Combinatorial Design of Congestion-Free Networks

Bülent Yener, Member, IEEE, Yoram Ofek, Member, IEEE, and Moti Yung

Abstract—This paper presents novel design methods, based on combinatorial design theory, to design a packet switched network with bursty traffic sources. The network design combines two important properties for arbitrary traffic pattern: 1) the aggregate throughput is scalable and 2) there is no packet loss within the subnet.

More specifically, given a bounded number of ports in every switching node, the design is based on the construction of multiple virtual rings under the following constraints: 1) the virtual rings are pairwise edge-disjoint and 2) there is at least one virtual ring between any pair of nodes. The target topology is obtained from the edge union of the multiple virtual rings. The two constraints ensure no loss due to congestion inside a network with arbitrary traffic pattern and that packets will reach (or converge) their destinations.

The virtual rings are constructed by using combinatorial block designs together with an algorithm for realizing any size networks. It is shown that the bound on the maximum route length, under the two constraints, is \( O(\sqrt{N}) \) for an \( N \)-node network. This sublinear bound facilitates the throughput scalability property.

I. INTRODUCTION

This paper presents novel design methods, based on combinatorial design theory, to design a packet switched network with bursty traffic sources. The design ensures that under arbitrary traffic pattern the throughput increases by adding links and switches (i.e., it is scalable), while ensuring that: 1) it is congestion-free and 2) packets will reach their destination (i.e., it guarantees convergence).

Current network designs do not provide these two properties. For example, the approaches based on a simple linear topology, such as a bus or a ring [5], [7], [2], are congestion-free but network throughput is not scalable. In contrast, network design based on an arbitrary topology has a scalable throughput but it is not congestion-free. Our objective is to develop network design tools to obtain congestion-free networks with scalable throughput.

Our solution is based on synthesizing a network from linear structures, i.e., virtual rings, which are used to linearize the network in order to ensure no packet loss due to congestion. The design techniques presented in this paper are based on combinatorial design theory. In particular, we make use of a block design which considers collection of unordered subsets (i.e., blocks) of a set. The blocks are chosen such that certain combinatorial properties among them are ensured.

The techniques result in networks on which the aggregate throughput is \( O(\sqrt{N}) \) and the maximum length of routing is bounded by \( O(\sqrt{N}) \)—independent of the traffic characteristics (where \( N \) is the number of nodes in the network). We note that previous approaches to multiple ring topology design (e.g., chordal rings [10], [3]) report designs with similar bounds on the routing length; however, it is achieved by compromising congestion-free routing property.

This paper is organized as follows. Section II explains the network model and the design paradigm. In Section III, two types of virtual ring designs are presented: 1) global (spanning) multiple virtual rings (where each virtual ring spans all the nodes of the network) and 2) partial multiple virtual rings (where each ring includes only a subset of the nodes). Global rings improve fault tolerance while partial rings simplify routing and ensure first-in first-out (FIFO) delivery of the packets. In Section IV, we derive a \( O(\sqrt{N}) \) bound on the maximum length of routing for both global and partial ring embeddings.

Direct application of the combinatorial design techniques produce networks with a limited number of nodes. Therefore, Section V introduces a scaling algorithm, based on algebraic operations over combinatorial designs, to obtain any size network without increasing the degree requirement (number of ports) of each switch. Section VI identifies and discusses some tradeoff issues in scaling. In Section VII, some of the fault-tolerance properties are discussed. The work is summarized in Section VII.

II. MODEL AND BACKGROUND

A. The Network Model

The network is represented by an undirected and connected graph \( G = (V, E) \) where the vertex set \( V \) contains \( N \) network nodes and the edge set \( E \) represents the bidirectional communication links between the nodes. Each node has a unique ID associated with an integer (i.e., \( A = 0, B = 1, \ldots \)).

There are two important properties that the network graph possesses by design: 1) it is synthesized from a collection of virtual rings and 2) there is (at least) one virtual ring between any pair of nodes. A virtual ring is a directed cycle that connects \( N' \subseteq N \) of the nodes (e.g., Fig. 1). The first property states that the edge set of the synthesized network graph can be partitioned into subsets such that each subset induces a virtual ring and the rings do not share any links (edges) (i.e., it is decomposable to cycles). For example, the network shown in Fig. 2 is decomposable to rings \( R_0, R_1, R_2, \) and \( R_3 \).
The second property provides the connectivity required by the routing algorithm to ensure convergence and delivery.

The properties of the target network are guaranteed by design. We first design the virtual rings using the combinatorial block design theory [1], [6], [11], and then perform a set-union operation on the virtual rings to construct the network. Thus, the network in Fig. 2 is indeed obtained from the union of the virtual rings.

The routing algorithm is designed to operate on the virtual rings. Thus, each time a virtual ring visits a node, a new virtual node is induced. Each virtual node corresponds to an input port and an output port of a switch. Thus, each node in the target network has one (or more in case of partial rings) virtual node on the rings. The assignment of virtual node numbers must capture the direction of the two rings as well as the location of the virtual nodes on them. 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) and \( l \) (to indicate its offset which is the distance from a predefined origin on the ring 0,1,\ldots,\( N-1 \)). (We note that in case of partial rings the range of the index \( l \) is approximately \( 2\sqrt{N} \).) The computation of the virtual node label \( VN_j \) is simply \( r \times N + l \). For example, in Fig. 1 node C has virtual address \( \{VN_2, VN_{11}, VN_{24}, VN_{33}\} \) assuming that node A is the origin\(^1 \) of each ring. Since in this construction the target network is regular (i.e., each node has the same degree), the virtual address of each node can be represented as a \( d \)-field array, where \( d \) is the degree of each node.

**B. Background: What is BIBD?**

In this section we present some background on combinatorial block design that we use to design networks. A block is simply a subset of a set, say \( S \), and block design considers choosing the blocks with certain properties. A block design is called incomplete if at least one block does not contain all of the set elements. A block design is balanced if each block has the same number of elements and each pair of elements occurs in a block the same number of times. In this work we consider balanced incomplete block design (BIBD) to design networks with guaranteed (worst-case) properties.

