3$	$Z = 4$	$Z = 5$	$Z = 6$	Avg. Congestion
64	183	9						1.047
256	988	36						1.035
1024	4868	237	3	3				1.049
4096	23529	900	60	9				1.042
65536	491528	18432	4608	1152	288	72	24	1.016

Table 3: Congestion analysis showing number of edges at each congestion level in sample Hamiltonian-pseudo-decomposition. Average congestion in all cases is ≤ 1.05.

associated with each node is $\frac{\log N}{2}$. Since the edge that is used to arrive at a node cannot be used to leave the same node there are exactly $\frac{\log N}{2} - 1$ candidate edges for each HC to leave this node. Thus an edge can be used by at most $\frac{\log N}{2} - 1$ hamiltonian cycles to leave this node. Now let us compute the time complexity. The Latin square has $\frac{\log N}{2}$ columns and each has $\frac{\log N}{2}$ nested loops. Each loop has a counter $0, \cdots, 3$ (i.e., 2^2 iterations). Thus, each HC costs $2^{2(\frac{\log N}{2})} = O(N)$. Since we generate $\frac{\log N}{2}$ HCs, the total time is $O(N \log N)$.$\square$

4 Routing and Flow Control

Routing and flow control on the networks with multiple global virtual ring embeddings is based on *convergence routing* of the MetaNet [16, 18]. Convergence routing is a variant of *deflection routing* [4, 14]. It combines, in a dynamic fashion, the on-line routing decision with the traffic load inside network. For example, the routing decision may depend on flow-control aspects like whether or not the serial transmission link is idle. Deflection routing typically minimizes intermediate buffering requirements and it suffers no loss due to congestion inside the network.

Convergence routing is based on *global sense of direction* such that the packet or cell distance from its destination, at the next node, strictly decreases compared to its current distance. It achieved by numbering the virtual nodes on the embedded rings sequentially. Convergence routing differs from previous deflection techniques since it assures that: (i) a packet reaches its destination (unless a failure occurs), and (ii) a packet can traverse each direction of a link at most once, and the number of hops it takes is bounded deterministically.

The packet enters the virtual embedded ring via its virtual node that is closest to the destination. The **Default** routing decision in convergence routing is simply to follow the virtual ring, which will guarantee that the packet will reach its destination. This simple method is, of course, not very efficient. Therefore, the routing mechanism at every intermediate node tries to decrease the distance to the destination - while preserving the global sense of direction - as much as possible.

Inside the network a non-default routing operation (called **Jump**) may be used for switching packets from one virtual ring to another in order to obtain faster convergence to the destination [26]. This kind of switching is possible only if the next ring link, which is reachable by the **Jump** operation, is *available*. A link is defined to be available if it is: (i) idle, and (ii) not *marked* by another packet as its default link. A link is *marked* in order to avoid congestion and loss, which means that a packet has priority over all packets from other ring links for using its default ring link. This priority is equivalent to the *ring traffic priority in buffer insertion rings*.

More specifically, routing of a packet at some node U is based on computing distances from U to the final destination, say V, on the d embedded virtual rings. Thus, each intermediate node U computes a distance vector $\Delta_V(U)$ for a packet destined to node V. This vector contains all the virtual nodes that are closer to the destination and can be accessed from node U, using a jump. The distance from the i'th element u_i of $\Delta_V(U)$ is denoted by δ_i and computed as: $\delta_i = v_i - u_i (mod\ m)$, where $m = N$ is the size of the virtual ring on which both $v_i \in VID(V)$ and $u_i \in \Delta_V(U)$ resides. ($VID(V)$ is the subset of node V virtual nodes). If u_i and v_j are on different rings then the distance between them is defined as $+\infty$. The δ_i values are sorted in increasing order and higher priorities are given to the outgoing links of a virtual node with smaller δ_i value (i.e., those that are closer to the destination).

