An Adaptive and Fault Tolerant Wormhole Routing Strategy for k-ary n-cubes

Daniel H. Linder, *Member, IEEE,* and Jim C. Harden, *Member, IEEE*

***Abstract*—The concept of virtual channels is extended to multiple, virtual communication systems that provide adaptability and fault tolerance in addition to being deadlock-free. A channel dependency graph is taken as the definition of what connections are possible and any routing function must use only those connections defined by it. Virtual interconnection networks allowing adaptive, deadlock-free routing are examined for three k-ary n-cube topologies: unidirectional, torus-connected bidirectional, and mesh-connected bidirectional.**

***Index Terms*—Adaptive routing, concurrent computing, deadlock, fault-tolerant computing, message passing architecture, virtual channels, virtual networks, wormhole routing.**

I. Introduction

THE interest in highly concurrent computers has led to much research in message passing architectures. Since shared memory is not used for communication in these architectures, the potential exists for scaling them to tens of thousands of processors [1], [2]. Instead of shared memory, all interprocessor communication occurs over high performance data channels linking the processors or *nodes.* Each node is a complete computer with its own local memory. Ideally, the communication system should allow messages to be quickly sent between any pair of nodes. The closer the system is to this ideal, the less cognizant the software need be of the actual topology of the channels. A direct channel between each pair of nodes would permit the fastest communication, but the cost would be prohibitive for large computers. A compromise topology is generally chosen with channels only between certain pairs of nodes. However, messages are then forced to travel through intermediate nodes to reach their destination. This concession can lead to a condition called deadlock.

Intuitively, deadlock occurs when messages traveling in the communication system develop *dependency loops* among themselves that prevent further movement. The possibility of deadlock is introduced when a message must travel through intermediate nodes because some portion of the message must always be stored at these nodes. Since this storage is finite, it may become filled. When this happens, new messages cannot enter the node until old messages start to leave.

Store-and-forward and *wormhole* routing are the two common message routing methods, and each uses different approaches to eliminate deadlock [3]–[6]. In store-and-forward routing, a message is treated as a packet that is transferred as a whole between each node along its path. Deadlock can be prevented if a pool of buffers is provided that is properly structured, but the number of buffers necessary to maintain this structure tends to grow with the size of the network. This is not desirable for a computer with thousands of nodes.

With wormhole routing, a message is decomposed into words or *flits* (flow control digits), and the flits of one message may be spread out among several nodes as the message moves. The conceptual model for the communications portion of a node is a set of queues—one for each input channel. Instead of one channel dumping a message into a buffer and that buffer being emptied into another channel, the queue of an input channel is connected to an output channel, and the message flits flow through the connection until the entire message has passed. Then the connection can be broken and other connections made. Deadlock can be prevented by restricting the combinations in which the input and output channels are connected. Only wormhole routing is examined in this paper since the nodal storage requirements (i.e., number of queues) grows with the number of input and output channels and not the total number of nodes in the system. Also, if the message traffic is not heavy, the *message latency* (time spent in transit) is less due to the inherent pipelining effect when the flits are spread over several nodes.

Restricting the input channel connections can prevent deadlock for wormhole routing, but this has the undesirable effect of restricting the paths that a message can take between a particular pair of nodes. If a communication system can send messages along alternate paths between the same pair of nodes, it is called *adaptable.* An adaptable system has two distinct advantages: 1) if one path is crowded with message traffic, another path can be taken to reduce message latency, and 2) if one path has a faulty node, another path can be taken to preserve communication. Previous wormhole routing schemes have restricted message traffic to a single path between any pair of nodes [6]. The purpose of this paper is to present simple, deadlock-free, adaptable, and fault-tolerant wormhole routing techniques for a popular topology—the k-ary n-cube.

The k-ary n-cube topology has been used in several significant computers like the Cosmic Cube and the Connection Machine [2], [8], and has proven to be useful for general purpose processing. High radix (high k), low dimensional (low n) cubes are especially easy to construct and scale, and they exhibit low message latencies for important physical field simulations like CFD (computational fluid dynamics) [9]. There are three basic types of k-ary n-cube: 1) unidirectional, 2) torus-connected bidirectional, and 3) mesh-connected bidirectional. Examples of all three forms will be shown later as each is studied.