**Definition 1:** A BIBD is a collection of \( k \)-element subsets (called blocks) of an \( N \)-element set \( S \), \( k < N \), such that each pair of elements of \( S \) occur together in exactly \( \lambda \) of the blocks.

**Theorem 1:** In a \( (N, b, r, \kappa, \lambda) \) BIBD with \( b \) blocks, each element occurs in \( r \) blocks where

\[
\begin{align*}
bk &= Nr, \\
\lambda(N - 1) &= r(k - 1).
\end{align*}
\]

**Proof:** It is based on different ways of counting. To prove 1), note that each block contains \( k \) elements which sums up to \( bk \). On the other hand, each element occurs in \( r \) blocks and since there are \( N \) elements the total yields \( Nr \). To prove 2), consider an element, say \( x \), which occurs in \( r \) blocks. Note that each one of these blocks \( x \) will be paired with \( k - 1 \) elements. Thus, the counting yields \( r(k - 1) \) pairs. However, \( x \) is paired with all other nodes (there are \( N - 1 \) of them) exactly \( \lambda \) times (similar proofs can be found in standard references on combinatorial design theory, e.g., [1], [11]).

Note that once \( N, k, \lambda \) are known, due to Theorem 1, \( b \) and \( r \) can be computed easily. Thus, a BIBD can be characterized by the triplet \( (N, k, \lambda) \). Furthermore, due to Fisher’s inequality

\[ b \geq N \]

designs with \( b = N \) and \( r = k \) are called symmetric designs and they have the following properties: 1) every block contains \( k \) elements; 2) every element occurs in \( r \) blocks; 3) every pair of elements occurs in \( \lambda \) blocks; and 4) every pair of blocks intersects in \( \lambda \) elements. In this paper we consider BIBD’s based on: 1) projective and 2) affine planes defined as follows [1], [6], [11].

**Definition 2:** A finite projective plane of order \( \eta \) is an \( (n^2, n + 1, 1) \) design \( n \geq 2 \).

It is well known in combinatorial design theory that BIBD’s based on a finite projective plane exists for all prime numbers \( \eta \) and projective planes are symmetric designs. For example, the block design with (see Fig. 3) parameters \( (13,4,1) \) (i.e., \( \eta = 3 \)) is a symmetric one with 13 elements and 13 blocks. It

\(^1\)The global ID of C’ virtual nodes are computed as \( 2 = 2 + (0 \times 9) \) on R0; \( 11 = 2 + (1 \times 9) \) on R1; \( 24 = 6 + (2 \times 9) \) on R2; \( 33 = 6 + (3 \times 9) \) on R3.
is easy to check that the properties $\{\overline{1}, \cdots, \overline{m}\}$ are satisfied.

We will use a projective plane to construct partial virtual rings to design the desired target network. The second type of BIBD used in this paper is based on affine planes as defined below.

Definition 3: An affine plane of order $n$ is an $(n^2, n, 1)$ design.

Every affine plane is resolvable, which is defined as follows.

Definition 4: A BIBD is resolvable if its blocks can be arranged into $r$ groups so that $b/r = N/k$ blocks of each group are disjoint and contain in their union each element exactly once.

Thus, in contrast with projective planes, where every pair of blocks intersect on $\lambda$ elements, groups in affine planes are called parallel classes. For example (see Fig. 4), in the block design with $N = 9$ elements, there are $b = 12$ blocks each with $k = 3$ elements and each node occurs in $r = 4$ blocks. Furthermore, for any pair of elements there are exactly $\lambda = 1$ blocks. We will use affine planes to construct spanning (i.e., global) virtual rings to design the desired target network, as we shall explain in the next section.

The conditions stated in (1)–(3) are necessary for the existence of a $(N, k, \lambda)$ design, however they are not sufficient. Thus, some designs that satisfy these equations do not exist [e.g., (43, 7, 1), (15, 7, 2)]. Solutions to some other designs are unknown (e.g., (157, 13, 1)) [6]. There are standard methods for actual construction of a BIBD based on finite fields [e.g., (21, 5, 1)-BIBD can be constructed by $3 \cdot 6 \cdot 7 \cdot 12 \cdot 14 \cdot (\text{mod} 23)$. PG(2, 4)], see [6].

We note that a BIBD can be described by an incidence matrix $A$ which has the blocks as its columns and elements as the rows. An entry $a_{ij}$ of the incident matrix $A$ is equal to one if the $i$th element resides in the $j$th block, otherwise it is equal to zero. For example, for a symmetric design with $N$ elements, the incident matrix is an $N \times N$ matrix. In Section VI, we will define scaling operations based on incidence matrix to design networks with desired size.

III. NETWORK DESIGN WITH BIBD

This section shows how to apply combinatorial block design theory to network design. We assume that for a target network with $N$ nodes there is a block design with $N$ elements (in Section VI we present scaling operations which merge multiple block designs to construct any size networks). First we show how to map the parameters of a BIBD to that of a network design problem. Then we present techniques to construct the networks that satisfy the no-loss and connectivity requirements stated before.

**A. From Combinatorial Design to Network Design**

Given the blocks of a BIBD with parameters $(N, k, \lambda)$, Table I summarizes the mapping between BIBD and network design. A solution to a BIBD provides a partition of the $N$ nodes into subsets such that: 1) there are exactly $\lambda$ subsets for each pair of the nodes and 2) the distance between any two nodes is at most $k - 1$. The first property ensures that the target topology for the network is strongly connected. The second property is important to obtain a bound on the maximum routing length which is measured by the number of hops. Each block of a BIBD leads to one or two virtual rings (depending on the embedding technique as explained below), and each node of the network has one or more virtual nodes on each virtual ring. A virtual node is created when a virtual ring traverses a physical node.