For example, consider Figure 2 and suppose that node C on ring HER0 receives a packet designated for node I. The default route of the packet is to VN_3 on the virtual ring HER0. If the packet remains on the HER0, the length of routing would be six hops. However, a **Jump** operation to the VN_{17} on virtual ring HER1 would decrease the distance to two hops, and if another **Jump** were possible from HER1 to HER2, the packet would reach to the destination via virtual node VN_{23}.

4.1 Controlling the Probability of Jumps

In the MetaNet the asynchronous access is regulated by a predefined **quota** given to each node in every **global** or **local** fairness cycle (see [25] for details). The size of the **quota** controls the internal flow in the network and determines the *availability* of a link or alternatively the probability P for a **Jump** operation. High **quota** increases the number of packets in the network which in turn increases the busy period of a link, and therefore, decreases the probability P for a **Jump**. Thus, the external access **quota** is the control parameter for the MetaNet routing and internal flow control.

4.2 Fault-Tolerant Properties of Convergence Routing

In this section, we explain how to route in case of broken rings due to link/node failures, and show that the multiple virtual rings can tolerate (i) single node failure, and (ii) single link failure on each virtual ring. We argue that the routing algorithm presented in this section efficiently handles the faults by the information (1) provided in the header of a packet, (2) and maintained in the virtual node that receives this packet.

Link failures

In case of a link failure only one hamiltonian circuit (i.e., two virtual rings) is broken and on this ring some nodes becomes inaccessible to the others. Suppose the link between the node A and B goes down (see Figure 2). Then the virtual rings HER0 and HER1 are broken (since these are the only virtual rings that includes one direction of the faulty full duplex link). In order to determine which virtual nodes become inaccessible upon an edge failure, it is sufficient for each virtual node just to know the closer end point of the faulty link. For example, VN_6 can compute in a constant time the inaccessible segment $[VN_1, \cdots, VN_5]$ of the broken ring by knowing that VN_0 is the closer end point of the faulty link on ring HER0.

Note that these are nodes that reside between the other end point of the faulty link (i.e., VN_1) and the down-stream neighbor of this virtual node on HER0. Therefore upon being informed about a failure, each virtual node on the broken ring performs the following steps to ensure that a packet is not sent to an inaccessible segment:

1. *information needed in the nodes:* each node stores the closer end point of the faulty link to determine the inaccessible nodes.

2. *distance computation in the routing algorithm:* at step 2 of the routing algorithm, the distance to an inaccessible node is considered to be $+\infty$ to ensure that the packet will not be sent to that node.

Node failures

In case of a node failure all the virtual rings are broken into paths. For example, consider the virtual rings in Figure 2 and suppose the node A has a failure then each virtual ring is broken into a directed path. Since each virtual ring has a dual, each path has a reverse, thus together they induce a dual bus structure. Since, a node failure causes two adjacent links to be faulty, it can be treated similar to the single link failure on each virtual ring. Therefore, upon being informed about a node failure, each node performs the steps stated above.

5 Bounds on The Length of Routing

5.1 Worst Case: Average Pairwise Distance on d Virtual Rings

One of the objectives of multiple virtual ring embedding is to minimize the expected length of the routing under heavy traffic conditions. In such a case, the probability for a **Jump** operation is small, thus the packets likely to follow the longest paths. In this section we assume the worst case traffic pattern, in which each source has $N-1$ packets and each destination receives $N-1$ packets, thus no **Jump** operation is possible. Therefore, we consider the pairwise distance, over d global virtual rings, and establish a bound on the worst-case length of the routing.

One of the objectives of multiple virtual ring embedding is to minimize the expected length of the routing under heavy traffic conditions. In such a case, the probability for a **Jump** operation is small, thus the packets likely to follow the longest paths. In this section we assume the worst case traffic pattern, in which each source has $N-1$ packets and each destination receives $N-1$ packets, thus no **Jump** operation is possible. Therefore, we consider the pairwise distance, over d global virtual rings, and establish a bound on the worst-case length of the routing.

