JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 26, 1–23 (1995)

# "Hypermeshes": Optical Interconnection Networks for Parallel Computing[1]

TED SZYMANSKI[2]

*Department of Electrical Engineering, McGill University, Montreal, Quebec, Canada H3A 2A7*

A formal model for a class of optical mesh-based interconnection networks called "hypermeshes" is proposed and characterized. Hypermeshes are based on the concept of orthogonal crossbar switches, with $N$ nodes arranged in $n$-dimensional mesh structure where all nodes aligned along a dimension are interconnected with an optical multichannel switch. The optical multichannel switches can be modeled as hypergraph "hyperedges" which can perform multiple data transfers over their members simultaneously. The hyperedges can be implemented with space division multiplexing (SDM) in the electrical or optical domains or with wavelength division multiplexing (WDM) over a single fiber in the optical domain. The use of WDM over a fiber also reduces the hypermesh "interconnection complexity" to rival that of a 2D mesh. An architectural characterization is performed and the combinatorial properties, including rearrangeability, permutation capability, partitionability, embedding capability, and bisection bandwidth, are characterized. It is shown that every algorithm which can execute conflict-free on an omega network can execute conflict-free on a hypermesh and that every graph which can be embedded into a hypercube with dilation $k$ can be embedded into a hypermesh with dilation $\leq k$. Hypermeshes are shown to have high bisection bandwidths, thereby minimizing the time for many common algorithms such as parallel sorting. It is shown that under the constraint of equivalent aggregate bandwidth the hypermeshes are considerably more powerful computational models than meshes, generalized hypercubes, and other orthogonal graphs. Two attractive optical implementations of hypermeshes using optical technology recently advocated in the literature are also proposed. 

## 1. INTRODUCTION

The bandwidth requirements in the next generation of multiprocessors will be an order of magnitude higher than in existing systems, leaving optics as perhaps the only feasible interconnection technology. Fiber optic technology is becoming attractive due to the continually decreasing cost, the very high aggregate bandwidth of a fiber (in the multi-Thz range), and the potential for large scale "wavelength division multiplexing" (WDM), i.e., the ability for a single fiber to support multiple optical channels in the GHz. bandwidth range, each on a separate wavelength.

In this paper, a formal model for a class of interconnection networks called "hypermeshes" is proposed and characterized. The contributions include: (1) a definition of the formal model based on the concept of orthogonal hypergraphs; (2) the detailed architectural characterization of the model's permutation capability, rearrangeability, partitionability, graph embedding capability, and bisection bandwidth; (3) the derivation of asymptotic metrics such as its crosspoint cost and transmission line cost; (4) a comparative evaluation with other architectural models (meshes and hypercubes) implemented in the same technology; and (5) the proposal of two efficient optical implementations. The hypermesh is a network architecture which has both optical and electrical implementations, and electrical implementations will be described as well. Electrical implementations can facilitate the transition between technologies.

Traditional interconnection networks for parallel processors can be grouped roughly into three families: (1) those based on *multistage networks* such as the SW-banyan and Omega networks (or "butterfly" and "perfect-shuffle" networks as they are often called); (2) those based on *point-to-point graphs*, including meshes, toroids, and hypercubes [9, 16]; and (3) those based on *buses*.

Point-to-point networks can be formally modeled as undirected graphs $G(E,V)$ defined over a set of homogeneous vertices $V$ and a set of edges $E$. Each vertex typically represents one or more processing elements (PEs) and each undirected edge between two vertices represents a bidirectional communication channel between the two vertices. A fundamental constraint of the graph model is the definition that each edge joins precisely two vertices [4]. Many interconnection networks can be viewed as "orthogonal graphs," and Scherson has presented a formal framework for classifying such networks [17]. An orthogonal graph is characterized by its radix $d$ and its dimensionality $n$. All $d$ nodes whose positions differ in exactly one dimension are said to be "*aligned*" along that dimension. The fundamental difference between many well known networks is the *precise manner* in which the nodes aligned along a dimension are interconnected.

[1] This research was supported by NSERC Canada Grant OGP0121601.

[2] E-mail address: teds@macs.ee.mcgill.ca.



0743-7315/95 $6.00

In a $d^n$-mesh, all $d$ nodes aligned along a dimension are interconnected with a ring [6]. A $d^n$-mesh with radix $d = 2$ and with $\log_2 N$ dimensions corresponds to the well known binary hypercube. Binary hypercubes can also be generalized to have higher radixes: In the graph model of a radix-$d$ *generalized hypercube*, all the vertices aligned along a dimension are interconnected with a clique (or fully connected graph) [1]. All of these orthogonal graphs can be "unfolded" to yield multistage graphs with similar properties [2, 7, 11, 31].

Bus-based networks such as the "spanning bus hypercube" have been discussed by Wittie [15]. In the spanning bus hypercube with $d^n$ nodes, all $d$ nodes aligned along a dimension are interconnected with a bus. By conventional definition a bus has a *single channel shared* over all its members, and hence it can transfer a single packet between two nodes on the same bus in *one logical step* [15]. Due to the sequential nature of operation, buses have relatively poor performance at higher traffic loads [15]. This poor performance can be alleviated in a number of ways; one approach is to use a ring to interconnect the nodes aligned along a dimension, which results in a conventional $d^n$ mesh. The performance of a 2D mesh can be further improved through the addition of a small number of buses, as proposed by Stout [21] and Kumar and Raghavendra [20]. However, the addition of a small number of busses does not significantly affect the architecture or bisection bandwidth of the underlying 2D mesh, and as a result these 2D mesh-based architectures cannot perform many algorithms such as sorting very effectively.

Many optical networks rely on distributed optical crossbar-like switches and they cannot be properly modeled as conventional point-to-point graphs [35]. In this paper, a class of multidimensional optical networks called hypermeshes is proposed in a formal graph-theoretic framework based on hypergraphs. Hypergraphs are generalizations of the conventional graph in which edges are generalized to yield "hyperedges," and where a hyperedge represents a logical relationship among an arbitrary number of vertices rather than just two vertices [4]. (The phrase "hyperedge" is used rather than "edge" to make explicit the generalization.)

Graph-theoretic models are used in practice because once a network can be formally modeled as a graph or hypergraph, all the algorithms and theorems from the field of graph-theory can be used to characterize the model. Implementations of the same formal model would share the same architectural characteristics. The need for formal models for optical networks is apparent from the literature; to date, no modeling formalism which captures all the features of optical technology has been proposed (see [35]).

The proposed hypermesh can be viewed as an multidimensional orthogonal network, with $N = d^n$ nodes arranged in $n$-dimensional Euclidean space, where all $d$ nodes aligned along each dimension belong to a hyperedge. Each hyperedge with $d$ members represents access to a distributed multichannel switch which can perform $O(d)$ data transfers over its members in *one logical step*. For the remainder of this paper we assume that each multichannel switch can perform a permutation of data over its members simultaneously, and we model each switch as a single hypergraph edge.

The definition of a hypermesh is "functional" and a fundamental attribute of all $d^n$ hypermeshes is that each hyperedge supports $O(d)$ independent channels which allow up to $O(d)$ *simultaneous* data transfers. There are many possible ways to implement the functionality: each hyperedge could utilize space division multiplexing by supporting $O(d)$ spatially distinct electrical or optical channels to perform the simultaneous mappings. Alternatively, each hyperedge could utilize wavelength division multiplexing over a single optical fiber to achieve the required functionality.

Distributed WDM optical crossbar-like switches form the basic building blocks (the hyperedges) of large multidimensional WDM optical hypermeshes. The maximal size of a WDM optical crossbar is limited to perhaps 16 or 32 in the foreseeable future, due to constraints on wavelength tunability [26, 27] and on maximal fiber length. Hence, to interconnect multiple PEs into a massively parallel machine, one must use multiple smaller WDM optical switches arranged into some type of efficient network architecture. In large networks composed of multiple smaller switches, even the routing of data between nodes can become quite complex, and the study of traffic flows in such networks involves graph theory, queueing theory, and architecture. The hypermesh is large multidimensional crossbar architecture, and it is the capabilities of this architecture that are formalized in this paper.

Hypermeshes are more expensive and considerably more powerful than spanning bus hypercubes [15], since the buses, which can transfer one packet at a time, have been replaced by $O(d)$ buses, which can transfer $O(d)$ packets in the same amount of time. When one interconnects thousands of the building blocks into higher dimensional networks, the costs and capabilities are considerably different. As a result of these changes, Wittie's architectural characterizations of spanning bus hypercubes [15] do not apply to hypermeshes. Hence, the hypermesh architectural model has different costs and architectural capabilities from a spanning bus hypercube or generalized hypercube, and these costs and capabilities are precisely characterized in this paper.

Previously, one- or two-dimensional electrical crossbar-like switching networks have been proposed by Szymanski [12, 14], Tanabe *et al.* [18], Wang *et al.* [25], and Mackenzie *et al.* [32]. However, large multidimensional crossbars are not easily manufacturable with conventional electrical technology, and it is the emerging optical technologies which makes multidimensional optical

crossbar networks attractive. A one-dimensional optical crossbar-like network was proposed by Wailes *et al.* [24]. Higher dimensional optical crossbar-like networks have been proposed by Szymanski, Dowd, and Li *et al.* in [5, 12, 14, 22, 36], and routing in these networks was considered in [13]. In this paper, it is shown that all of these networks belong to a single architectural class with common architectural attributes, which include high bisection bandwidth, high permutation capability, and high embedding capability. It is shown that optical versions of these networks do not fit into any of the existing architectural characterizations based on graph models, and that the optical networks cannot even be properly formally modelled as graphs. The unique capabilities of the multidimensional networks warrants the definition of a distinct class of network which accurately reflects the architecture. In this paper, we will refer this class of network as a "hypermesh," which reflects both the multidimensional orthogonal mesh-like structure and the hypergraph-based formal model.

A "bisector" of a graph or hypergraph can be informally defined as any line or surface which partitions the graph or hypergraph into two disconnected components of equal size. The bisection bandwidth of a network (graph or hypergraph) can be defined as the minimum bandwidth crossing a network bisector, taken over all possible bisectors. As a result of their definition, hypermeshes have bisection bandwidths which are higher than those of conventional 2D or 3D meshes and toroids, and comparable to that of binary hypercubes. Many parallel algorithms such as sorting make extensive use of data transfers over network bisectors, and the speed of this operation is bounded by the bisection bandwidth. It will be shown that under the constraint of *equivalent aggregate bandwidth*, hypermeshes can perform parallel sorting faster than conventional meshes, based on the higher bisection bandwidth.

The hypercube is considered to be one of the most powerful formal models for parallel computation [29], and hence it is interesting to compare a hypermesh with a hypercube (both in the same domain, either electrical or optical). The degree-log $N$ hypermesh uses fewer crossbar switches and fewer directed links by a factor of $O(\log \log N)$, where $N$ is the network size [13]. The reduction in directed links represents significant decrease in the complexity of the backplane interconnect. It is also shown that the reduction in crossbar and link cost does not result in any significant loss of computational power when compared to a hypercube. This result is interesting given that the hypercube is amongst the most powerful computational models known.

In the electrical domain a 2D or 3D hypermesh is more complex to interconnect than the 2D or 3D mesh, which is the price paid for the higher bisection bandwidth. However, in the optical domain the use of wavelength division multiplexing over a single fiber can reduce the hypermesh interconnection complexity significantly, so that it rivals that of a conventional 2D or 3D mesh. Thus, while hypermeshes are more complex to interconnect than 2D or 3D meshes in the *electrical domain*, hypermeshes have interconnection complexity comparable to 2D and 3D meshes in the *optical domain*.

Two novel and attractive implementations of hypermeshes are proposed. The first "fiber-optic" implementation relies on a chip-set for fiber-optic networks proposed in [28], which was funded by Darpa's Optical Computing Program. It should be noted that this chip-set can be used to construct any network, including 2D or 3D meshes, hypercubes and the proposed hypermeshes. It is shown that the hypermesh architecture is more powerful than the others, and that the improvement is due to the architecture and not due to the chip-set or technology. The second proposed implementation involves the use of newer optical technology which is not commercially available yet, namely rapidly tunable laser diodes and optical receivers, and the use of WDM over a single fiber of minimum physical distance.

This paper is organized as follows. Section 2 reviews graph models of networks and orthogonal graphs. Section 3 defines a formal model of hypermeshes based on orthogonal hypergraphs and describes efficient implementations. Section 4 considers the permutation capability, rearrangeability, partitionability, and graph embedding properties of hypermeshes. Section 5 performs a quantitative comparative evaluation between meshes, hypermeshes, and hypercubes based on a queueing model by Kleinrock [30]. Section 6 performs a comparison of sorting on meshes, hypermeshes, and hypercubes. Section 7 contains some concluding remarks.

## 2. A REVIEW: GRAPH MODELS OF NETWORKS AND ORTHOGONAL GRAPHS

### 2.1. The Definition of a "Time Step"

Throughout this paper, we assume that every undirected edge $e$ joining two vertices $(u, v)$ represents two data channels (or transmission lines), one from $u$ to $v$ and one from $v$ to $u$, each with a bandwidth of $B$ bit/s.

In order to properly model real physical distances, a modeling convention used by Kleinrock in [30] is adapted. The basic unit of time in a discrete-time network is the "*packet cycle time*" denoted $T_R \cdot T_R$ is defined as the time required to transfer a fixed size packet containing $P$ bits from one node to a neighbor over the longest edge (i.e., transmission line), with length $L$ meters and with bandwidth $B$ bit/s.

$$T_R \equiv T_s + T_p + T_n.$$

The first term $T_s$ is often called "transmission delay" in the queueing literature [30]. It represents the time required for all the bits in the packet to depart from the sender over the channel, and hence $T_s = P/B$. The second term $T_p$ represents the time-of-flight "propagation delay" over the channel, i.e., the time required for the last bit in the packet to propagate through the medium and reach the destination. $T_n$ represents the "node latency," which includes the deterministic logic delays incurred when a packet passes through a node. See Kleinrock's text for elaboration of these quantities [30].

If a packet can be delayed in a queue(s), then the time a packet spends waiting in queues is called the "queueing delay" or "queueing time" [30]. In a discrete time network the queueing delay is also expressed in terms of the packet cycle time, and it can be determined with a discrete-time queueing analysis [30].