Thus, the two most important parameters of a block design are $k$ and $r$ such that: 1) the size $k$ of each block determines the maximum length of routing and 2) the total number of occurrences $r$ of each node determines the degree requirement for this node in the target network. Particularly, smaller $k$ leads to a better bound on the length of routing and smaller $r$ requires smaller number of ports and links on the target network.

Note that for $\lambda = 1$, each block of size $k$ is unique for all possible pairs of $k$ nodes that it contains. That implies that each possible pairing of nodes in a block corresponds to a unique candidate edge for the target topology. Furthermore, since such an edge never occurs in any other block, the virtual rings are mutually edge-disjoint. For example, consider the block design in Fig. 4 in which each block has three elements (e.g., block $B_2$ contains the nodes $E = 4, F = 5, G = 6$) and no pair of these nodes can occur in any other block. Thus, each block of size $k$ can induce a complete graph of $k$ nodes. It is known in graph theory that any graph with $k$ nodes can be embedded into a complete graph with $k$ nodes. However, in order to keep the number of links on the target graph reasonable, we consider embedding of a simple topology into each block (e.g., simple cycles or trees). Note also that it is possible to embed a different topology into each block to connect the nodes.

**B. Global Virtual Rings**

In this section we show how to construct multiple global virtual rings for an $N$ node network. Since each ring spans all the nodes, multiple global rings enhance fault-tolerant properties of the routing algorithm [13].
TABLE I

<table>
<thead>
<tr>
<th>Block Design</th>
<th>Network Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>elements</td>
<td>nodes (i.e., switches)</td>
</tr>
<tr>
<td>blocks</td>
<td>virtual rings</td>
</tr>
<tr>
<td>$\lambda$ (i.e., pairwise occurrence)</td>
<td>virtual rings between each pair</td>
</tr>
<tr>
<td>$\tau$ (i.e., total occurrence of an element)</td>
<td>degree of a node</td>
</tr>
<tr>
<td>$k$ (i.e., block size)</td>
<td>length of routing</td>
</tr>
</tbody>
</table>

1) Hamiltonian Virtual Rings: A Hamiltonian virtual ring is a simple cycle that includes each node exactly once. We use a BIBD with the parameters $(N, \sqrt{N}, 1)$ to design such rings (see [12] for different methods to construct spanning rings based on circulants). Recall that such a design has the resolvability property which provides grouping of the blocks into $\sqrt{N} + 1$ parallel classes. Resolvability ensures that: 1) each block belongs to a unique parallel class and 2) the union of elements of the blocks in a parallel class includes all of the $N$ nodes. In the network design, resolvability property is important since each parallel class of $N$ nodes can be used to construct a Hamiltonian circuit.

We proceed with an example and show how to construct a network of nine nodes, each with degree four, by using resolvable block design with parameters $(9, 3, 1)$ (see Fig. 4). The resolvable design is obtained by grouping the blocks into four parallel classes; for example, the first parallel class consists of the blocks $B_0, B_1, B_2$. Parallel classes can be constructed in polynomial time if the resolvability property exists. Recall that a parallel class must span all the nodes and contains $\sqrt{N}$ blocks each with $\sqrt{N}$ elements. Thus, pairwise intersection of blocks in a parallel class is empty. Furthermore, intersection of any two blocks must contain a common element to satisfy the conditions in Theorem 1. Consequently, a simple algorithm would work on the incident matrix of a block design and examine the columns. A column (i.e., block) would be assigned to one of the $\tau$ parallel classes (initially empty) if its pairwise intersection with the columns already in this class does not contain a nonzero element. Note that this is equivalent to determining orthogonal columns (i.e., their scalar product is zero).

The blocks of each parallel class can be ordered arbitrarily to obtain different Hamiltonian circuits. For example, the circuits in Fig. 1 are obtained by ordering the blocks within a parallel class according to the first element of each block. The undirected (bidirectional) rings are constructed in two steps: 1) connect each node within a block to its neighbors and 2) connect the last node of each block to the first node of the next block in a cyclic way. An undirected Hamiltonian circuit can be used to obtain two virtual rings in the opposite directions. For example, in Fig. 1 two virtual rings $R_0$ and $R_1$ obtained from the parallel class with blocks $B_0, B_1, B_2$ are illustrated.

2) Pseudo-Hamiltonian Virtual Rings: It is also possible to construct spanning cycles that may include each node at least once. We call such rings pseudo-Hamiltonian, since a node can be visited more than once. One technique is to build edge-disjoint spanning trees—one from each parallel class. A tree can be constructed from the blocks of a parallel class using the following algorithm: 1) build a spanning tree within each block for all the blocks and then 2) connect the subtrees within the blocks of a parallel class. Once an edge is selected it is marked so that a block of a different parallel class does not use this edge to make an edge. Since the trees are built one at a time, the algorithm is greedy; however, since the links made in step 1) cannot occur in any other block and the links in step 2) are chosen to avoid replication, the trees are distinct. Furthermore, since each block can induce $k^2$ possible links (recall that we can choose any pair to make an edge) and $k = 1$ links are needed to string the blocks in a parallel class, $O(k)$ edge-disjoint spanning trees can be built (i.e., without exhausting the links that can be produced from a block).

Once the spanning trees are constructed a virtual ring can be embedded on each tree with a directed Euler tour, which visits each direction of a full duplex link exactly once [8], [9].

Note that, in general, different subgraphs can be embedded into each parallel class and rings can be constructed by Euler tours. For example, it is possible that one parallel class has a spanning tree, another one has a hypercube, etc. However, to limit the scope of this paper we will focus on trees and simple cycles.

3) Edge Selection and Construction of the Network: There are two types of edges induced by Hamiltonian-based and pseudo-Hamiltonian-based construction of global virtual rings: 1) the edges within a block and 2) the edges between the blocks. The edges within each block are used for routing with bound $O(\sqrt{N})$. (Recall that the number of nodes in a block is $O(\sqrt{N})$.) In contrast, the edges between the blocks are used simply to close the cycle and useful for fault tolerance. Note that these edges are not necessary for routing since routing bound is ensured by the edges within a block. Consequently, edge-disjoint requirement can be relaxed for the global rings in order to use all the parallel classes.