Suppose that a generic node can access $\lceil \frac{N}{d} \rceil$ disjoint sets of nodes at each global ring. Then the total distance from this generic node to the others in one ring is

$$\sum_{i}^{\frac{N}{d}} \approx \frac{N^2}{2d^2}$$

Thus yielding to the average distance (from a generic node to all the others) as $\frac{N}{2d}$.

However, the set of $\lceil \frac{N}{d} \rceil$ nodes at each ring are not necessarily disjoint. The union of these sets can not cover all the nodes. Let p denote the fraction of the nodes that are "uncovered". Thus $(1-p)N$ of the nodes will have distance from a generic node at most N/d. This is on the average at most $\frac{N}{2d}$. The rest of the nodes will have distance between $\frac{N}{d}$ and $\frac{N}{2}$ on each ring which is on the average is $\frac{N(d+2)}{4d}$.

Therefore, in order to have a bound from a generic node to all others of $\frac{N}{d}$, we solve the following equation for p:

$$\frac{N}{2d}(1-p) + \frac{N(d+2)}{4d}p \leq \frac{N}{d}$$

which has a solution for $p \leq \frac{2}{d}$. Consequently, as the number of virtual rings (thus the degree of each node) increases for the same N, the ratio p becomes negligible. Although in the hypercube N and d are mutually dependent, these parameters in the circulant networks are flexible. Thus d can be increased as long as new *jumps* which are relatively prime to N and less than $N/2$ can be found [8]. Thus we conclude that

Claim 5 *Pairwise average distance on the networks with d global virtual rings is bounded by $O(\frac{N}{d})$.* □

5.2 Best Case: Minimum Diameter Circulant Design

In this section we consider circulant networks which have minimum diameters. The diameter of the circulant network implies a bound on the maximum length of routing, if the network is lightly loaded and **Jump** operations are always possible. Although the problem of finding the diameter of a circulant is difficult, there is theorem [9] which enables us to determine the minimum diameter among all circulants obtained with two jumps on $N > 6$ nodes. Precisely, if the two jumps are $m, m+1$ then for $m = \lceil \frac{-1+\sqrt{2N-1}}{2} \rceil$ the minimum diameter, among all circulant graphs with two jumps on N nodes, is m.

If m and $m+1$ are both relatively prime to N, then we know that the corresponding rings are hamiltonian. Thus, the circulant based network with N nodes, each with degree exactly 4, would have the diameter approximately $\lceil \frac{\sqrt{2N}}{2} \rceil$. However, since the value of m depends on N, it may not be possible to have both of the jumps to be relatively prime to N. As a result, the connectivity and hamiltonian property of the circulant network can not be ensured, since the edges created by such jumps will induce node-disjoint partial rings. Therefore we must have one of the jumps as relatively prime to N. Note that m grows much slower than N (see Table 4) that implies that length of routing increases in slower rate than the size of network grows. Furthermore, for $N = 2^k$ ($k = 3, \cdots$) one of the jumps is relatively prime to N. Thus, if N is a power of 2 then it is ensured that there exists a hamiltonian cycle in the circulant network with N nodes and degree 4. (As a result of this observation, a hybrid approach, which takes a combination of global and partial rings becomes a promising design paradigm.)

We note that this bound on the diameter of the network is better than the ones on the loop topologies with the same degree (such as the daisy chain, double loop, etc.) reported in [19, 20, 7]. This property becomes more remarkable with a comparison to another regular graph of similar size. For example, a circulant with 25 nodes each with degree 4 has the same diameter 3 with a De Brujin graph with 27 nodes. However, the degree requirement on each node on the

N	m	$m+1$
8	1	2
16	3	4
32	4	5
64	6	7
128	8	9
256	11	12
512	16	17
1024	23	24
2048	32	33

Table 4: The m, $m+1$ values for the jumps to obtain diameter m for $N = 2^k$ node MetaNet

De Brujin graph is close to 6, thus more links are necessary to achieve the same diameter [9]. Now let us determine a bound which is valid for various values of degree d.