There exist many different models for the propagation delay of a transmission line of length $L$, including a linear delay $O(L)$, logarithmic delay $O(\log L)$, or unit delay $O(1)$. In optical systems optical bit-streams travel down a fiber or waveguide at roughly $(2/3)c$, where $c$ is the speed of light. Hence, a linear delay model is warranted for optical systems. The linear delay model also applies for high-bandwidth electrical transmissions along an electrical transmission line.

EXAMPLE 1: Consider a general purpose multiprocessor compactly arranged within a 3D volume of length 4 m (or roughly 12 ft.) on each side, yielding a maximum volume of 64 $m^3$, large by current standards. Assume a packet size of 128 bits; the latest generation of microprocessors have 64 bits of address space and read or write 64 bit of data at a time, hence a 128-bit packet seems to be the minimal requirement for interconnecting such general purpose micro-processors. The velocity of light in a fiber is roughly $2 \times 10^8$ m/s. At a "state-of-the-art" transmission bandwidth of 1 Gbit/s. [26–28], each bit in an optical transmission occupies 0.2 m of fiber and it propagates forward at a speed of 0.2 m/ns. Assuming transmissions travel at most 4 m (the length of an orthogonal axis), then the propagation delay is at most 20 ns, and the transmission delay is 128 ns. Thus, in this realistic scenario the propagation delay is a small fraction (15%) of the transmission delay.

Also, the (inherent) node latency can be small in a properly designed system; in the well known "fiber-distributed data interface" (FDDI) ring network standard, the inherent latency within each node is 10 bit times, just enough to observe the relevant token header bits and make a routing decision. In a similar high-speed slotted ring, the node latency is typically a few bit times, just enough to examine and overwrite a full-empty bit in the slot header. Thus, in a typical FDDI fiber-optic system the node latency is a very small fraction of the packet transmission time, typically less than 10% of the transmission delay.

The technique of splitting the packet delay into three identifiable components (the transmission delay, propagation delay and node latency) was proposed by Kleinrock in his well known text as a basis for comparing various networks [30], and will be used in this paper. Example 1 has illustrated that the propagation delay and node latency will be a small fraction of the packet cycle time in systems using existing fiber-optical technology. In any case, the model makes all these delay components explicit and each can be assigned arbitrary values in all the analyses to follow.

### 2.2. *The Definition of a "Neighbor"*

In the graph model of a network, two vertices $u$ and $v$ are defined to be "neighbors" if they are end-points of the same edge (i.e., $e = (u, v)$) [4]. As a consequence of the definition, it follows that (1) a transmission from vertex $u$ to its neighbor $v$ can occur without conflict (since $u$ has a dedicated channel to $v$), and (2) the time required to transmit a packet is upper bounded by parameter $T_R$ (which reflects the length and bandwidth of the longest transmission line in the graph model).

According to conventional graph-theoretic definitions, two vertices of a hypergraph are defined to be "neighbors" if they belong to the same hyperedge [3, 4]. As a consequence of this definition, it follows that (1) a transmission from vertex $u$ to a neighbor $v$ can occur without conflict if no other neighbor is simultaneously transmitting to $v$ over the same hyperedge, and (2) the time required to transmit a packet is upper bounded by parameter $T_R$ (which reflects the length and bandwidth of the longest transmission line in the hypergraph model).

EXAMPLE 2: Consider a bus spanning $d$ nodes arranged in a (one-dimensional) linear array. Since access to the bus is sequential, the bus is better represented as a hypergraph hyperedge, where membership to a hyperedge implies access to a (shared) bus. All $d$ nodes are "neighbors" by the above definition, since any one node can transmit a packet to any other node in one basic unit of time (the packet cycle time), although they cannot transmit simultaneously.

EXAMPLE 3: Consider a collection of $d$ processors interconnected with a distributed $d \times d$ crossbar switch (see Fig. 3a and Fig. 4 below). There are some difficulties with modeling this system as a conventional graph: (a) Should it be modeled as a graph with $d$ vertices of degree one (representing the processors) and one special vertex of degree $d$ (representing the crossbar switch), with $d$ directed edges from the processors to the switch (and visa versa), for a total of $2d$ directed edges and a diameter of 2? Or should it be modeled as a fully-connected graph (i.e., clique) with $d$ vertices of degree $d$ each, with one edge from every vertex to every other vertex, for a total of $d(d-1)/2$ undirected edges, and a diameter of 1? The problem with the first approach is that it suggests a diam-

eter of 2 when common convention accepts that all these $d$ vertices are neighbors with a diameter of 1. It also requires the creation of a special vertex representing the crossbar switching function, when the optical networks in Figs. 3a, 4c, and 4d have no such localized crossbar. The problem with the second approach is that it implies (1) the existence of $d(d-1)/2$ edges, which the system does not have, and (2) it implies that the bisection bandwidth is considerably larger than it really is. One solution is to model this system as a hypergraph, with these $d$ nodes belonging to a hyperedge, where in this case membership to the hyperedge implies access to a distributed crossbar-like switch. These $d$ nodes are therefore neighbors with diameter 1, as convention dictates, and there is no need for the creation of an artificial vertex for the crossbar functionality.

A multidimensional WDM fiber-optic network based on the structure in Fig. 4c was described in [12, 22]. In each paper it was stated that the $d$ nodes in Figure 4c joined through the star-coupler are nearest neighbors. Note that these $d$ nodes cannot be neighbors in a graph model, unless they form a clique with $d(d-1)/2$ edges, which would imply an incorrect number of edges/channels and an incorrect bisection bandwidth. The hypergraph is the only formal model which allows the nodes in Fig. 4c to be neighbors in the graph-theoretic sense, hence capturing the essence of the optical technology. The same applies for the WDM implementation in Fig. 4d.

### 2.3. Graph Models of Meshes

A $d^n$ mesh (where $N = d^n$) consists of $N$ nodes arranged in $n$-dimensional space, where all $d$ nodes aligned along a dimension are interconnected with a linear array. A 2D mesh is shown in Fig. 1a. For a given size $N$, as the dimensionality $n$ increases the radix $d$ decreases. At the extreme, one obtains a mesh with dimension $\log N$ and radix 2, which corresponds to the well-known binary hypercube (when its redundant wrap-around links are removed). The binary hypercube with $N$ nodes can be modeled as a point-to-point graph where

$$V = \{v | v \in 0, 1, \ldots, 2^n - 1\}$$

$$E = \left\{ \begin{array}{ll} (u,(u + d^j) \bmod N) & \cup \\ (u,(u - d^j) \bmod N) & \\ \forall u \in V, \quad \forall 0 \le j < n & \end{array} \right\}.$$

In the context of Scherson's orthogonal graphs, the binary hypercube can be formally modeled as an orthogonal graph as follows:

$$V = \{v | v \in 0, 1, \ldots, 2^n - 1\}$$

$$E = \{(u, v) | u, v \in V, u \otimes v = 1 \bullet m, m \in 0, \ldots n - 1\}.$$

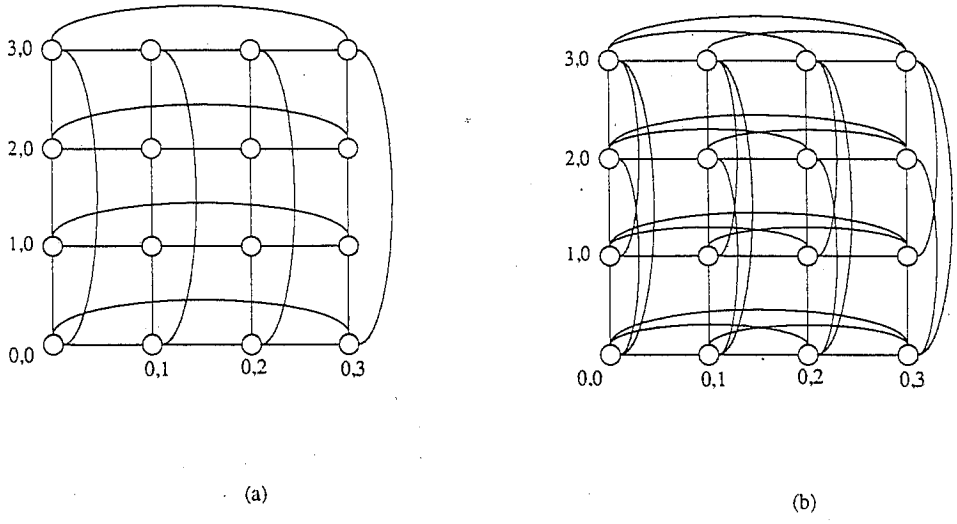
where $\otimes$ denotes an associative vector operator (in this case, the digit-by-digit exclusive-or of two binary vectors), and where $x \bullet y$ denotes $x \cdot 2^y$ (see [17]).

Binary hypercubes can also be generalized to have higher radices [1, 2]. A higher radix $d^n$ *generalized hypercube* can be formally modeled as an orthogonal graph where all nodes aligned along a dimension are interconnected with a clique, as follows:

$$V = \{v | v \in 0, 1, \ldots, d^n - 1\}$$

$$E = \{(u, v) | u, v \in V, u \otimes v = b \cdot (1 \bullet m), \quad m \in 0, \ldots, n - 1\}.$$

Here $b \in 0, 1, \ldots, d - 1$, and where $b \cdot (1 \bullet m)$ denotes $b \cdot (d^m)$ (see [17]).




(a) (b)

FIG. 1. (a) A $4^2$ mesh (with "wrap-around"). (b) A $4^2$ generalized hypercube.

## 3. HYPERMESHES

The $d^n$-hypermesh network can be modeled as an orthogonal hypergraph $H(V, Z)$ defined over a set of vertices $V$ and a set of "hyperedges" $Z$. There are $d^n$ vertices in the hypermesh, and each vertex has a unique integer address from 0, ..., $d^n - 1$. The integer address of each vertex can be viewed as a "coordinate vector" of $n$ elements in radix $d$, where the elements correspond to the vertices' coordinates in the $n$-dimensional Euclidean space:

$$\text{vertex coordinate vector} = (x_{n-1}, x_{n-2}, \ldots, x_1, x_0) \rightarrow$$
$$\text{vertex integer address} = x_{n-1} \cdot d^{n-1} + x_{n-2} \cdot d^{n-2} \cdots x_0 \cdot dc^0.$$

Each hyperedge is a set of $d$ vertices which are aligned along some dimension. Let each hyperedge represent a distributed multichannel switch with $d$ channels (i.e., $d$ buses), each capable of supporting one data transfer, such that the multichannel switch supports $d$ data transfers simultaneously. (Alternatively, one may model each channel as a hyperedge capable of supporting a single data transfer. This model would be useful when each switch had fewer or greater than $d$ channels.)

By definition, in a $d^n$-hypermesh there are $n$ dimensions, with $d^{n-1}$ hyperedges aligned along each dimension, for a total of $n \cdot d^{n-1}$ hyperedges. Each hyperedge can be assigned a unique identifier consisting of a vector of $n$ elements, where each element is either a radix-$d$ digit or a "wild-card" symbol $\Theta$ which denotes a variable, $\Theta \in (0, \ldots, d-1)$. The hyperedge identified by the label $z(x_{n-1}, \ldots, x_{m+1}, \Theta, x_{m-1}, \ldots, x_0)$ consists of the set of $d$ vertices as follows:

$$z(x_{n-1}, \ldots, x_{m+1}, \Theta, x_{m-1}, \ldots, x_0) \equiv \bigcup_{i=0}^{d-1} (x_{n-1}, \ldots, x_{m+1}, i, x_{m-1}, \ldots, x_0).$$

With this convention, a formal model of a $d^n$-hypermesh is given by:

$$V = \{v | v \in 0, 1, \ldots, d^n - 1\}$$
$$Z = \begin{Bmatrix} z(x_{n-1}, x_{n-2}, \ldots, x_1, \Theta) \cup \\ z(x_{n-1}, x_{n-2}, \ldots, \Theta, x_0) \cup \\ \vdots \\ z(\Theta, x_{n-2}, \ldots, x_1, x_0) \end{Bmatrix}$$

$\forall x_i \in (d-1, \ldots, 0)$ and $\forall 0 \leq i < n$.

In the above definition, it was assumed that the hypermesh is "regular" and that each vertex represents one Processing element. In practice, a hypermesh can be "irregular"; i.e., the number of nodes aligned along different dimensions could vary with each dimension. Furthermore, each vertex could represent a cluster of processing elements which are treated as a single vertex at this level of abstraction. The clustering approach can amortize the hyperedge cost over a larger number of processors, as is done in the Connection Machine.