The target network is connected (and regular for the Hamiltonian case since the degrees of all nodes will be the same) and obtained by taking the edge-union of the virtual rings. Depending on the degree constraint, only a subset of the parallel classes may be used in the design of the network. For example, for degree four, suppose we choose the first and the last parallel class, then the virtual rings would be the ones shown in Fig. 1 and the synthesized network would be the one in Fig. 2.

C. Partial Virtual Rings

In this section we consider partial virtual rings such that each ring has the same size and includes only a distinct subset of the nodes. However, the design technique presented in this section ensures that there is a virtual ring between any pair of the nodes and the rings are mutually edge-disjoint. The partial ring design is scalable and supports FIFO discipline at the receiving nodes (thus, can be used in asynchronous transfer mode (ATM) networks).

The partial virtual ring construction is based on symmetric block designs for which $k = r$ and $N = b$ (see Fig. 5). In a symmetric design every pair of blocks has exactly $\lambda$ elements in common. This is an important property for fast reconfiguration in case of faults even for $\lambda = 1$ since we
Fig. 5. Embeddings of virtual rings into the blocks of a \((13, 4, 1)\) design.

Based on symmetric designs, each block is used to construct one or more partial rings. We consider two embedding structures to obtain virtual rings within each block of a symmetric design: 1) a simple embedded ring (SER) which is a Hamiltonian cycle of the nodes that share the same block and 2) a tree (spanning tree of the nodes within a block) embedded ring (TER).

1) SER’s: A simple directed cycle \(C_j\) is constructed by connecting node \(b_i\) to node \(b_{i+1 \mod k}\) of block \(B_j\) with a directed edge. For example [see Fig. 5(b)], an SER can be embedded into the block \(B_0 = \{0, 3, 9, 1\}\) by establishing the directed edges \([0, 3], [3, 9], [9, 1], [1, 0]\). Since only one direction of each full duplex link is included into the \(C_j\), we can obtain a dual \(\overline{C_j}\) ring by taking the opposite direction of \(C_j\). Note that each node \(U\) has one or more virtual nodes on some of the virtual rings. Union of the edges of \(C_j\) and \(\overline{C_j}\) induces an undirected (bidirectional) ring \(R_j = C_j \cup \overline{C_j}\) into each block.

D. Design Tradeoffs for the Target Network

In this section we discuss the impact of parameters \(\lambda, k, r\) on the target topology with respect to fault tolerance, length of routing, and number of links.

1) Symmetric Versus Nonsymmetric Designs: The design techniques presented for the partial virtual ring embeddings are based on symmetric designs in which every pair of blocks have exactly \(\lambda\) elements in common. It is also possible to use a nonsymmetric design to construct the partial rings in which any two blocks have at most one common element. However, there is a tradeoff between using symmetric and nonsymmetric block designs for network construction. A symmetric design requires less number of links whereas a nonsymmetric one yields smaller block size and thus decreases the length of...
routing better. For example, a symmetric design with $b = 57$ blocks and $\lambda = 1$ would lead to a network of 57 nodes with $r = 8$ and block size $k = 8$. (That is a $(57, 57, 8, 8, 1)$ design.) In contrast a nonsymmetric design, with the same number of blocks (i.e., a $(29, 57, 9, 3, 1)$ design), would lead to a network of only 19 nodes with the parameters $r = 9$ and $k = 3$.

2) Tradeoff Between the Design Parameters: For a given $N$, the larger the value of the parameter $\lambda$ increases the fault-tolerant design of a network since there will be multiple blocks (virtual rings) between any pair of nodes. However, larger $\lambda$ yields to larger size blocks which in turn affects the length of routing (i.e., the number of hops). For example, consider a design with $N = 37$ nodes, then $\lambda = 1$ yields to $k = 4$ and $r = 12$. In contrast, for the same $N$, $\lambda = 2$ results $k = r = 9$. Therefore, there is a tradeoff between the degree of a node, and the routing length and fault tolerance. Furthermore, $\lambda = 1$ yields to fixed path routing with FIFO order (since a node on a partial ring receives all of the traffic via the same port), which is an important requirement for real-time traffic and ATM protocols. As a result, for the remainder of this paper we will limit our discussions to the block designs with $\lambda = 1$.

IV. BOUNDS ON THE LENGTH OF ROUTING

The routing algorithm is based on virtual rings, each of which operates (or emulates) as a buffer insertion or a slotted ring with spatial bandwidth reuse (e.g., see the MetaRing design [4]). It is assumed that each ring has links with the same capacity. In order to ensure no loss due to congestion, a packet has priority over all other packets for using the next ring link (this priority is the same as the ring traffic priority in buffer insertion rings). The routing over the network is done simply by forwarding the packets over a virtual ring that connects the source and destination. Each global ring induces a distinct linear ordering of the nodes in the network. Thus, it provides a global sense of direction for convergence of the routing. The default routing of a packet is to follow the links of the virtual ring that it enters to the network. However, based on the distance (which is computed on the rings) to the destination, the packet may get forwarded from one ring to another. Such a shortcut routing operation is possible depending upon the load conditions since our objective is to ensure congestion-free routing. Network control and fairness are regulated by the predefined asynchronous access quota given to each node in every global or local fairness cycle. See [8], [9], and [14] for the details of routing and network control.

The performance measure considered in this paper is the average number of hops assuming: 1) uniform destination distribution and 2) heavy load traffic conditions, which means that inside the network packets are sent over the full paths without shortcuts (i.e., worst-case analysis). Thus, in order to minimize the routing length (number of hops), we are looking for permutations of the nodes on the multiple virtual rings such that the average distance from one node to all other nodes is minimized.