6 Performance Study

The performance measure considered in this paper is the average number of hops assuming: (i) uniform destination distribution, and (ii) heavy load traffic conditions, which means that inside the network packets are sent over the longest possible paths (i.e., worst case analysis).

Therefore note that in the worst case, only the **Default** routing operations are considered to be possible, since the probability of finding an idle link would be very low. The expected number of hops can be used as a measure of the potential throughput of the network (rather than the actual throughput). In our simulations, we consider the following values of P to capture the *availability* of a link which in turn reflects the internal load of the network:

1. The worst case: P=0. The network is operating under heavy traffic conditions such that each source sends $N-1$ packets and each destination receives $N-1$ packets. In this case only the **Default** operations are taken during the execution of the algorithm.

2. The average case: P=0.5.

3. The best case: P=1. The network is assumed to be very lightly loaded and the **Jump** operations are favored.

In the worst case scenario no **Jump** operation is possible. Thus once a packet enters to a virtual ring, it remains in that ring until it is delivered to the destination. This implies that permutation of the nodes over the virtual rings to minimize the pair-wise distance becomes an important performance issue.

6.1 Experimental Results

In this section we present simulation results to measure the performance of the convergence routing on the circulant networks. The algorithm is simulated by first determining the closest virtual node of the source to the destination V. At each intermediate node U, which receives the packet, the distance vector $\Delta_V(U)$ and NEXT are constructed. If the link to a node in NEXT which has the minimum distance to the destination, can be taken by the **Default** operation then the packet is sent to the next ring link. However, if the routing operation is a **Jump**, then it is taken with probability P. If the first choice link is not available then the link with the second smallest distance is considered and this process continues until the next candidate link is via the **Default** operation. The algorithm halts when the destination node is reached. Performance of the convergence algorithm is analyzed both as a function of the number of nodes N and the degree constraint d. Furthermore, different P values are considered to capture the internal load in the network.

		Average Number of Hops		
DEGREE	n_i values	$P = 0$	$P = 0.5$	$P = 1$
4	1,7	21.54	11.09	9.06
6	1,7,13	16.15	7.56	5.89
8	1,7,13, 17	13.51	6.39	5.332
10	1,7,11,13,17	10.86	5.68	5.01
12	1,7,11,13,17, 19	9.32	5.13	4.50
14	1,7,11,13,17,19,23	8.09	4.72	4.25
16	1,7,11,13,17,19,23,29	7.21	4.26	3.78

Table 5: Average Length of Routing on a Network w/ 128 Nodes

	Average Number of Hops		
# of Nodes	$P = 0$	$P = 0.5$	$P = 1$
16	3.20	2.80	2.53
32	5.54	4.21	3.47
50	8.51	6.19	6.61
64	10.92	7.00	6.62
128	21.54	11.09	9.06
256	42.85	19.35	17.01
512	86.17	37.61	31.39

Table 6: Average Length of Routing on a Network w/ degree 4 and $n_1 = 1, n_2 = 7$

Experimental bounds on the expected length of the routing

First we explain Table 5, in which the performance of the convergence routing is studied for the synthesized networks with 128 nodes, as a function of the degree d of each node. The jump sequence $< n_i >$ (i.e., a set of distinct jumps) used for construction of each network is shown in the second column. The jumps are prime numbers and chosen according to the claim 1, (i.e., each jump is relatively prime to 128). In Table 6, the degree of the synthesized network is fixed as 4 and the number of the nodes is increased.

The average number of hops on each network is computed for three different probability values to capture the heavy, average and light traffic conditions, respectively. The results in these tables lead to the following two remarks:

1. the performance difference of the algorithm under the heavy and the light traffic conditions (i.e., $P = 0, P = 1$) is not too large. This demonstrates that the combinatorial design of the virtual rings minimizes the performance degradation of the routing algorithm under heavy internal load.