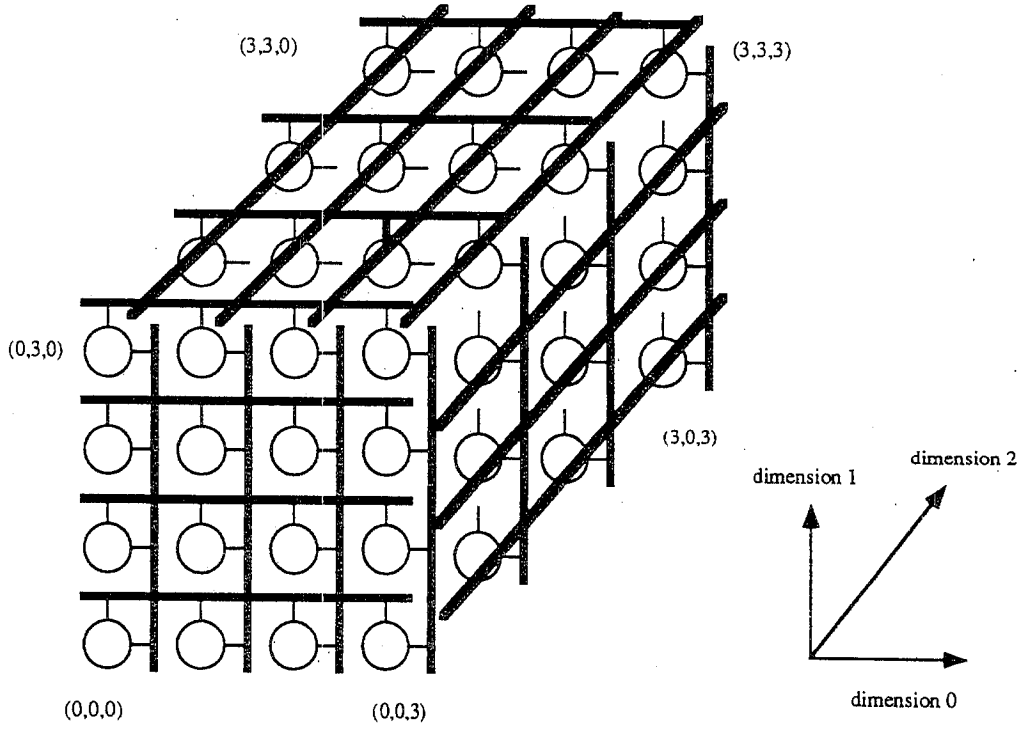
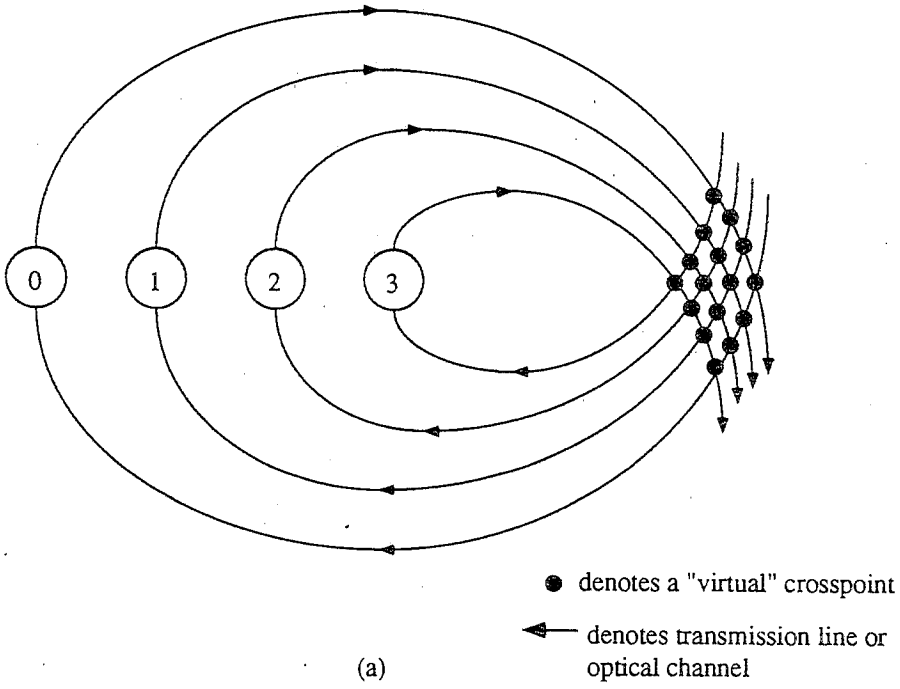
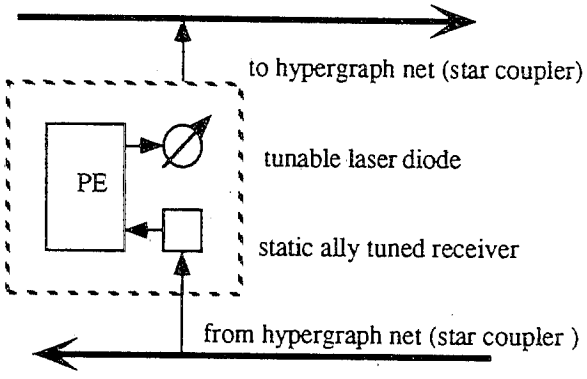



FIG. 2. A $4^3$ hypermesh. Thick bars denote hypergraph "hyperedges," which represent distributed crossbar switches.




(a)




(b)

FIG. 3. (a) A logical diagram of a distributed optical crossbar switch interconnecting $d$ PE nodes, with $2d$ directed edges and $d^2$ "virtual crosspoints." Crosspoints are set by tuning the optical component(s) to the desired wavelength. (b) A PE node in a 1D hypermesh, with a tunable laser and static optical receiver.

In a $d^n$ hypermesh, each vertex is a member of $n$ hyperedges, one in each of $n$ dimensions. The hyperedges have the ability to perform permutations over their members in one step, over $d$ independent data channels, as shown in Fig. 3a. However, certain implementations may also allow partial or complete broadcasts, partial or complete combining mappings, or various combinations, through different settings of virtual crosspoints. A hypermesh were each hyperedge performs broadcasts was proposed in [32] and called a "beta-hypermesh."

A model of a $4^3$ hypermesh is shown in Fig. 2, where hyperedges are represented by the thick bars. The routing algorithm in a hypermesh is similar to the routing algorithm in a conventional toroid. Each node compares its own address with the destination's; If the node's address and the destination tag differ in $m$ digits, then the packet can exit over either of the $m$ hyperedges corresponding to the dimensions in which the address and tag differ and follow a minimum distance path.

### *3.1. A WDM Optical Hypermesh with Electrical Switching*

A recent paper [28] funded by Darpa's Optical Computing Program has proposed an integrated *chip-set* for

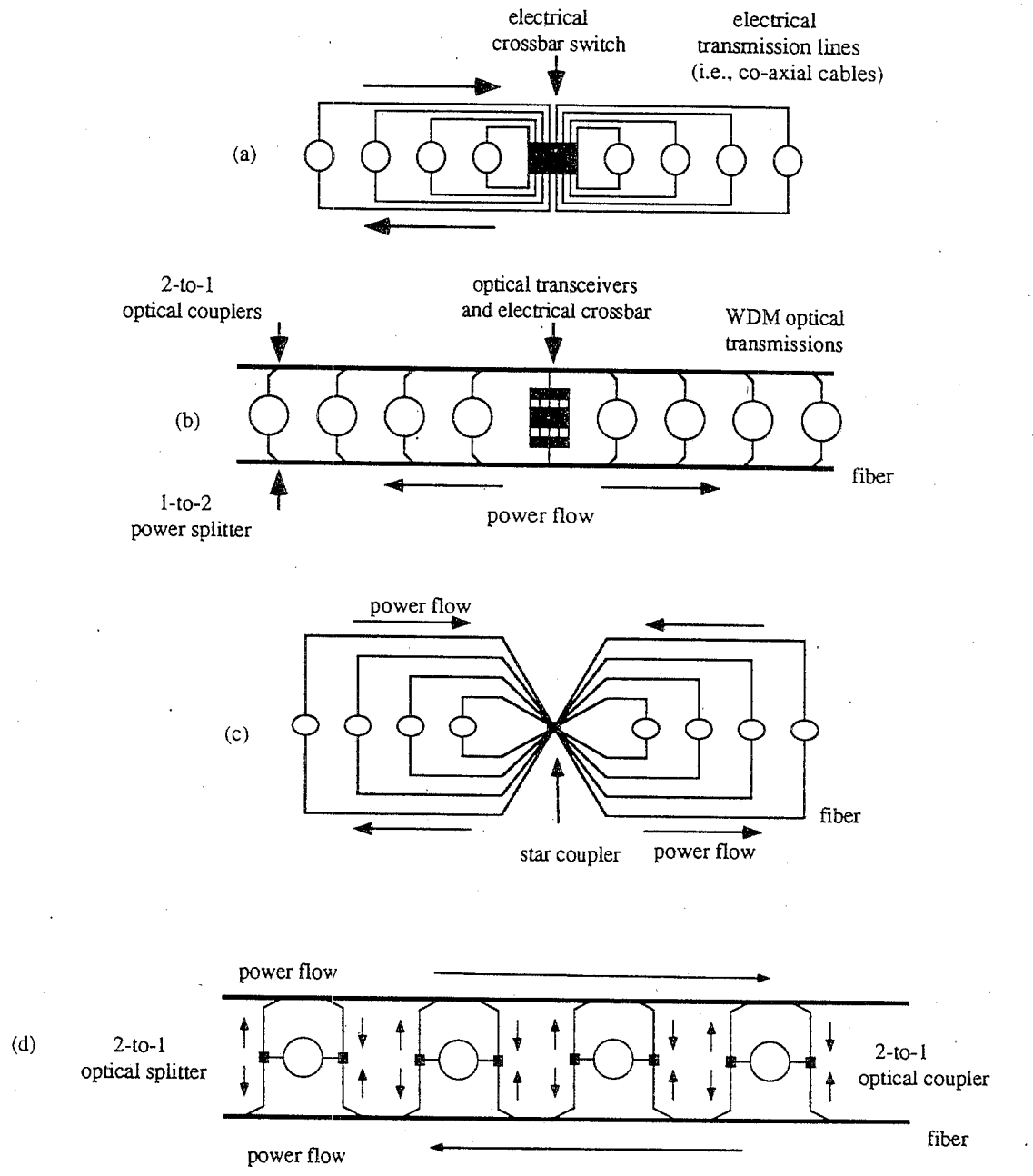



FIG. 4. Four implementations of a hyperedge: (a) electrical transmission with centralized electrical switching, (b) optical transmission with centralized electrical switching, (c) optical transmission with centralized optical switching over a large large star-coupler, and (d) optical transmission with distributed optical switching over two minimum distance fibers.

fiber-optic networks. The chip-set consists of (1) monolithic GaAs electrical-to-optical (EO) transceivers, (2) monolithic GaAs optical-to-electrical (OE) transceivers, and (3) an integrated silicon crossbar switch (32 × 32, but expandable to sizes 64 × 64 or 128 × 128) operating at approx. 1 Gbit/s per channel.

This integrated *chip-set* can be incorporated into a hypermesh hyperedge, as shown in Fig. 4a. The crossbar switch can be placed at the center of the linear array, with fibers emanating to and from every node in the array. One potential drawback of this construction is the large number of fibers which interconnect the nodes to the centralized crossbar. A hyperedge with $d$ nodes has a total of $2d$ directed transmission lines, $d$ leading from all nodes into the crossbar chip from above, and $d$ exiting the crossbar from below and leading back to all nodes.

The use of WDM with nontunable components can reduce this complexity considerably, as shown in Fig. 4b. There are two fibers, one carrying transmissions from the nodes into the crossbar, and visa versa. Each fiber in the net now carries up to $d$ simultaneous transmissions over $d$ distinct wavelengths $\{\lambda_0, \ldots, \lambda_{d-1}\}$; each node with ad-

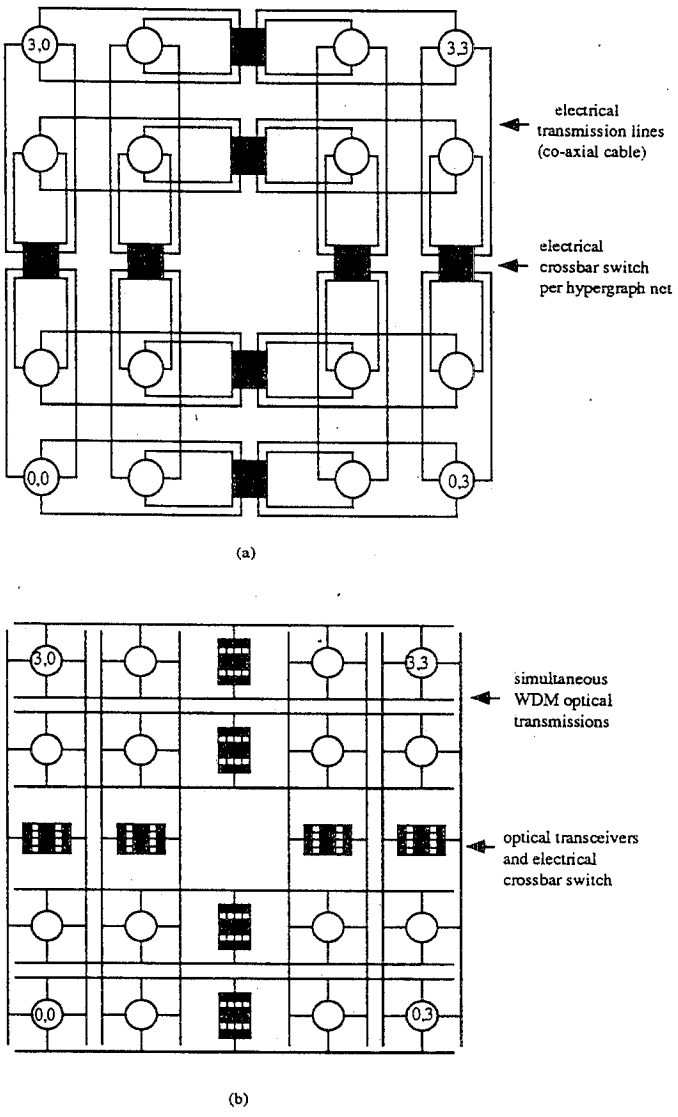



FIG. 5. (a) A 2D hypermesh with electrical switching and electrical transmission. (b) A 2D hypermesh utilizing a chip-set consisting of integrated monolithic OE and EO converters and integrated electrical crossbar switches.

dress $i$ can transmit its data to the localized crossbar on a separate reserved wavelength $\lambda_i$ using a nontunable laser diode. The optical transmissions are converted to electronics, switched in the crossbar, converted back into the optical domain, and transmitted back to the destination nodes, again over separate reserved wavelengths using nontunable laser diodes. (In general, each hyperedge in a $d^n$ hypermesh carries $O(d)$ wavelengths). Two possible hypermesh implementations using this chip-set are shown in Fig. 5.

### 3.2. A WDM Optical Hypermesh with Optical Switching

One-dimensional WDM optical crossbar-like switches form the basic building blocks in a large, multidimensional WDM optical hypermesh. Representative WDM optical switches which can be used as the basic building blocks have been described in [12, 22, 26, 27]. These switches typically are small in size and require fast multiple access protocols to resolve contention.

Let each node $i$ in the hyperedge ($0 \le i \le d-1$) have a laser tunable over the $d$ wavelengths $\{\lambda_0, \ldots, \lambda_{d-1}\}$ and a nontunable receiver tuned to wavelength $\lambda_i$ Such a node design is shown in Fig. 3b. Such nodes are interconnected with fiber, as shown in Fig. 4c. The optical transmissions from all nodes are merged in the large centralized star coupler, which broadcasts a fraction of the power back to each node. Node $i$ can transmit to node $j$ by tuning its laser to wavelength $\lambda_j$ and initiating the transmission. In absence of an optical "collision" over wavelength $\lambda_j$, the destination node $j$ will receive the data. All $d$ nodes on the net can perform data transfers *simultaneously*, over separate wavelengths. When multiple senders transmit on the same wavelength simultaneously optical collisions will occur, thereby necessitating the use of a "multiple-access protocol" to control access to each wavelength. Efficient WDM multiple access protocols, in terms of minimizing the overhead for contention resolution and maximizing the traffic carried by the optical crossbar, are described in [12, 22, 26, 27].