Topological design techniques presented in this paper aim to minimize the expected length of the routing under heavy traffic conditions. In such a case, the probability of performing shortcut operation is small, thus the packets are likely to follow the path on a single ring by taking default operations. Thus, the size of each virtual ring and the location of the nodes on the rings determines the length of routing. In this section we first consider global rings and then the partial rings with SER and TER structures to drive analytical bounds on the length of routing assuming that only the default routing operations can be performed (worst-case).

A. Networks with Multiple Global Virtual Rings

Following we present a simple lower bound on the length of routing which is valid for any networks with multiple Hamiltonian embeddings.

Claim 1: \((N - 1)/d\) is a lower bound on the maximum length of routing on \(d\) global virtual rings.

Proof: Consider a generic node \(U\) and suppose that this node can access at each hop to \(d\) distinct nodes over \(d\) virtual rings (i.e., one node on each virtual ring). Then the maximum distance from \(U\) to any other node would be at most \((N - 1)/d\) hops. However, in the global ring construction, the set of the nodes that a generic node can access within \((N - 1)/d\) hops are not necessarily disjoint; thus, the maximum length of routing would be lower bounded by \((N - 1)/d\). □

B. Networks with Partial Virtual Rings

Assuming that the target topology is constructed by the symmetric block design with parameters \(N = n^2 + n + 1\), \(k = n + 1\), \(\lambda = 1\), in this section we compare the SER and the TER structures with respect to the length of routing.

First note that in a symmetric design with \(\lambda = 1\) there is exactly one partial ring (incomplete block) that includes a particular pair of the nodes. As a result, only the default routing operation via the next ring link is possible. Therefore, there is exactly one port to receive all of the packets originated from a source and FIFO discipline is ensured. As a corollary, we remark that a message entering to a ring designated to a physical node \(V\) never leaves this ring. Next we establish bounds on the maximum length of routing.

Claim 2: The maximum length of routing with multiple partial virtual rings is at most:

1) \(n/2\) (for SER);
2) \(2(n - \lceil \log (n + 1) \rceil)\) (for BTER).

Proof: 1) The maximum length of routing is bounded by the size of a virtual ring, which is simply the block size \(n + 1\). Since each virtual ring has a dual the maximum distance from a node to the others is \((n/2) < (\sqrt{N}/2)\).
2) On a virtual ring, a packet sent from \(U\) to \(V\) never leaves this ring and must traverse all of the virtual nodes, except the ones from sink \(V\) to the root and from the root to the source \(U\). Thus, we first compute the size of the virtual ring and then subtract the number of virtual nodes that are not visited to reach from \(U\) to \(V\). Assume without loss of generality a full binary tree of the nodes (for the largest size virtual ring) in a block. A simple counting shows that the virtual ring size is bounded by \(2(n + 1)\). Thus, the total number of virtual nodes visited is at most \(2(n + 1) - 2(\lceil \log (n + 1) \rceil - 1) = 2(n - \lceil \log (n + 1) \rceil) < 2\sqrt{N}.\) □
V. CONSTRUCTION OF NETWORKS OF ANY SIZE

In Sections III and IV we presented the basic methods for combinatorial construction of networks based on block designs; particularly, we considered symmetric designs and assumed that there is a symmetric design with \( N \) elements to construct a network of \( N \) nodes. However, it may not always be possible to find a block design with \( N \) elements to construct a network with \( N \) nodes. Furthermore, recall that even if the necessary conditions for existence of a block design is satisfied (see Theorem 1), actual construction cannot be achieved. In general, symmetric designs with parameters \( (n^2 + n + 1, n + 1, 1) \) can be constructed, based on projective geometries [1], [11], if \( n \) is a prime power, there are solutions. However, our objective is to design networks with a constrained number of links for engineering purposes. Since the degree of a node is a function of design parameter \( r \), the number of symmetric designs with small \( r = n + 1 \) are quite limited (e.g., see [6] for a list of available designs). For example, the set of integers for which there is a constructive symmetric design is \( \{7, 13, 21, 31, 57, 73, 91, 133, 273, 307, 381\} \).

Therefore, in this section we generalize the basic design methods and show how to construct networks of arbitrary size while keeping the degree of each node constant. We note that the scaling methods presented in this section are valid for both global and partial virtual ring construction; however, the techniques are presented on symmetric designs (i.e., for partial rings) for the sake of simplicity.

Given a positive integer \( N \) as the number of the nodes of a network to be constructed, our approach is based on decomposition of \( N \) by using the set of integers for which there exists a block design (i.e., using \( d_\ell \in \Delta \)). The scaling operations used for the decomposition are deletion (\( - \)), insertion (\( + \)), and multiplication (\( \ast \)) with the precedence relations from lowest to highest. For example, let’s discuss how to construct a network with 100 nodes. There are many ways to decompose 100 using the integers in \( \Delta \):

\[ 91 + 7 = 98 \] nodes covered, \( 100 - 98 = 2 \) remaining nodes are uncovered by this decomposition (we will show in the next section how to handle the uncovered nodes).

\[ 31 + 21 + 13 + 13 = 90, \] remaining nodes = 1.

The choice of the scaling operations constitutes a scaling formula (denoted by \( S \)) for the network and determines the target topology. Next we present operations for scaling and then examine the tradeoffs in detail.

A. Scaling by Insertion and Deletion

Insertion and deletion operations are defined between a block design and a positive integer \( M \). The insertion adds \( M \) nodes to a block design and deletion removes \( M \) nodes from it. Intuitively, insertion and deletion operations modify the incidence matrix of a block design. An insertion operation corresponds to: 1) addition of a new row to \( A \) and 2) copying the \( a_{i,j} \) values of the row \( i \) into this new row. The node that corresponds to row \( i \) of \( A \) is mapped to a new node to be inserted to the block design. Therefore, insertion of \( M \) nodes to a block design of \( N \) nodes corresponds to scaling the incidence matrix \( A_{N,N} \) up to \( C_{N+M,N} \).

Mappings: \( A \to 0, B \to 1, C \to 11, D \to 2, E \to 10, F \to 9, G \to 12 \)

Fig. 6. Scaling a 13-element design to 20-element by inserting seven elements.