2. the worst case bound on the length of routing is empirically verified as approximately N/d which is consistent with our analytic bound. In order to make this remark more visible, in Figure 9, we plotted the analytical computations and the first column of the Table 6.

3. Finally we note that the performance of convergence routing under worst case traffic conditions is similar to the hypercube (i.e., $O(N)$), while under light traffic conditions it approximates to the diameter of the network.

7 Summary and Discussions

In this paper we presented tools to embed multiple hamiltonian rings with convergence routing for loss-free routing of non-reserved bursty traffic. Convergence routing with virtual rings is suitable for ATM LANs with bursty data sources. This will enable the use of ATM LANs for high performance distributed/parallel processing applications.

We considered two regular topology networks. The first structure is based on the *circulant* design, and can be applied to any d and N provided that $d/2$ is even, and each construction step is relatively prime to N. The second structure we examined in this paper was the *hypercube* with even dimension. We have shown how to design a linear time algorithm to obtain two edge-disjoint hamiltonian circuits (HC) of any even degree hypercube. Furthermore, we presented an ϵ-decomposition in which a small percentage of the edges occur in more than one HCs.

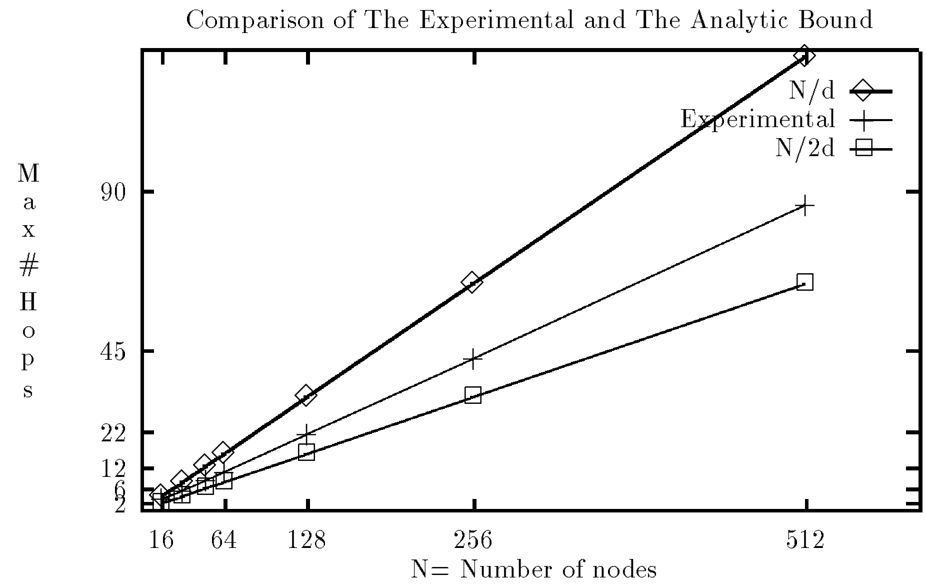

Figure 9: Average Length of Routing for $P = 0$, $d = 4$

We described how multiple virtual ring embeddings can provide high transmission concurrency and fault-tolerance. We showed how convergence routing can be employed on these regular topologies to obtain good bounds on the routing length, while ensuring loss-free flow control.

Acknowledgements

We would like to thank Moti Yung for his valuable comments on the earlier versions of this work.

References

[1] B. Alspach, J.-C. Bermond, and D. Sotteau. *Cycles and Rays*. Kluwer Academic Publishers, Hahn et al. (eds.), Netherlands, 1990.

[2] I. Anderson. *Combinatorial Designs: Construction Methods*. John Wiley Sons, New York, 1990.