In Fig. 4c, the optical coupler is located in the center of the linear array of $d$ nodes, and $2d$ fibers are required to link the nodes with the coupler. The "interconnection complexity" can be reduced significantly by exploiting WDM, as shown in Fig. 4d. In Fig. 4d, there are two fibers, carrying "upstream" and "down-stream" traffic, respectively; the transmission from a node enters both fibers. Each linear fiber collects the optical transmission as it passes by each node, requiring a two-to-one optical coupler at each node. Each fiber also delivers a small fraction of its optical power to each node that it passes, requiring a one-to-two power splitter. Node $i$ can transmit to node $j$ by tuning its laser to wavelength $\lambda_j$ and initiating the transmission. In absence of an optical "collision" over wavelength $\lambda_j$, the destination node $j$ will receive the data, as before. Again, all $d$ nodes on the hyperedge can perform permutations *simultaneously*, over separate wavelengths.

The configuration in Fig. 4d has a number of attractive features. First, it eliminates the large centralized star-coupler required in Fig. 4c (which tend to be very expensive) and replaces it with many smaller components (one-to-two and two-to-one couplers, which tend to be cheaper). Second, it *minimizes the propagation delay*; all fibers are "straight lines" with no internal fiber loops. Hence, the propagation delay between any pair of nodes is proportional to their horizontal distance only and it is minimal. This configuration is particularly useful when executing deterministic algorithms such as sorting [13] or the FFT [14]. Our analysis in Section 6 indicates that propagation delays can become nonnegligible in hyper-

meshes under certain circumstances and the configuration in Fig. 4d minimizes these delays.

Given the close proximity of all the optical components (a few meters in a typical multi-processor), optical power attenuation will not pose a problem. A single laser diode can provide the power necessary for the configurations in Fig. 4. A laser diode transmitter typically provides a few milliwatts of optical power, whereas each receiver is typically sensitive to a fraction of a microwatt of power. Assuming each fiber tap in Fig. 3d reduces the power flow by 5% (this figure can be improved by using a better tap-design) and the total attenuation is no greater than 1000, then the linear array could contain roughly 128 nodes.

### 3.3. Asymptotic Properties of Hypermeshes

The crossbar, crosspoint and transmission line costs of various networks are shown in Table I. All distributed crossbar switches interconnecting $d$ nodes in a hyperedge are assumed implemented as shown in Fig. 3a: There are $d$ nodes, with $2d$ directed edges and a total of $d^2$ crosspoints. The crosspoints can be real in electrical implementations or "virtual" (i.e., nonexistent) in optical implementations.

*Hypercubes and Generalized Hypercubes.* The costs of a binary hypercube are well known. In a $d^n$ generalized hypercube [1], each routing node has degree $n \cdot (d-1) + 1$, since it has a directed edge to $(d-1)$ other nodes in each of $n$ dimensions, plus one link to the associated processor, and the remaining figures are easily verified. See [1] for details.

*Hypermeshes.* A $d^n$ hypermesh (with size $N = d^n$) has $d^{n-1} = N/d$ hyperedges in each dimension. There are $n$ dimensions, and hence $nN/d$ hyperedges in total. The total crosspoint cost depends on the vertex construction. In one possible vertex implementation, access to each hyperedge is via transmit and receive buffers, and the switching of data from one incoming hyperedge to another outgoing hyperedge is accomplished by the processor within the vertex; no crossbar switch is needed within the vertex. Alternatively, each vertex in a $d^n$ hypermesh could use a small $n \times n$ crossbar to allow data incoming on one hyperedge to be switched in hardware and sent out over another hyperedge simultaneously and without processor intervention. This additional crossbar does not add to the capability of hypermeshes significantly, when executing many standard algorithms such as sorting. With this added crossbar within each vertex, the asymptotic crosspoint cost of the entire hypermesh is

$$N \cdot (n+1)^2 + (nN/d) \cdot d^2 = O(Nn \cdot (n+d)).$$

Without the added crossbar, the asymptotic crosspoint cost is slightly lower; when $n \ll d$, the difference in crosspoint count is insignificant:

$$(Nn/d) \cdot d^2 = O(Nnd).$$

TABLE I
Exact Costs

| Network type | Directed links | Crosspoints |
|---|---|---|
| $2^{\log_2 N}$-Hypercube | $N \cdot (\log_2 N + 1)$ | $N \cdot (\log_2 N + 1)^2$ |
| $d^n$-Hypercube | $N \cdot (n(d-1)+1)$ | $N(n(d-1)+1)^2$ |
| $d^n$-Hypermesh | $N \cdot 2n$ | $Nn(n+d)$ |
| $d^n$-Mesh | $N \cdot 2(n+1)$ | $N \cdot (n+1)^2$ |

TABLE II
Graph-Theoretic Properties

| Network type | Neighbors | Diameter |
|---|---|---|
| $2^{\log_2 N}$-Hypercube | $\log_2 N$ | $\log_2 N$ |
| $d^n$-Hypercube | $n(d-1)$ | $n$ |
| $d^n$-Hypermesh | $n(d-1)$ | $n$ |
| $d^n$-Mesh | $2n$ | $nd/2$ |

*Note.* A hypermesh implementation where each PE node has a small $n \times n$ crossbar is assumed. In practice, the degree of the graph-based networks is increased by 1 to allow the PE to access each router node.

*Graph-Theoretic Definitions.* The distance between any pair of nodes in a graph is defined as the number of edges which a shortest path between the two traverses [4]. The diameter of a graph is defined as the maximum distance between any pair of nodes [4]. The distance between a pair of nodes in a hypergraph can be similarly defined as the fewest number of hyperedges which a shortest path between the two traverses [3]. The diameter of a hypergraph can be defined as the maximum distance between any pair of nodes [3]. Table II illustrates the number of neighbors and diameter in hypercubes, meshes and hypermeshes with equivalent size, based on these conventional graph-theoretic definitions. The hypermesh diameter is significantly smaller than many others and comparable to that of the generalized hypercubes. It should be noted that these graph-theoretic definitions do not model the real physical distance over the edges or hyperedges. To determine the real delay in transferring a packet between the furthest nodes one should multiply the graph-theoretic distances by the "packet cycle time" for each network (defined in Section 2). This approach is used in Section 6.

## 4. COMBINATORIAL PROPERTIES OF HYPERMESHES

In this section, the combinatorial properties of the formal model of hypermeshes which are relevant to parallel

computing are examined. These properties include permutation capability, rearrangeability, partitionability, and embedding capability. Since this class of network is relatively new a formal characterization of its capabilities is required.

### 4.1. Permutation Capability

Let $\Omega_{d,n}$ denote a $d^n \times d^n$ omega network, with $d^n$ input or output terminals. This network requires $n$ stages of $d \times d$ crossbar switches, with $d^{n-1}$ crossbar switches per stage and with a "radix-$d$ shuffle" wiring permutation before each stage. To avoid the proliferation of symbols, also let $\Omega_{d,n}$ denote the permutations realizable by the same network; the actual meaning, either the network or its permutation capability, will be clear from the context. The radix-$d$ shuffle of $N$ elements, denoted $\rho_d$, is defined as follows [1]:

$$\rho_d = \left(d \cdot x + \left[\frac{d \cdot x}{N}\right]\right) \bmod N.$$

The radix-2 shuffle of $N$ elements is equivalent to the binary perfect shuffle, which corresponds to a circular left rotation of the index bits,

$$\rho_2[x_{n-1}x_{n-2} \cdots x_1x_0] = x_{n-2}x_{n-3} \cdots x_0x_{n-1},$$

where $x_i$ denotes a bit. Define the $i$-th binary "butterfly" permutation as

$$\beta_i[x_{n-1}x_{n-2} \cdots x_i \cdots x_1x_0] = x_{n-1}x_{n-2} \cdots x_{i+1}x_0x_{i-1} \cdots x_1x_i,$$

where each $x_i$ represents a bit. Using Parker's notation [10], let $E^y$ denote the class of butterfly permutations satisfying

$$\text{if } E[x_{n-1} \cdots x_y \cdots x_0] \rightarrow E[x_{n-1} \cdots \overline{x_y} \cdots x_0]$$
$$\text{then } E[x_{n-1} \cdots \overline{x_y} \cdots x_0] \rightarrow E[x_{n-1} \cdots x_y \cdots x_0].$$

Thus $E^y$ consists of the set of permutations achievable by data exchanges over dimension $y$ in a binary hypercube.

Let $A$ and $B$ represent two classes of permutations, and let $\pi_1 \in A$ and $\pi_2 \in B$. Then let $A \cdot B$ denote the class of permutations where $\pi \in A \cdot B \rightarrow \pi(i) = \pi_2(\pi_1(i))$. In the following we will summarize two known properties using this notation, and prove some relevant properties.

PROPERTY 1. $\Omega_{2,\log N} = E^{n-1} \cdots R^0$.

PROPERTY 2. $\Omega^{-1}_{2,\log N} = E^0 \cdots R^{n-1}$.

Properties 1 and 2 indicate that the binary omega and omega-inverse networks can simulate hypercube dimensions in decreasing or increasing orders, and were noted by Stone, Lawrie, and Pease in the 1970s. Its is known that binary omega networks can act as concentrators, merging networks, "compact broadcast" networks, etc. Parker used algebraic manipulation of permutations to formally prove Properties 1 and 2 for binary omega networks.

However, neither Stone, Lawrie, Pease, nor Parker quantified the permutation capability of higher-radix omega networks. For completeness, the following two properties prove that the higher-radix omega networks have a superset of the permutation capability of the binary omega networks, and that they too can realize many useful permutations, can act as concentrators, merging networks, compact copy networks, etc.

PROPERTY 3. $\Omega_{d,n} \subset E^{n'-1} \cdots E^0$, $\forall d = 2^j$, $1 \le j \le n' = \log_2 N$, $d^n = N$.

PROPERTY 4. $\Omega^{-1}_{d,n} \subset E^0 \cdots E^{n'-1}$, $\forall d = 2^j$, $1 \le j \le n' = \log_2 N$, $d^n = N$.

*Proofs.* Label the $N$ input (output) terminals in stage $s$ from $0 \cdots N-1$ from top to bottom. Label the crossbar switches in stage $s$ from 0 to $N/d - 1$ from top to bottom. Let $\pi_s$ denote the permutation of the $N$ indices of the $N$ input elements which appear at the input terminals to stage $s$ for $1 \le s \le n$.

Given an identify permutation at the input of the network, and assuming that all prior stages perform the identity permutation, then $\pi_s$ is given by $s$ repeated applications of the $d$-way shuffle, i.e.,

$$\pi_s(x_{n-1}, \ldots, x_0) = \rho_d^s(x_{n-1}, \ldots, x_0) = x_{(n-1-s) \bmod n}, x_{(n-2-s) \bmod n}, \ldots, x_{0-s) \bmod n},$$

where $x \bmod n = x$ if $x \ge 0$ and $x \bmod n = x + n$ if $x < 0$.

The indices of the input data appearing at the $d$ input terminals incident to a crossbar switch with label $z_{n-1}z_{n-1} \cdots z_1$ in stage $s$ are therefore given by the inverse of the above permutation, i.e.,

$$\pi_s^{-1}(z_{n-1}, \ldots, z_1, i) = \rho_d^{-s}(z_{n-1}, \ldots, z_1, i);$$

i.e., the permutation appearing at the first stage of switches is given by

$$\rho_d^{-s}(z_{n-1}, \ldots, z_1, i) = i, z_{n-1}, \ldots, z_1,$$

and the permutation appearing at stage $s \ge 1$ is given by

$$\rho_d^{-1}(z_{n-1}, \ldots, z_1, i) = z_{s-1}, \ldots, z_1, i, z_{n-1}, \ldots, z_s.$$

Each crossbar in the each stage can permute its $d$ inputs arbitrarily, and all $d$ inputs incident to each crossbar differ in only one radix $d$ digit. Thus the first stage performs a superset of the following permutations:

$$E^{n'-1} \cdot E^{n'-2} \cdots E^{n'-\log_2(d)}.$$

Subsequent stages are handled similarly. It follows that all $n$ stages perform the following permutations:

$$\Rightarrow \Omega_{d,n} \subset E^{n-1} \cdot E^{n-2} \cdots E^0.$$

Using the permutation identity $(A \cdot B)^{-1} = B^{-1} \cdot A^{-1}$, then it follows that

$$\Rightarrow \Omega_{d,n}^{-1} \subset E^0 \cdot E^1 \cdots E^{n-1}. \quad \blacksquare$$

The advantages of these expressions over Parker's [10] are (1) they apply for higher-radix Omega networks, and (2) they express the permutation capability in terms of conflict-free exchanges over hypercube dimensions, using the $E^i$ notation. A binary Omega network is shown in Fig. 6a. Figure 6b illustrates an embedding of the $\Omega_{2,4}$ network into the $\Omega_{4,2}$ network, illustrating *graphically* the ability of the latter to realize a superset of $E^3 \cdot E^2 \cdot E^1 \cdot E^0$.

Define a "variable-radix" omega network as one where the crossbar degree may vary across different dimensions. An $N \times N$ variable-radix omega network is specified by a series of radices $r_{n-1}, r_{n-2}, \ldots, r_0$ such that $N = r_{n-1} \times r_{n-2} \times \cdots \times r_0$. In a multistage implementation of a variable-radix omega network, every radix $r_i$ corresponds to a stage of radix $r_i$ crossbar switches preceded by a radix-$r_i$ shuffle wiring pattern (see [2]).

PROPERTY 5. *Provided that* $\log_2(r_i) \in \{1, 2, \ldots\}$ $\forall 0 \leq i \leq n-1$, *then the permutations realizable by the variable-radix omega network is a superset of those realizable by the binary omega network.*

*Proof.* The proof follows from that of property two by generalization and is not reproduced here. ■

PROPERTY 6. *A $d^n$-hypermesh (where $d = 2^j$ for some $j \geq 1$) can simulate $\Omega_{d,n}$ (and $\Omega_{d,n}^{-1}$), with all routes traversing only minimum distance paths. (The same applies to a variable-radix hypermesh where the radices are powers of* 2.)

*Proof.* Follows from the proof of properties 3 and 4. ■

Define a dual-hypermesh as a hypermesh where each hyperedge can support two simultaneous transmissions from each member vertex and two simultaneous receptions into each member vertex.