The main motivation of insertion operation is to optimize on the maximum block size by inserting the new elements carefully. Thus the insertion operation is a mapping \( F \) which takes an incident matrix \( A_{N,N} \) and an integer \( M \) and constructs a scaled matrix \( C_{N+M,N} \). Construction is based on choosing \( r_1, \ldots, r_M > \) row indices of matrix \( A \) (to be mapped to the new elements one at a time using the scaling algorithm in Fig. 5) to minimize maximum block size. Formally,

\[
F : (A_{N,N} < r_1, \ldots, r_M >) \to C_{N+M,N} \text{ such that } \\
\min\{\max_i \{\sum_{j=1}^{M+1} c_{i,j} \} \forall j = 1, \ldots, M \}.
\]

Similarly, deletion of \( M \) nodes scales the matrix \( A_{N,N} \) down to \( C_{N-M,N} \) by choosing \( r_1, \ldots, r_M > \) rows and deleting them from the matrix \( A \). The main advantage of deletion is the smaller block size, thus shorter route length. For example, consider the \( (13, 4, 1) \) design to obtain a 19-node network by inserting six more nodes into its blocks; the maximum block size would be seven, whereas the maximum block size obtained by deletion of two nodes from a \( (21, 5, 1) \)
Fig. 7. Insertion with minimum increase in block size.

design is five. The main advantage of insert operation over delete operation is that any two blocks after insertion may have at least one common element which may become an important property for fault tolerance. In contrast, after deletion, any pair of blocks would have at most one common element.

B. Scaling by Multiplication

These scaling operations are defined between a block design of $N$ nodes and a positive integer $M$. The result of multiplication is to scale up to $MN$ nodes. In the following definition, $\%$ is used to represent modulo operation:

$$ F : (A_{N,N}, M) \rightarrow C_{MN,N} $$

$$ a_{k,i,j} = a_{k,j} \text{ if } k\%N = 1 \text{ for } k = 1, \ldots, MN; i, j = 1, \ldots, N. $$

(5)

Note that the multiplication operation keeps the number of blocks the same as before and expands the size of each block as a function of $M$. As we shall discuss below, such scaling will maintain the degree constraint in the expense of routing length.

C. Choice of the Scaling Formula and the Base Design

We note that there is a tradeoff between using many smaller size block designs and a few larger size block designs to construct the target network. For example, if we construct a 100-node network by the scaling formula $S1 \{31, 6, 1\} + 2(21, 5, 1) + 2(13, 4, 1) + 1$ then the maximum block size will be at most 24, which is determined by the insertion of modified blocks of size ten [which is obtained $2(21, 5, 1)$] and of size eight to the block design $31, 6, 1$. Instead, if the scaling is done according to the formula $S2 \{91, 10, 1\} + 7(3, 1) + 2$ then the maximum block size will be at most $10+3+1 = 14$. On the other hand, the scaling formula $S2$ implies that each node occurs in ten virtual rings (the parameter $r$), thus the degree of each node is 20. However, with the scaling formula $S1$ the degree of each node would be 12 because the base design has $r = 6$.

1) Choice of the Base Design and Optimality: What determines the impact of a formula on the target network is the choice of the base design. Once the base design is decided, scaling operations are performed on this design to obtain the target network size. Thus, for a given degree bound, a base design can be chosen from $\Delta$ as the block design which has the largest $r$ parameter smaller than the degree bound.

In what follows, we show a scaling algorithm which designs a network with $N$ nodes while ensuring that the maximum degree of a node remains constant. The algorithm performs the scaling operations in such a way that maximum block size is minimized. Thus, the scaling is optimal since it results in minimum increase at the length of routing while ensuring constant degree for each node.

D. The Scaling Algorithm

The scaling algorithm is based on following the precedence of scaling operations on the base design. Thus, the first step in the algorithm is to scale the base design with power and multiplication operations (if indicated by the scaling formula). Let $k$ be the block size of the base design after such operations. Next, the rest of the nodes are appended—one at a time—to the blocks of the modified base design to scale the number of nodes to $N$.

The mapping between the rows for insertion operation determines the maximum block size and thus the length of routing. It must be ensured that scaling distributes the nodes of the smaller design into the blocks of the base design as evenly as possible. In Fig. 7, we present an algorithm for scaling which minimizes the maximum block size.

Let $A$ be the set of the nodes to be included to the design (after the base design is modified) and let $Base$ be the set of the nodes of the base design (nonmodified). For example, in the scaling formula $S1$ the $Base$ has 31 nodes and the set $A$ has cardinality $\|A\| = (2 \times 21) + (2 \times 13) + 1$. Let Used $\subseteq Base$ be the set of nodes of the base design that are associated with an element in $A$ during the execution of the algorithm. Denote by $R_{min}, R_{max}$ the set of minimum and maximum size blocks constructed during the scaling, such that the size of the blocks.
Algorithm Reconfiguration-by-Block-Merge
1. Choose a node $U \notin B_{xy}$ and assign $U$ as the leader of the merged blocks.
2. $U$ determines in constant time the blocks $B_{ux}$ and $B_{uy}$ such that $U, X \in B_{ux}$ and $U, Y \in B_{uy}$
3. $B_{ux} = B_{ux} \cap B_{xy}$ to insert the component under $X$ in the disconnected block $B_{xy}$ to the topology embedded into block $B_{ux}$
4. $B_{uy} = B_{uy} \cap B_{xy}$ to insert the component under $Y$ in the disconnected block $B_{xy}$ to the topology embedded into block $B_{uy}$
5. Set $B_{uxy} = B_{ux} \cap B_{uy}$ to merge the corresponding subgraphs under the leader node $U$.

in these sets is $\min$ and $\max$, respectively. The set $L$ in the algorithm has the candidate nodes, which are the nodes of the base design to be associated with a new element.

Claim 3: Given a scaling formula $S$, algorithm scaling $(S, N)$ scales the base design with minimum increase in its block size to obtain an $N$-node network.