[3] J. Aubert and B. Schneider. Decomposition de la somme cartesienne d'un cycle et de l'union de deux cycles hamiltoniens en cycles hamiltoniens. *Discrete Math.*, 38:7–16, 1982.

[4] P. Baran. On distributed communication networks. *IEEE Trans. on Communications Systems*, CS-12(1-2):1–9, March 1964.

[5] J. Bosak. *Decompositions of Graps*. Kluwer Academic Publishers, Boston, 1988.

[6] I. Cidon and Y. Ofek. MetaRing - a full-duplex ring with fairness and spatial reuse. *IEEE Trans. on Comm.*, COM-41(1):110–120, January 1993.

[7] J.A. Silvester C.S. Raghavendra. A survey of multi-connected loop topologies for local computer networks. *Computer Networks and ISDN systems*, 11:29–42, 1986.

[8] Buckley F. and Harary F. *Distance in Graphs*. Addison-Wesley., New York, 1990.

[9] J.Wang F.Boesch. Reliable circulant networks with minimum transmission delay. *IEEE Transactions on Circuits and Systems*, CAS-32:1286–1291, 1985.

[10] M.F. Foregger. Hamiltonian decompositions of product of cycles. *Discrete Math.*, 24:251–260, 1978.

[11] Krizanc D. Kaklamanis C. and Thanasis Tsantilas. Tight bounds for oblivious routing in the hypercube. *Math. Systems Theory*, 24:223–232, 1991.

[12] A Kotzig. Every cartesian product of two circuits is decomposable into two hamiltonian circuits. *Preprint, Univ. de Montreal, Montreal*, 1973.

[13] T. Leighton. *Parallel Algorithms*. Morgan Kaufmann, San Mateo, CA., 1992.

[14] N. F. Maxemchuk. Routing in the manhattan street network. *IEEE Trans. on Communications*, COM-35(5):503–512, May 1987.

[15] Y. Ofek. Overview of the MetaRing architecture. *Computer Networks and ISDN Systems*, 6-8:817–830, 1994.

[16] Y. Ofek and M. Yung. Principles for high speed network control: lossless-ness and deadlock-freeness, self-routing and a single buffer per link. *9-th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 161–175, August 1990.

[17] Y. Ofek and M. Yung. The integrated MetaNet architecture: A switch-based multimedia LAN for parallel computing and real-time traffic. *IEEE INFOCOM'94*, 1994.

[18] Y. Ofek and M. Yung. Routing and flow control on the MetaNet: an overview. *Computer Networks and ISDN Systems*, 6-8:859–872, 1994.

[19] C.S. Raghavendra, M. Gerla, and D.S. Parker. Multi-connected loop topologies for local computer networks. *INFOCOM 82*, pages 184–190, March 1982.

[20] C.S. Raghavendra, M. Gerla, and A. Avi zienis. Reliable loop topologies for large local computer networks. *IEEE Trans. on Computers*, C-34, No. 1:46–55, January 1985.

[21] G. Ringel. Uber drei kombinatorische probleme am n-dimensionalen wurfel und wurfelgitter. *Abh. Math. Semi Univ. Hamburg*, 20:10–19, 1955.

[22] Rosen.K.H. *Elementary Number Theory and Its Applications.* Addison-Wesley, New York, 1993.

[23] J. Turner. New directions in communications (or which way to the information age?). *IEEE Communications Magazine*, 24(10), October 1986.

[24] B. Yener, T. Boult, and Y. Ofek. Hamiltonian decompositions of regular topology networks for convergence routing. *Technical Report CUCS-011-94, Computer Science Dept., Columbia University*, 1994.

[25] B. Yener, Y. Ofek, and M. Yung. Design and performance of convergence routing on spanning trees. *Proc. IEEE GLOBCOM'94*, 1994.

[26] B. Yener, Y. Ofek, and M. Yung. Topological design of loss-free switch-based lans. *To appear in IEEE INFOCOM'95 (available as IBM Research Report RC 19649 (87112))*, 1994.