PROPERTY 7. *The dual $d^n$ hypermesh is rearrangeably nonblocking. Equivalently, any permutation can be realized within $2 \cdot n - 1$ dimension traversals.*

*Proof.* Follows from property 6, by simulation of the $2n - 1$ stage Clos network. ■

The paths taken through the dual-hypermesh (or dual-hypercube) can be found by routing the same permutation through the $\Omega_{2,n}\Omega_{2,n}^{-1}$ network, which can be done with the usual looping algorithm [8]. There is an obvious one-to-one correspondence between paths in the multistage network and paths in the hypermesh (or hypercube).

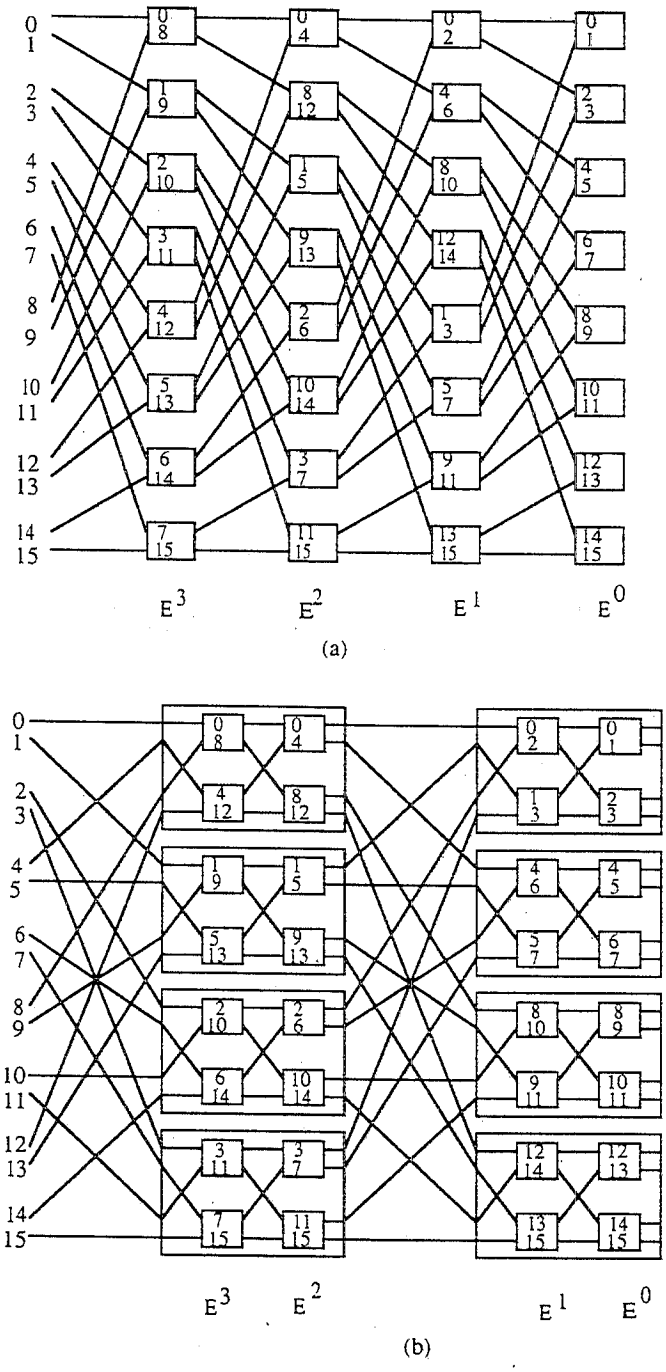



FIG. 6. (a) A $2^4 \times 2^4$ Omega network, along with its ability to simulate hypercube dimensions in decreasing order. (b) A higher radix $4^2 \times 4^2$ omega network, along with its ability to embed and simulate the radix-2 omega network.

PROPERTY 8. *The $d^n$ hypermesh is recursively partitionable into independent subnetworks with arbitrary quantized sizes, in size increments of $d^{n-1}$.*

*Proof.* (See [12].) For example, a regular 4096 node hypermesh can be partitioned many ways regardless of

its radix and dimensionality. A nonrecursive partitioning includes {1024, 1024, 1024, 1024} or {1024, 3072}. Recursive partitionings include for example {2048, 1024, 512, 256, ... }. Also, the fact that hypermeshes can have variable radices allows one to partition hypermeshes into smaller hypermeshes whose sizes are not necessarily powers of 2, which can be useful when embedding guest graphs. ■

The previous properties indicate that all algorithms and properties of omega networks apply equally to hypermeshes. In fact, hypermeshes can be thought of as Omega networks in which packets only traverse the necessary dimensions, rather than all dimensions.

### 4.2. Graph Embeddings

By definition an *embedding* of a guest graph $G_1(V_1, E_1)$ into a host graph $G_2(V_2, E_2)$ is a one-to-one mapping:

$$\sigma(u) \rightarrow v, \quad u \in V_1, v \in V_2|$$
$$e(u_1, u_2) \in E_1 \rightarrow e(\sigma(u_1), \sigma(u_2)) \in E_2.$$

A "dilation-$k$ embedding" of $G_1(V_1, E_1)$ into $G_2(V_2, E_2)$ requires a 1-to-1 assignment of $V_1$ onto $V_2$ and a one-to-many mapping of $E_1$ onto $E_2$, such that nearest neighbors in $G_1$ are no further than $k$ edge traversals apart in $G_2$ (i.e., a single edge in $G_1$ may map to no more than $k$ edges in $G_2$). The *load* of an embedding is defined as the maximum number of guest graph nodes mapped onto a host graph node. The *congestion* of an embedding is defined as the maximum number of guest graph edges mapped onto a host graph edge. (For example, see [29] for definitions.)

Since hypergraphs have no edges between only two vertices, it is not possible to define the "embedding" of a graph onto a hypergraph $H(V, Z)$ in this precise manner. Therefore, define a dilation-$k$ embedding of a graph $G_1(V_1, E_1)$ onto a hypergraph $H_2(V_2, Z_2)$ as a one-to-one mapping of $V_1$ onto $V_2$ and a one-to-many mapping of $E_1$ onto $Z_2$, such that nearest neighbors in $G_1$ are no further than $k$ hyperedge traversals apart in $H$ (i.e., a single edge in $E_1$ may map to no more than $k$ hyperedges in $Z_2$). Define the *load* of a hypergraph embedding as before, and define the *congestion* of a hypergraph embedding as the maximum number of guest graph edges of the form $(u, v)$ (for fixed $v$) which map onto a single hyperedge containing $v$, maximized over all $v$ and all hyperedges. The justification for this definition is as follows: If there are $C$ such guest graph edges and if all $C$ guest graph edges were to perform a data transfer to $v$ simultaneously, the same transfer in the hypergraph would cause a congestion problem and the hyperedge would be forced to perform the transfers to $v$ sequentially.

THEOREM 1. *Any M-node* graph (*the "guest" graph*) *which can be embedded into an N-node hypercube with dilation k, congestion C and load L can be embedded into a $d^n$-hypermesh with dilation $\leq k$, with a congestion of $(C \cdot \log_2 N/\log_d N)$ and a load L.*

*Proof.* Suppose the mapping is preserved and consider some arbitrary node with label $x$ in both the hypercube and the hypermesh. Since the hypercube congestion is at most $C$, then the maximum number of guest graph edges incident to this node is $\leq C \cdot \log_2 N$. The number of hyperedges incident to node $x$ is $n = \log_d N$. The result follows. ■

THEOREM 2. *Any hypercube algorithm in which all nodes perform (or do not perform) a data transfer over the same dimension simultaneously will operate without conflict on a hypermesh.*

*Proof.* From properties 1, 2, 3, and 4. ■

The significance of Theorem 2 is that any potential increase in hyperedge congestion obtained when a hypercube algorithm is mapped onto a hypermesh will not result in any blocking in practice, provided that the hypercube dimensions are used sequentially. A majority of hypercube algorithms have precisely this property (i.e., see [29], where many such algorithms are described).

EXAMPLE 4. Any $M_1 \times M_2 \times \cdots \times M_k = M$ node array can be embedded into an $N$-node hypercube with load 1, dilation 1, and congestion 1 [29], where

$$\log_2 N = \sum_{i=1}^{k} [\log_2(M_i)].$$

It follows from Theorem 1 that the same array can be embedded into an $N$-node $d^n$ hypermesh with load 1, dilation 1, and a congestion of at most $(\log_2 N/\log_d N)$.

EXAMPLE 5. Any $M_1 \times M_2 \times \cdots \times M_k = M$ node array, where $M_j = 2^{f(j)}$ and where $f(j) \in 1, 2, \ldots, \forall 1 \leq j \leq k$, can be embedded into an $N$-node hypercube with dilation 1, congestion 1, and load 1 [29], where

$$\log_2 N = \sum_{i=1}^{k} f(i).$$

It follows from Theorem 1 that the same array can be embedded into an $d^n$-node hypermesh, where $d^n \geq N$, with load 1, dilation 1 and a congestion of at most $(\log_2 N/\log_d N)$.

EXAMPLE 6. A $2N - 1$ node complete binary tree can be embedded into an $N$- node hypercube with load $\log_2 N$ and dilation 1 [29], such that edges in every level of the tree are mapped onto one dimension of the hypercube. Theorem 2 states that any algorithm which traverses the tree one level at a time will execute conflict-free on a hypermesh.

The previous examples indicate that many of the known results obtained for graph embeddings onto a hypercube can be used directly for hypermesh. However, it is likely that new embeddings designed specifically for the hypermesh will out perform those designed for the hypercube.

## 5. STOCHASTIC PERFORMANCE

An *exact analysis* for the *expected queueing time* of meshes, hypercubes and hypermeshes is presented. The analysis will illustrate the effects of architectural changes on the queueing time, regardless of the implementation technology, based on an analytic model and modeling assumptions proposed by Kleinrock [30].

An analysis for a generic $M$-channel $N$-node communication network was derived in [30]. The $i$-th channel is associated with an $M/M/1$ queueing system with Poisson arrivals at a rate $\lambda_i$ and with independent exponential service time of mean $1/(\mu \cdot c_i)$ (seconds). $1/\mu$ is the average message size and $c_i$ is the capacity of the $i$th channel. With these assumptions, the network is "product-form," and Jackson's theorem on product-form networks can be used to yield the expected delay [30]. The following assumptions are made (for a justification see [30]): (a) each node is equally likely to send a message to every other node in a fixed period of time; (b) the routing algorithm traverses dimensions in a static order; (c) the load is evenly distributed, i.e., $\lambda_i = \lambda_j$, $\forall i \neq j$; (d) the capacities have been optimally assigned; and (e) the cost per capacity per channel is unity. The analysis in this section considers only the "queueing time," i.e., the expected time spent waiting in the $M/M/1$ queues.

Under these assumptions, an exact expression for the expected queueing time in an $M$-channel $N$-node network is given by [30]

$$T = \bar{k}\left(\sum_{i=1}^{M} \sqrt{\frac{\lambda_i}{\lambda}}\right)^2 \Big/ \mu C(1 - \bar{k} \cdot \gamma),$$

where

- $\bar{k}$ is expected number of $M/M/1$ queues encountered by a packet, under randomly distributed traffic in the topology;
- $\lambda = \sum_{i=1}^{M} \lambda_i = M\lambda_i$;
- $\lambda$ is; the utilization factor
- $C = \sum_{i=1}^{M} C_i$ = aggregate bandwidth (or "capacity") of the topology

This exact closed-from expression can be used to evaluate the effects of architectural changes, regardless of the implementation technology (electrical or optical). By symmetry, in all graph-based networks considered the expected queueing time can be simplified to yield

$$T = \frac{\bar{k} \cdot M}{\mu C(1 - \bar{k} \cdot \gamma)},$$

where $M$ is the number of transmission lines in the network (see Table III). By inserting the expression for $M$ and factoring out the common term $N/(\mu \cdot C)$ the exact queueing times for the graph-based networks are easily obtained.

For $d^n$ meshes, the expected queueing time is given by

$$T = \frac{n \cdot (n + 1) \cdot d/2}{(1 - (nd/4) \cdot \gamma)},$$

where $d$ is the radix, $n$ is the dimensionality, and $\bar{k} = n(d/4)$. For a binary hypercube, the expected queueing time is given by

$$T = \frac{n \cdot (n + 1)/2}{(1 - (n/2) \cdot \gamma)}$$

where $d = 2$, $n = \log_2 N$, and $\bar{k} = n/2$. For generalized higher radix $d^n$ hypercubes, the expected queueing time is given by

$$T = \frac{(n(d - 1) + 1) \cdot n(d - 1)/d}{(1 - (n(d - 1)/d) \cdot \gamma)} \geq \frac{n^2(d - 1)^2/d}{(1 - (n(d - 1)/d) \cdot \gamma)},$$

where $\bar{k} = n(d - 1)/d$. (See [1] for a slightly different derivation for generalized hypercubes made under similar assumptions.) The exact analysis of a hypermesh depends on the precise implementation details. To capture

TABLE III
Inter-PE Bandwidth and Transmission Delay

| Network type | No. directed links | Link BW ($B$) | Transmission delay ($P/B$) |
|---|---|---|---|
| $d^n$-Mesh | $N \cdot 2(n + 1)$ | $C/(N \cdot 2(n + 1))$ | $P \cdot N \cdot 2(n + 1)/C$ |
| $d^n$-Hypercube | $N \cdot (n(d - 1) + 1)$ | $C/(N \cdot (n(d - 1) + 1))$ | $P \cdot N \cdot (n(d - 1) + 1)/C$ |
| $d^n$-Hypermesh | $N \cdot 2n$ | $C/(N \cdot 2n)$ | $P \cdot N \cdot 2n/C$ |

*Note*. All networks have equivalent aggregate bandwidth = $C$ bit/s; packet size = $P$ bits.