Proof: Termination is guaranteed by line 3 (please refer to Fig. 7) which causes a monotone decrease in the set $A$. For the correctness first we must show that an element in $A$ can always be associated with an element in the base design. Note that this is ensured since $L \neq \emptyset$ as long as $A \neq \emptyset$ (in line 5.2) where $L$ is the set of candidate nodes of the base design. Minimum size blocks are constructed by maintaining a minimum difference between the $\min$ and $\max$ as an invariant to ensure that new nodes to the blocks are distributed as evenly as possible. This invariant is maintained by always trying first to pick an element from the blocks with cardinality $\min$ (see line 6). If no block of size $\min$ is found, then the variable $\min$ is incremented (line 9). To ensure minimum difference between the variables $\min$ and $\max$, the latter is incremented if all the elements of the base design is in the Used set.

In general, let $K$ be the sum of the block size of all of the designs included into a given scaling formula, then the maximum block size is at most $K$. Furthermore, knowing $K$ we can compute an upper bound on the maximum length of routing according to Claim 2 by substituting $n + 1 = K$. Consequently, the following claim establishes an upper bound on the length of routing.

Claim 4: If the base design has size $\sqrt{N}$, then a scaling of $N$ to $fN$ results a network with $fN$ nodes, each with degree $2\sqrt{N}$, and the length of routing is at most $f\sqrt{N}$.

Proof: Note that $\sum_{i=1}^{f} k_i = K \leq f\sqrt{N}$ [i.e., the sum of the block size of all the designs $(n_i, k_i, 1)$] indicated by $S$. The degree of each node is determined by twice the number of occurrences of that node in the blocks of the base design which is $\sqrt{N}$.

E. Scaling Examples

Next we present some examples to show how to use the scaling methods to construct networks of maximum size for a given degree $d$ while ensuring the maximum length of routing to be bounded by $d^2$. Consider Table II and suppose that the maximum degree of a node is given as $d = 64$. The maximum size network for which the length of routing is at most $d^2 = 4096$ can be computed as follows. First, the base design $(381, 20, 1)$ is scaled to $(381, 20, 1) \times 204 = 77724$ nodes with maximum block size $20 \times 204 = 4080$. In order to ensure the bound on the routing we could add $(13, 4, 1) \times 4$ which increases the block size to 4096 and number of nodes to $N = 77776$. However, instead if we add $(57, 8, 1) \times 2$—to the scaled based design—then the maximum number of nodes would be 77838.
Furthermore, note that the degree of a node induced by addition of \((13, 4, 1)\) is only \(2(20+4) = 48\), which is less than the degree constraint for a node (i.e., 64). Thus we can afford using additional block designs to establish subrings (clusters) of size 13 in order to decrease the average number of hops. In general, besides the final (scaled) block design, some of the intermediate block designs can be used in the construction of the network in order to minimize the maximum length of routing among some subset of the nodes in the target network. We shall address clustering techniques in our future work; however, we note that the tradeoff is between the length of routing and the total number of links. In order to show the asymptotic behavior of the degree/node ratio, we plotted the values from Table II in Fig. 8. Note also that for a given \(d\),

![Fig. 10. Reconfiguration for the link failures in TER \((X = 1, Y = 7, \text{ and } U = 8)\).](image)

**TABLE II**

<table>
<thead>
<tr>
<th>max. degree (d)</th>
<th>max. # of hops = (d^2)</th>
<th>max. # of nodes = (N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>64</td>
<td>((13, 4, 1) \times 16 \rightarrow N = 208)</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>((21, 5, 1) \times 20 \rightarrow N = 420)</td>
</tr>
<tr>
<td>12</td>
<td>144</td>
<td>((31, 6, 1) \times 24 \rightarrow N = 744)</td>
</tr>
<tr>
<td>16</td>
<td>256</td>
<td>((57, 8, 1) \times 32 \rightarrow N = 1824)</td>
</tr>
<tr>
<td>20</td>
<td>400</td>
<td>((91, 10, 1) \times 40 \rightarrow N = 3640)</td>
</tr>
<tr>
<td>32</td>
<td>1024</td>
<td>((133, 12, 1) \times 85 + (13, 4, 1) \rightarrow N = 11318)</td>
</tr>
<tr>
<td>40</td>
<td>1600</td>
<td>((381, 20, 1) \times 80 \rightarrow N = 30480)</td>
</tr>
<tr>
<td>64</td>
<td>4096</td>
<td>((381, 20, 1) \times 204 + (57, 8, 1) \times 2 \rightarrow N = 77838)</td>
</tr>
</tbody>
</table>
choosing the largest $r$ results also larger values of $N$. For example, for $d = 64$ a symmetric design with $r = 31$ can be used for larger $N$ (i.e., $(93,32,1) \times 128 \rightarrow N = 127104$).

VI. FAULT-TOLERANCE ISSUES

In this section we discuss the fault-tolerant properties of the networks constructed with the combinatorial design methods suggested in this paper. We consider how to maintain global host-to-host communication (i.e., all the nonfaulty nodes continue sending and receiving packets). Particularly, we consider a single link failure and compare the embedding structures: SER and TER.

A. Maintenance of Global Host-to-Host Communication

Maintenance of continuous global end-to-end communication is ensured by the existence of a virtual ring between any pair of nodes. Thus, in case of a link failure, a fast algorithm for reconfiguration of the virtual rings is needed. In this section we present a reconfiguration algorithm based on merging the blocks of the underlying design. We also note that efficiency of such an algorithm is limited by the underlying embedding structure. In particular, we show that—after a link failure—maintenance of the Hamiltonian property may not be possible. Thus, using trees (TER’s) are more robust than using simple virtual rings (SER’s) for construction and maintenance of fault-tolerance networks.