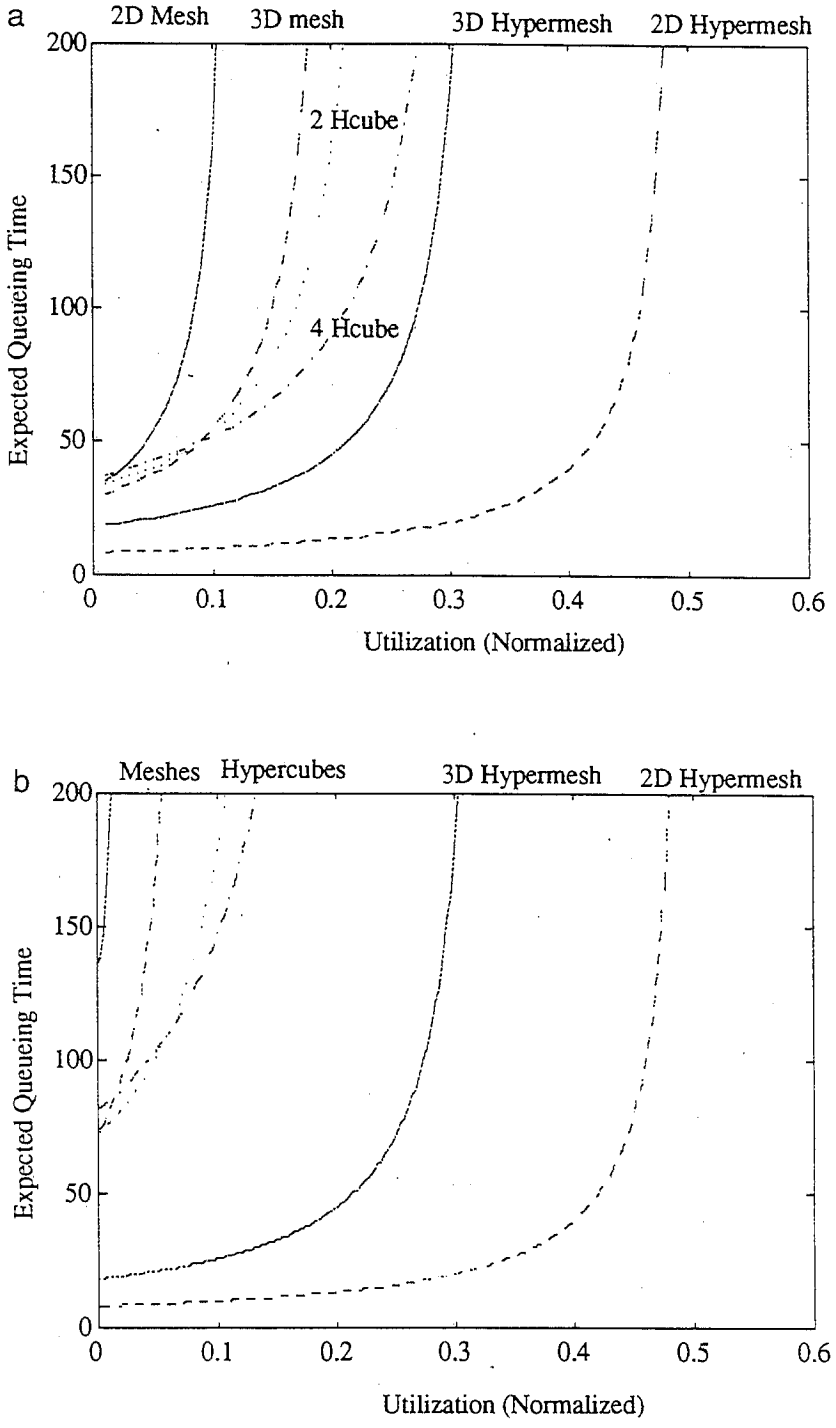



FIG. 7. Exact expected queueing time in meshes, hypercubes, and hypermeshes: (a) normalized delay vs utilization for $N = 256$ nodes; (b) normalized delay vs utilization for $N = 4096$ nodes.

the essential characteristics of the architecture and to avoid implementation details, let the $d^n$ hypermesh simulate a $d^n$ omega network, so that packets traverse all dimensions in decreasing order. Assume the buffering is within the hyperedge, and within a vertex data arriving on dimension $i$ is immediately sent out on dimension $i - 1$. Hence, the expected queueing time in the hypermesh is given by

$$T = \frac{2n^2}{(1 - n\gamma)},$$

where $\bar{k} = n$.

The expected queueing time of the these network architectures are shown in Fig. 7, under the constraint of equivalent aggregate bandwidth. For each network, the queueing time increases rapidly with increasing utiliza-

tion and saturates at a particular load given by $\gamma_{sat} = 1/\bar{k}$. These curves illustrate the maximum traffic handling capability of each architecture (the point of saturation); according to Fig. 7, the traffic capability of the hypermesh is considerably higher than that of the meshes and hypercubes, and the queueing time of hypermeshes is considerably lower.

We point out that Kleinrock's model only yields the expected queueing time and it is primarily useful for identifying the maximum traffic handling capability of each architecture. The appeal of Kleinrock's model is that the network size $N$, the aggregate capacity $C$ and the average message size all factor out, leaving an exact analytic model which depends only on the topological characteristics of the network architecture [30]. In the next section we analyse the performance of a specific algorithm, considering node latencies and propagation delays.

## 6. TEMPORAL COMPLEXITY OF SORTING ON MESHES, HYPERMESHES, AND HYPERCUBES

Consider a specific parallel algorithm, parallel sorting [19, 33], on these three network architectures, under the constraint of equivalent aggregate bandwidth. In this section, we allow arbitrary node latencies, arbitrary propagation delays over fiber, and durations of the data transfer step and computation steps.

The equivalent aggregate bandwidth constraint reflects many realistic engineering constraints in network design. The limited "pin-out" constraint of existing ICs often becomes a dominant constraint of system performance. Existing "pin-grid-array" (PGA) ICs have a limited number of IO pins, and since each pin has a limited clock rate, the aggregate bandwidth of any one IC is limited. We point out that the *equivalent pin-out* constraint considered in [34] is *identical* to the *equivalent aggregate bandwidth* constraint.

As a constraint, there exist $N$ fiber-optic chip sets similar to those described in [28]; the goal is to design an optimal communications network for parallel processors using these chip sets. The crossbars in [28] have degree 16 and are cascadable to have degree 32 or 64, and hence thousands of such small crossbars will be required to implement a large network. We will consider three different architectures, meshes, hypercubes, and hypermeshes, all using the same number of chip-sets. Each transceiver and crossbar chip has a bandwidth of 16 Gbit/s, and each crossbar IO port has a bandwidth of 1 Gbit/s, representing roughly the "state-of-the-art" in IC and fiber-optic laser diode technology [26–28].

Let each PE in the $N$-node mesh or $N$-node hypercube utilize one chip-set (crossbar and transceivers) to implement a hardware *router chip*, which handles the routing and switching functions. The bandwidth of an inter-PE link in meshes and hypercubes is given by $C/L$, where $C$ is the aggregate bandwidth of the network and where $L$ is the number of transmission lines in each network. The bandwidth of an inter-PE connection across a hyperedge in a hypermesh is also given by $C/L$, where $L$ is now the number of transmission lines in the hypermesh. In each network (mesh, hypercube, hypermesh), the aggregate bandwidth is held constant at $C = 16 \cdot N$ Gbit/s, since each implementation uses the same number of chip-sets. See Table III for the transmission delays in various networks.

A data-flow graph for Batcher's bitonic sorting algorithm is illustrated in Fig. 8a, using a standard notation for sorting networks. Assuming that data-flow graph vertex $i$ is mapped onto hypercube vertex $i$, then all the data transfers correspond to permutations of the form $E^i$ for $0 \le i < \log N$ (see Fig. 8b). The flow-graph describes the required data movements only, and it can implemented with either *message-passing* or *shared-memory* multiprocessors.

We will adapt the convention of Thompson and Kung [19]: Let $T_R$ denote the time it takes to send a packet of data from one PE to a nearest neighbor, and let $T_C$ denote the time taken to perform a computation (in this case, a floating-point compare and possibly exchange). The number of data-transfer and computation steps required by the hypercube or hypermesh is given by $(\log_2 N)(1 + \log_2 N)/2$. The number of data-transfer and computation steps required by Thompson and Kung's shuffled bitonic sort algorithm on 2D meshes and on higher dimensional meshes are shown in Table IV.

The time to perform parallel sorting in each network is equal to the number of steps required multiplied by the time required to implement each step. The time required to perform each data transfer step is the "packet cycle time" $T_R$ defined in Section 2, which includes the trans-

TABLE IV
Complexities of Various Sorting Algorithms

| Network type | No. data transfer steps | No. computation steps |
|---|---|---|
| 2D-Mesh ($d = \sqrt{N}$) | $14(d-1) - 8\log_2(d)$ | $2\log_2^2(d) + \log_2(d)$ |
| $d^n$-Mesh ($d = \sqrt[n]{N}$) | $(3n^2 + n)(d-1) - 2n \cdot \log_2 N$ | $(1/2) \cdot \log_2^2(N) + \log_2 N$ |
| $d^n$-Hypercube | $(\log_2 N)(\log_2 N + 1)/2$ | $(\log_2 N)(\log_2 N + 1)/2$ |
| $d^n$-Hypermesh | $(\log_2 N)(\log_2 N + 1)/2$ | $(\log_2 N)(\log_2 N + 1)/2$ |

*Note.* The binary hypercube is also a generalized hypercube with radix 2.

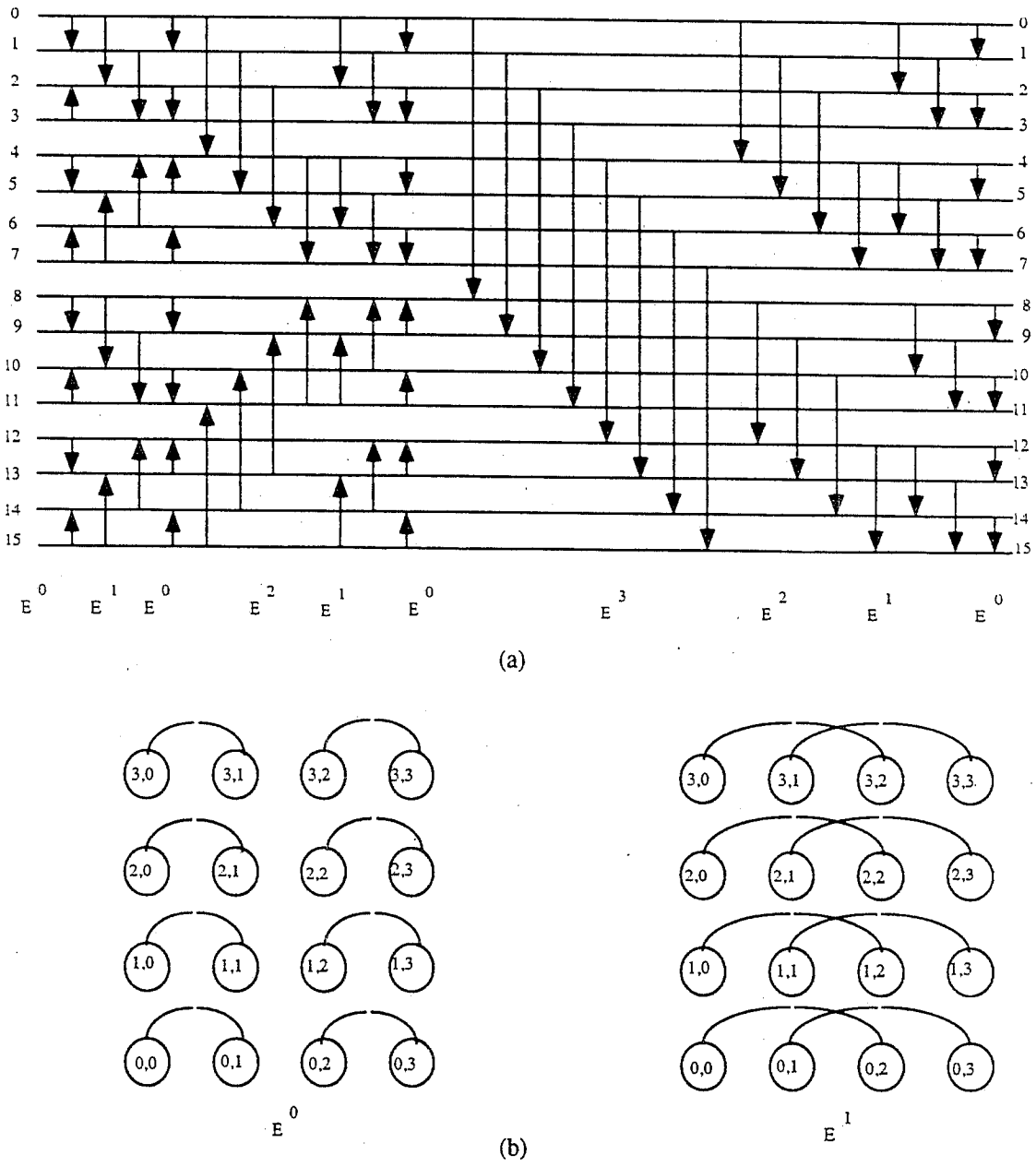



FIG. 8. (a) Data-flow graph for a 16-element bitonic sort algorithm. (b) Permutations corresponding to data transfers over hypercube dimensions.

mission delay, the propagation delay over the longest transmission line and the inherent node latency. $T_R$ varies with each network due to the equivalent aggregate bandwidth constraint. The time to perform a computation step is a parameter $T_C$, which is constant across all networks.

The time to perform sorting in a $d^n$-mesh is

$$T_{\text{sort,mesh}} = ((3 \cdot n^2 + n)(d - 1) - 2n \cdot \log_2 N) \cdot T_{\text{R(mesh)}} + ((1/2) \cdot \log_2^2(N) + \log_2 N) \cdot T_c,$$

where

$$T_{\text{R(mesh)}} = [P \cdot N \cdot 2(n + 1)/C + T_{\text{p(mesh)}} + T_{\text{n(mesh)}}],$$

where the first term in $T_{\text{R(mesh)}}$ is the transmission delay in a mesh from Table III. The time to perform sorting in a $d^n$-hypercube is

$$T_{\text{sort,hypercube}} = (\log_2 N \cdot (\log_2 N + 1)/2) \cdot T_{\text{R(hypercube)}} + (\log_2 N \cdot (\log_2 N + 1)/2) \cdot T_C,$$

where

$$T_{\text{R(hypercube)}} = [P \cdot N \cdot (n(d - 1) + 1)/C + T_{\text{P(hypercube)}} + T_{\text{n(hypercube)}}],$$

and the time to perform sorting in a $d^n$-hypermesh is

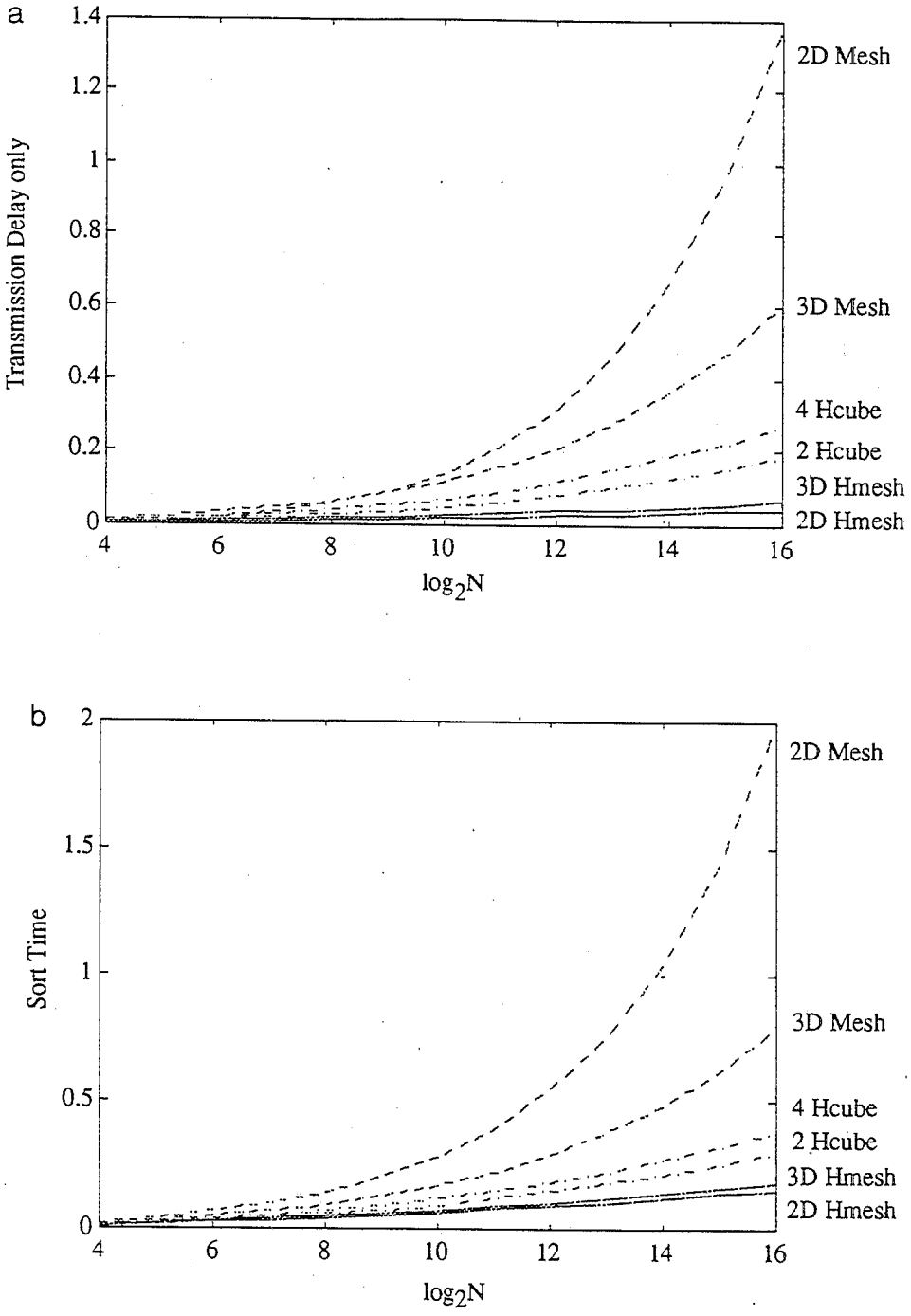



FIG. 9. Performance of parallel sorting on meshes, hypercubes, and hypermeshes Notation: "2 Hcube" denotes the binary (radix-2) hypercube; "4 Hcube" denotes the radix-4 generalized hypercube; "2D Hmesh" denotes a two-dimensional hypermesh; "3D Hmesh" denotes a three-dimensional hypermesh. (a) Lower bounds on the sort time (in $10^{-4}$s) vs $\log_2 N$. (b) Sort time (in $10^{-4}$s) vs $\log_2 N$., based on parameters typical of existing technology. For the 2D and 3D meshes the propagation delay is 0; for the hypercubes and hypermeshes, the propagation delay is 20 ns.

$$T_{\text{sort,hypermesh}} = (\log_2 N \cdot (\log_2 N + 1)/2) \cdot T_{\text{R(hypercube)}} + (\log_2 N \cdot (\log_2 N + 1)/2) \cdot T_{\text{C}}$$

$$T_{\text{R(hypermesh)}} = [P \cdot N \cdot 2n/C + T_{\text{P(hypermesh)}} + T_{\text{n(hypermesh)}}].$$

A lower bound for the sort time in these networks is shown in Fig. 9a; to determine the *lower bound*, the node latencies, propagation delays, and computation time are all set to zero. Figure 9a illustrates the total time spent just in *transmitting data* in each network. It is important to note that these lower bounds are based on the transmission delays, and are valid given typical state-of-the-art fiber-optic technology, such as the chip-set described in [28], which was funded by Darpa's Optical Computing Program, or the technology described in [26, 27]. According to these lower bounds, the hypercube is significantly

faster than the 2D mesh, and the hypermesh is significantly faster than the hypercube. The 1D lower bound also applies to the one-dimensional pipelined bus with the same aggregate bandwidth, and the 2D lower bound also applies to a 2D pipelined bus with the same aggregate bandwidth (see the Appendix).

Figure 9b illustrates the sorting time with a realistic set of parameters for hypermeshes and hypercubes: For the 2D and 3D meshes, the propagation delays are set to be 0, so that the performance of the 2D and 3D meshes is again a *lower bound* on their sort time given the other parameters. For the hypercube and hypermesh a propagation delay of 20 ns is used in every data transfer step, sufficient to allow an optical transmission over 4 m of fiber (or 12 ft.). This choice will underestimate the performance of the hypermesh implementation in Fig. 4d, since many of its transfers are over very short distances. In all networks the node latency is 10 ns and the computation time is 40 ns. It is important to note that even with the inclusion of realistic node latencies, computation times and propagation delays, the architectural comparisons in Fig. 9b are similar to those in Fig. 9a; the hypercube is significantly faster than the 2D mesh, and the hypermesh is significantly faster than the hypercube. The net conclusion from Fig. 9 is that the transmission delays dominate the performance of fiber-optic networks given realistic fiber-optic technology, such as that described in [26–28].

Figure 10a illustrates the sort time as a function of node latency, with all other parameters fixed. As the node latency increases, the performance of the 2D and 3D meshes degrades faster than the others, due to their large diameters. Figure 10b illustrates the sort time of hypermeshes and hypercubes as a function of the propagation delay per data transfer step, with all other parameters fixed. The propagation delay for 2D and 3D meshes is fixed at 0, so the sort times for the 2D and 3D meshes are *lower bounds*. As the propagation delay of the hypermeshes and hypercubes increases, their sort time increases slowly. Even with a 100 ns propagation delay per data transfer step, the hypermesh or hypercube are still faster then the meshes. Note that a propagation delay of 100 ns corresponds to optical transmissions over 66 ft. of fiber. The net conclusion from Fig. 10 is that the particular choices of propagation delays, node latencies, and computation times do not significantly affect the overall sort time, which is dominated by the transmission delays given the state-of-the-art optical technology such as that described in [26–28]. The improvements of the hypermesh over the hypercube and meshes is due to the fundamental differences in the network architecture. Furthermore, the improvements increase as the network size increases (Fig. 10 is for a moderate size of $\log_2 N = 12$).

One of the fundamental properties of any architecture is its bisection bandwidth. The improved performance of sorting is based on the higher bisection bandwidth in hypermeshes. Figure 11 illustrates the bandwidth of these networks crossing the bisector, given permutations of the form $E^y$. (See [14] for a discussion of the bisection bandwidth in hypermeshes.)

From these figures, we conclude that under the constraint of equivalent aggregate bandwidth and the constraints of existing state-of-the-art fiber-optic technology, the hypermesh is considerably faster than both the binary hypercube and the conventional mesh, especially as the network size increases. The results of this section are consistent with those in the previous section, which relied on an exact general network performance model by Kleinrock.

## 7. CONCLUSIONS

The maximum size of optical multichannel switches will be limited to relatively small sizes in the foreseeable future, due to constraints on wavelength tunability [26, 27] or ultra-high speed electronic technology [28]. Hence, to interconnect multiple microprocessors into a massively parallel machine, new optical network architectures which utilize multiple smaller optical multichannel switches will be required. A graph-theoretic model for a class of interconnection networks called "hypermeshes" was proposed, and the architectural attributes of the model were characterized. Hypermeshes are based on the concept of orthogonal "hypergraphs," with $N$ nodes arranged an $n$-dimensional space, where all nodes aligned along a dimension are members of a hypergraph hyperedge, and where hyperedges can perform $O(d)$ data transfers over their members in one step.

Hyperedges are the basic building blocks of large hypermeshes, and they can be implemented efficiently with existing optical technology. Two attractive optical implementations were proposed. The first relies on a *chip-set* funded by Darpa's Optical Computing Program [28]. The second relies on newer optical technology, namely fast tunable optical components and WDM to implement switching in the optical domain. Furthermore, the use of WDM reduces the hypermesh interconnection complexity considerably to rival that of a conventional mesh.

The hypercube is known to be one of the more powerful models of parallel computations. It is shown that the hypermesh has comparable computational power to the hypercube, while simultaneously having a lower asymptotic cost. Hypermeshes have relatively large bisection bandwidths, comparable to those of hypercubes and much larger than conventional meshes. Large bisection bandwidths minimize the time required to perform a number of important algorithms, including sorting. It is shown that in a stochastic environment, the performance of hypermeshes significantly exceeds that of meshes and hypercubes, and that when executing a specific parallel algorithm (parallel sorting), the performance of hypermeshes significantly exceeds that of meshes and hypercubes.

While architectural comparisons of meshes and hyper-

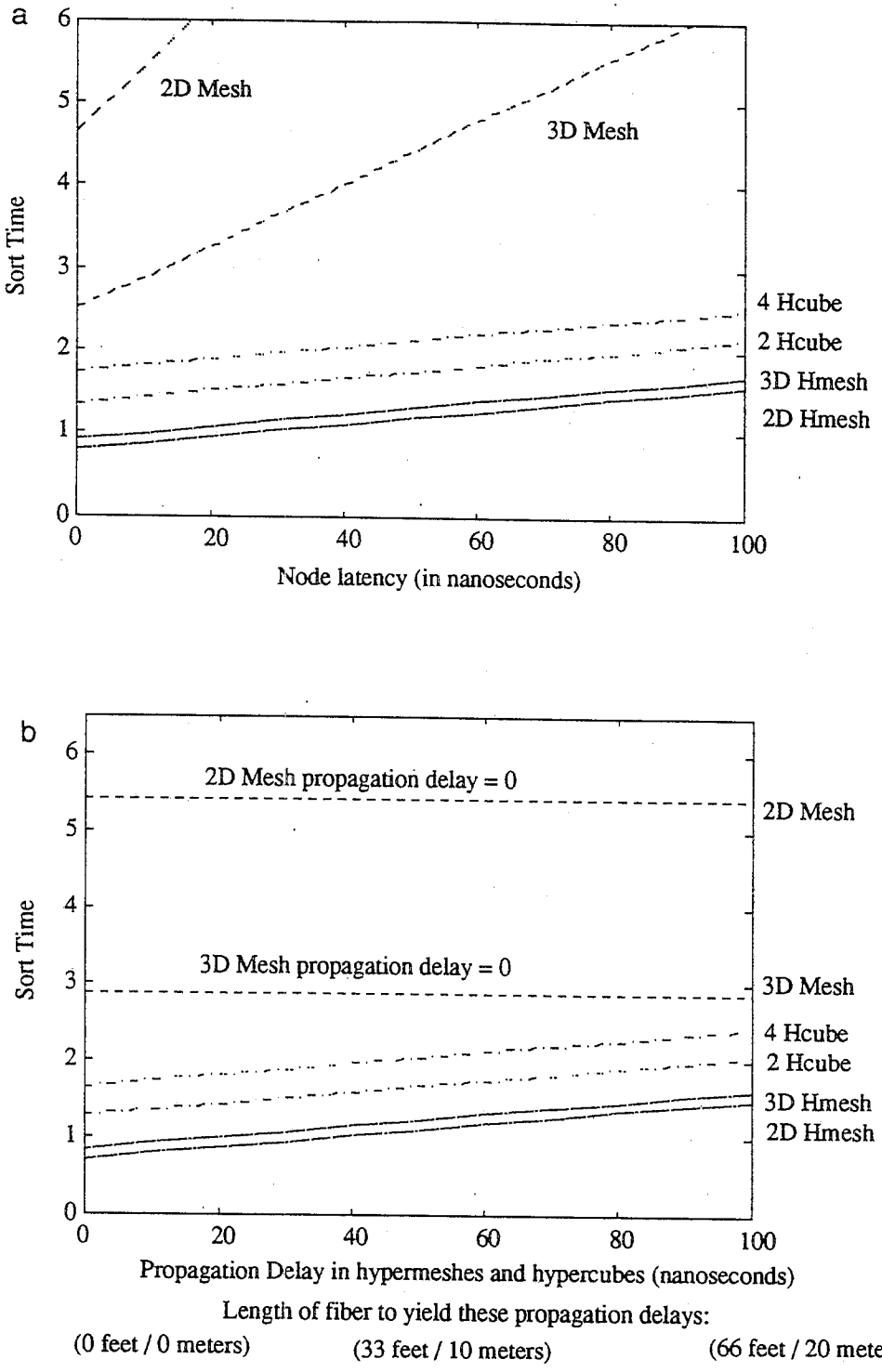



FIG. 10. Effects of node latencies and propagation delays on sort time. (a) Sort time (in $10^{-5}$ s) vs node latency, all other parameters are fixed. For the 2D and 3D meshes, the propagation delay is 0; for the hypercubes and hypermeshes, the propagation delay is 20 ns ($\log_2 N = 12$). (b) Sort time (in $10^{-5}$ s) vs propagation delay in hypermeshes and hypercubes, all other parameters fixed. For the 2D and 3D meshes, propagation delay is 0; for the hypercubes and hypermeshes, the propagation delay varies from 0 to 100 ns ($\log_2 N = 12$).

cubes in the VLSI domain have been previously performed, the impact of fiber-optics on architectures such as meshes and hypercubes, given realistic parameters, has not previous been examined. A thorough comparison of three different architectures, meshes, hypercubes, and the proposed hypermeshes, all in the fiber-optic domain, was performed. Realistic parameters representing existing fiber-optic capabilities (taken from [26–28]) were used