1) Reconfiguration Algorithm: Consider Fig. 10(a) in which a block with eight elements is used for a TER [this block belongs to the BIBD with parameters $(57,8,1)$]. Suppose the link between nodes $X = 1$ and $Y = 7$ becomes faulty in block $B_{xy}$. Assuming $\lambda = 1$, in Fig. 9 we present an algorithm for reconfiguration which is run by the (lexicographically) smaller end point of the faulty link. For simplicity the algorithm is presented for TER; however, it is also valid for SER since operations in the algorithm are defined on blocks.

Consider the example in Fig. 10(b) in which node 8 is chosen as the leader and the blocks $B_{1}$ and $B_{16}$ are merged under the leader node. The connected components in the block $B_{1}$ are included to the merged block via the end nodes of the faulty link (1,7). After the execution of the reconfiguration algorithm, a new spanning tree of the configured subgraph is computed and a new TER is constructed for assigning new virtual addresses. However, the topology induced by the reconfiguration algorithm may not yield an SER embedding as explained below.

2) Maintenance of SER Structure: The main advantage of the SER structure is that global end-to-end communication is destroyed two link failures for $\lambda = 1$. This is true since a faulty link disconnects a virtual ring and its dual. However, the union of the edges of these rings together induce a connected subgraph which includes all of the nodes of the underlying block. Reconfiguration involves using two nonfaulty blocks. Consider the undirected ring embedded into block $B_{0}$ in Fig. 10(a) and assume that the links $(1,49)$ and $(7,9)$ become faulty. The reconfiguration algorithm in Fig. 9 can be used by choosing eight as the new root and merging the blocks $B_{1}$ and $B_{2}$. The subgraph obtained after reconfiguration is not necessarily Hamiltonian. Thus, construction of an SER on this subgraph may not be possible as shown in Fig. 10(b). Hence, in order to ensure continuous host-to-host communication, a
TER on a spanning tree of the reconfigured subgraph needed to be constructed.

3) **TER Structure with Multiple Link Failures:** Although a single link failure is enough to destroy global host-to-host communication, the main advantage of TER structure is its high reconfiguration capability. For example, suppose another link failure (say between $A$ and $C$) occurs on the reconfigured topology. We consider two cases: 1) the faulty edge belongs to the block which had a fault before and 2) the faulty link destroys a new block. In the first case we note that the blocks $B_{ux}$ and $B_{uy}$ cannot be used to reconfigure the network after the failure of link $(A, C)$, since otherwise $\lambda = 1$ is violated. Thus there must be two blocks $B_{uy}$ and $B_{uc}$ such that $U, A \in B_{ux}$ and $U, C \in B_{uc}$ (due to the fact that there is a block between each pair of nodes). As a result we can merge these two blocks by the above algorithm and keep $U$ as the leader (root). In the second case, the new failure occurs in one of the nonfaulty blocks used for reconfiguration and $U$ cannot remain as the root (since $\lambda = 1$). A new root will be chosen to merge the disconnected topology.

**VII. SUMMARY**

In this paper we presented a new design technique for loss-free packet switched networks. Our design has focused on degree constraint at each node, and is based on combinatorial design of multiple virtual rings. The target network was synthesized from an edge union of these rings. It was shown how the design techniques can be used to build networks of any size with a tradeoff between the route length and the degree of each node (thus the number of links in the network). It was shown analytically that this design yields to an optimum bound on the worst-case route length.

We also discussed the fault-tolerant property of networks built with our design tool, and concluded that this design tool provides complete configurability under link failures.

**REFERENCES**


**Bülent Yener** (’S94–’A95–’M97) received the B.S. and the M.S. degrees in industrial engineering from the Technical University of Istanbul, Istanbul, Turkey, in 1982 and 1984, respectively. He received the M.S. and the Ph.D. degrees in computer science from Columbia University, New York, NY, in 1987 and 1994, respectively.

He is currently an Assistant Professor and the Director of the Networking and Communication Laboratory, Department of Computer and Information Sciences, New Jersey Institute of Technology, Newark. He is also a Visiting Scientist in the Computer Science Department, Columbia University. His current research interests are in protocol design and network optimization with a focus on group multicasting and routing in mobile networks.

Dr. Yener is a member of the Association for Computing Machinery (ACM).

**Yoram Ofek** (’S86–’M87) received the B.Sc. degree in electrical engineering from the Technion—Israel Institute of Technology, Haifa, in 1979, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Illinois, Urbana, in 1985 and 1987, respectively.

From 1979 to 1982 he was a Research Engineer with RAFAEL, Haifa. From 1983 to 1984 he was with Fermi National Accelerator Laboratory, Batavia, IL, and from 1984 to 1986 he was with Gould Electronics, Urbana, IL. Since 1987, he has been with the IBM T. J. Watson Research Center, Yorktown Heights, NY, as a Research Staff Member. He has conceived, initiated, and led the research activities into four novel network architectures: 1) optical hypergraph for combining multiple passive optical stars with flow control and global synchronization; 2) ring networks with spatial bandwidth reuse and fairness, MetaRing, in which key components are used in the SSA standard (ANSI X3T10) and several IBM products; 3) embedding of virtual rings in arbitrary topology network, MetaNet, for bursty data traffic with no packet loss, fairness, and reliable/real-time broadcast/multicast; and 4) global networks for real-time traffic with GPS-based synchronization for providing deterministic QoS guarantees—as in circuit switching.

Dr. Ofek was the Program Co-Chairperson of the 6th and Chair of the 7th IEEE Workshop on Local and Metropolitan Area Networks. He has served on various program committees and was a guest editor in several journals.

**Moti Yung** received the Ph.D. in computer science from Columbia University, New York, NY. Currently, he is a Vice President with CertCo (formerly Bankers Trust Electronic Commerce), New York, dealing with electronic commerce applications and infrastructure. He was with IBM Research from 1988 until 1996. He has been working on a wide range of cryptography and network security issues arising from foundations and principles to analysis and design of components, primitives, and protocols. He was a key contributor to IBM products and offerings (authentication technologies, network security products, server security, technology for Internet security and IBM’s SecureWay). He has also worked on networking protocols and architectures, and distributed algorithms.