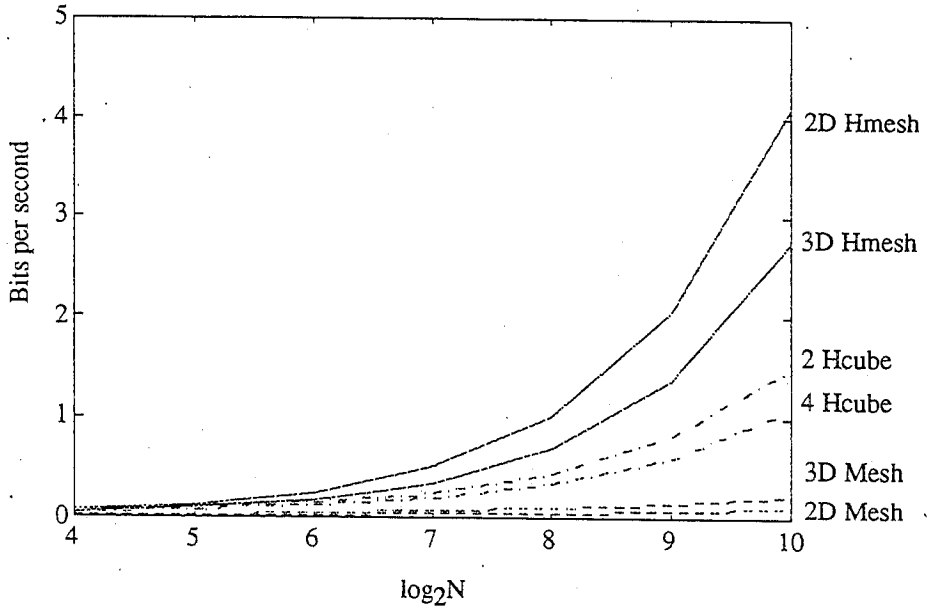



FIG. 11. Bisection bandwidth ($\times 10^{12}$) vs $\log_2 N$.

in the comparisons. The comparisons indicate that with existing fiber-optic capabilities, the hypermesh is the preferred network architecture. These comparisons were further validated with an exact general analytic model for arbitrary networks proposed by Kleinrock [30].

## APPENDIX A: DELAYS IN A PIPELINED OPTICAL BUS

A pipelined optical bus was described in [23]. In order to compare the higher dimensional hypermesh with the higher dimensional pipelined bus array, an accurate expression for the delays in a pipelined bus is derived. It is shown that an optical pipelined bus is essentially an optical implementation of a linear array which can deliver one permutation of packets in time $T \approx d \cdot (P/B)$, and that a single optical hyper-edge is essentially an optical crossbar-like switch which can deliver one permutation of data in time $T \approx (P/B) + T_p$.

An inherent capability of the optical pipelined bus is that an entire $P$-bit optical packet can reside "in flight" in the fiber between two nodes. To ensure this capability, an amount of fiber capable of dynamically buffering an entire $P$-bit packet must be inserted in *between every pair of adjacent nodes* (see [23; p. 280]). The propagation delay on the fiber between two adjacent nodes is now greater or equal to the transmission delay $T_s = P/B$, since otherwise packets will collide in the fiber. Hence, $T_p \geq T_s = (P/B)$. In this manner, all nodes can insert packets onto the optical bus simultaneously, and all packets can occupy non-overlapping sections of fiber as they travel down the fiber.

However, it now requires at least $2 \cdot T_s$ time for node $i$ to transmit a $P$-bit packet over an optical channel of bandwidth $B$ to node $(i + 1)$ or $(i - 1)$. The first $T_s$ term is transmission delay. The last bit of the packet must traverse the entire fiber-loop between the two adjacent nodes before being received, and the propagation delay is also $T_p \geq (P/B)$. Hence, the time to transfer a packet between *adjacent nodes* is now $T_R \geq 2 \cdot (P/B)$. This unit is called a "petit cycle" in [23].

From an architectural perspective, a "pipelined bus" has identical attributes to a linear array of $d$ nodes, with a packet buffer in the form of a fiber loop associated with each node. In fact, [23] claims that a pipelined bus can be built electronically by associating a $P$-bit shift-register buffer with every node, where the shift registers are connected in a *linear array*.

When executing algorithms using permutations of the form $E^i$ the performance of the 1D linear array can be *lower bounded* by choosing $T_R = (P/B)$ and $T_p = T_n = 0$. (i.e., the propagation delay and node latency are zero). This lower bound also applies to the pipelined bus, which is fundamentally a linear array. In fact, the pipelined bus must be *slower* than the 1D linear array with parameters $T_R = (P/B)$ and $T_p = T_n = 0$, since these parameters are lower bounds for all the respective delays. When executing algorithms using permutations of the form $E^i$ the performance of the 1D or 2D array processor with pipelined buses is actually *slower* than the 1D or 2D meshes with parameters $T_R = (P/B)$ and $T_p = T_N = 0$ by a factor of two, since the time taken to transfer a packet between adjacent nodes is $T_R \geq 2 \cdot (P/B)$.

EXAMPLE. The pipelined bus requires a fiber loop capable of buffering a complete packet *in between every pair of nodes*. Assume 128-bit packets ($P = 128$), a state-

of-the-art laser bandwidth of 1 Gbit/s ($B = 1 \times 10^9$; see [26–28]), and a linear array with 64 nodes. Roughly 25.6 meters of fiber is required to buffer one packet optically, and hence a fiber loop of length 25.6 meters is inserted in between every pair of adjacent nodes in the array. The time taken to transmit a packet between two adjacent nodes is now $2 \cdot (P/B) = 256$ nanoseconds (the laser is transmitting data for 128 nanoseconds, and the propagation delay over fiber between adjacent nodes is 128 nanoseconds). A pipelined bus with 64 nodes would require roughly 1,612 meters of fiber and would have a end-to-end propagation delay of about 8,064 nanoseconds, equivalent to $63 \cdot (P/B)$. The "bus cycle" duration (see [23]) is then $64 \cdot (P/B) = 8{,}192$ nanoseconds, and the system can deliver one permutation of data in every bus cycle. To compute the bisection bandwidth, we observe that the packets in a pipelined optical bus cross the bisector *sequentially*, which results in a low bisection bandwidth equivalent to that of a 2D mesh.

With this choice of parameters, an optical crossbar (or a hyper-edge) can perform a permutation of data in time $T \approx (P/B) + T_p$; assuming a 20 nanosecond propagation delay, $T \approx 128 + 20 = 148$ nanoseconds.

The fact that packets cross the bisector *sequentially* will always be a fundamental limitation of the pipelined bus; *linear arrays and pipelined busses have low bisection bandwidths*. When the transmission delay $(P/B)$ dominates the propagation delay $T_p$, which is the case for existing optical technology, the shortcomings of the pipelined optical bus implementations described in [23] can be avoided by using the optical crossbar implementations originally proposed by this author in [12] or those presented in [26, 27]. Just as a one dimensional optical crossbar outperforms a one dimensional pipelined bus, a higher dimensional hypermesh will outperform a higher dimensional pipelined bus, as illustrated in this paper.

## REFERENCES


1. Bhuyan, L. N., and Agrawal, D. P. Generalized hypercube and hyperbus structures for a computer network, *IEEE Trans. Comput.* **C-33,** 4 (Apr. 1984), 323–333.
2. Bhuyan, L. N., and Agrawal, D. P. Design and performance of generalized interconnection networks, *IEEE Trans. Comput.* **C-32,** 12 (Dec. 1983), 1081–1090.
3. Bermond, J. C., Bond, J., and Scale, J. F. Large hypergraphs of diameter 1. In B. Bollobas (Ed.). *Graph Theory and Combinatorics*. Academic Press, New York 1984.
4. Berge, C. *Graphs and Hypergraphs*. North-Holland, Amsterdam, 1973.
5. Dowd, P. Wavelength division multiple access channel hypercube processor interconnection, *IEEE Trans. Comput.* **41**(10) (Oct. 1992), pp. 1223–1241.
6. Dally, W. J., and Sietz, C. L. Deadlock free message routing in multiprocessor interconnection networks. *IEEE Trans. Comput.* **C-36,** 5 (May 1987), 547–553.
7. Lawrie, D. H. Access and alignment of data in an array processor. *IEEE Trans. Comput.* (Dec. 1975), 1145–1155.
8. Lenfant, J. Parallel permutations of data: A Benes network control algorithm for frequently used permutations. *IEEE Trans. Comput.* **C-27,** 7 (July 1978), 637–647.
9. Memmi, G., and Raillard, Y. Some new results on the (d,k) graph problem. *IEEE Trans. Comput.* **C-31,** 8 (Aug. 1982), 784–791.
10. Parker, D. S. Notes on shuffle-exchange type switching networks. *IEEE Trans. Comput.* **C-29,** 3 (Mar. 1980), 213–222.
11. Pease, M. C. The indirect binary n-cube microprocessor array. *IEEE Trans. Comput.* **C-26** (May 1977), 458–473.
12. Szymanski, T. H. A fiber-optic "hypermesh" for SIMD/MIMD machines. *1990 Supercomputing-90 Conference*, New York, Nov. 1990, pp. 710–719.
13. Szymanski, T. H. O(log N/log log N) Randomized Routing on Degree-log N Hypermeshes, *1991 Int. Conference on Parallel Processing*, Aug. 1991, pp. 443–450.
14. Szymanski, T. H. The complexity of FFT and related butterfly algorithms on meshes and hypermeshes. *1992 Int. Conference on Parallel Processing*, August 1992.
15. Wittie, L. D. Communication structures for large networks of microcomputers. *IEEE Trans. Comput.* **C-30,** 4 (Apr. 1981), 264–273.
16. Preparata, F. P., and Vuillemen, J. The cube-connected cycles: A versatile network for parallel computation. *Comm. ACM* (May 1981), 300–309.
17. Scherson, I. Orthogonal graphs for a class of interconnection networks. *IEEE Trans. Parallel Distribut. Systems* **2,** 1 (Jan. 1991), pp. 3–19.
18. Tanabe, N., Suzuoka, T., Nakamura, S., Kawakura, Y., and Oyanagi, S. Base-m n-cube: High-performance interconnection networks for highly parallel computer prodigy. *1991 Int. Conf. Parallel Processing*, August 1991, pp. 509–516.
19. Thompson, C. D., and Kung, H. T. Sorting on a mesh connected processor array. *Comm. ACM* (1972), 263–271.
20. Kumar, V. K., and Raghavendra, C. S. Array processing with multiple broadcasting. *J. Parallel Distribut. Comput.* **4** (1987), 173–190.
21. Stout, Q. F. Mesh connected computers with broadcasting. *IEEE Trans. Comput.* (1983), 826–830.
22. Dowd, P. High performance interprocessor communication through optical wavelength division multiple access channels. *1991 Symp. on Computer Architecture*, pp. 96–105.
23. Guo, Z., Melhem, R., Hall, R., Chiarulli D., and Levitan, S. Pipelined communications in optically interconnected arrays. *J. Parallel Distribut. Comput.* **12** (1991), 269–282.
24. Wailes, J., and Meyer, D. Multiple channel architecture: A new optical interconnection strategy for massively parallel computers. *J. Lightwave Tech.* **9,** 12 (Dec. 1991), 1702–1716.
25. Wang, I., and Yun, D. Y. Y. A 2D reconfigurable array for efficient parallel computing. *1st IEEE Symposium on Parallel and Distributed Processing*, Dallas, 1989, pp. 273–280.
26. Arthurs, E. Goodman, M.S. Kobrinski H. and Veechi, V.P. HYPASS: An optoelectronic hybrid package switching system. *IEEE J. on Select. Areas Comm.*, **6,** 9 (Dec. 1988), 1500–1510.
27. Arthurs, E. et. al., Multiwavelength optical crossconnect for parallel-processing computers. *Electron. Lett.* **24,** 2 (Jan. 1988), 119–120.
28. Pedrotti, et. al., 16 × 16 optical-fiber crossbar switch operating at 0.85 $\mu$m. *J. Lightwave Tech.* **8,** 9 (Sept. 1990), 1334–1342.
29. Leighton, F. T. *Parallel Algorithms and Architectures: Arrays, Trees and Hypercubes*. Morgan–Kauffman, Los Altos, CA, 1992.
30. Kleinrock, L. *Queueing Systems, Volume II*. Wiley–Interscience, New York, 1976.
31. Stone, H. Parallel processing with the perfect shuffle. *IEEE Trans. Comput.* (Feb. 1971), 153–161.
32. Mackenzie, L. M., Ould-Khaoua, M., Sutherland, R. J., Kelly, T.

Cobra: A high-performance interconnection for large multicomputers. Computing Science Research Report 1991/R19, University of Glasgow, Oct. 1991, pp. 1–21.

33. Valiant, L. G., and Brebner, G. J. Universal schemes for parallel communications. *Proc. 13th Annual ACM Symp. on Theory of Computing*, 1981, pp. 263–277.
33. Corbett, P. F., and Scherson, I. Sorting in mesh connected multiprocessors. *IEEE Trans. Parallel Distribut. Systems* **3**, 5 (Sept. 1992), 622–625.
34. Abraham, S., and Padhmanabhan, K. Performance of multicomputer networks under pin-out constraints. *J. Parallel Distribut. Comput.* **12** (1992), 237–248.
35. Szymanski, T. Graph-theoretic models for photonic networks. In I. Scherson (Ed.). *Proceedings of the New Frontiers: A Workshop on Future Directions of Massively Parallel Processing*. IEEE Computer Society Press, Los Alamitas, CA, 1993, pp. 85–96.

Received April 22, 1992; revised May 5, 1993; accepted November 24, 1993

36. Li, Y., Rao, S., Redmond, I., Wang, T., and Lohmann, A., Free space WDMA optical interconnects using mesh connected bus topology, *Int. Conf. Optical Computing*, 1994, pp WA2, 191–192.

---

TED SZYMANSKI received the Ph.D. degree in electrical engineering from the University of Toronto in 1988. During 1987–1991, he was a principal investigator at the NSF Center for Telecommunications Research at Columbia University, performing research in ATM switching and WDM optical architectures. Currently he is a project leader in the Canadian Institute for Telecommunications Research at McGill University, leading a project entitled "Large ATM Architectures based on Terabit Photonic Backplanes." A multichannel free-space photonic backplane which incorporates many of the architectural concepts described in this paper is currently under construction at McGill University.