II. Virtual Communication Systems for k-ary n-cubes

As discussed earlier, the wormhole routing technique associates a queue with each communication channel. Let a physical channel be the actual interconnect between a pair of nodes. If only one queue is used for each physical channel, a unidirectional

Manuscript received May 16, 1988; revised February 15, 1989. This work was supported by the Defense Advanced Research Projects Agency under Contract DAAA15-86-K-0025, monitored by the Department of the Army. Additional support was provided by grants from Sun Microsystems, Inc.

The authors are with the Department of Electrical Engineering, Mississippi State University, Mississippi State, MS 39762.

IEEE Log Number 9040678.

k-ary n-cube will not be deadlock-free regardless of how the input to output channel connections are restricted at each node. Although a mesh-connected k-ary n-cube can be made deadlock-free using only one queue for each physical channel, messages are constrained to travel through each dimension in a defined order. This allows for only one message path between each pair of nodes and provides no adaptability. To free the communication system from the constraints of a fixed physical topology, the concept of a *virtual network* is introduced.

Previous investigators have shown how *virtual channels* can be used to avoid deadlock in unidirectional k-ary n-cubes as well as a variety of other topologies [6]. Virtual channels are logical abstractions that share the same physical channel. Even though they are time multiplexed over a single physical channel, a separate queue must be maintained in the node for each virtual channel. This concept is extended here to multiple, virtual communication systems that provide adaptability and fault tolerance in addition to being deadlock-free.

Fig. 1 illustrates the concept of virtual communication systems underlying a single physical communication system. There is a mapping from the virtual nodes and channels to the physical nodes and channels although it is not necessarily one-to-one. The only restriction on the mapping is that nodes adjacent in the virtual systems map to nodes adjacent in the physical system. All virtual channels that map to a single physical channel are time multiplexed over the physical channel. The remainder of this paper will focus on describing various virtual communication systems for k-ary n-cubes. Some definitions are required, though, before precise descriptions can be given (several definitions are based closely on the work in [6]).

Definition 1: A *physical interconnection network,* PI, is a strongly connected directed graph, $\mathrm{PI} = \mathrm{PG}(\mathrm{PN}, \mathrm{PC})$. PN is a set of nodes representing processors, and PC is a set of edges representing actual physical channels connecting the processors. Let pn_i and pc_i be the ith node and channel, respectively. Also, the source node of pc_i is ps_i and the destination node is pd_i. Later the simple i subscripts will be expanded into several integers to indicate the relationships between nodes and channels.

Definition 2: A *virtual interconnection network,* VI_i, is a directed graph, $\mathrm{VI}_i = \mathrm{VG}_i(\mathrm{VN}_i, \mathrm{VC}_i)$. The subscript i is used because a single physical interconnection network could have several virtual interconnection networks mapped to it. VN_i is a set of virtual nodes that are mapped to PN by the function nmap: $\mathrm{VN}_i \rightarrow \mathrm{PN}$. VC_i is a set of edges representing virtual channels that are mapped to PC by the function cmap: $\mathrm{VC}_i \rightarrow \mathrm{PC}$. Let vc_{ij} be the jth virtual channel in the ith virtual interconnection network. Associated with each virtual channel, vc_{ij}, is a queue vq_{ij}. Also, the source node of vc_{ij} is vs_{ij} and the destination node is vd_{ij}. Later the simple i and j subscripts will be expanded into several integers to indicate the relationships between nodes and channels.

Definition 3: A *connection* exists from channel vc_{ij} to vc_{ik} if the flit at the head of vq_{ij} will be transferred to the tail of vq_{ik} when the space in vq_{ik} becomes available.

Definition 4: A *channel dependency graph,* D_i, for the virtual interconnection network VI_i is a graph $D_i = G_i(\mathrm{VC}_i, E_i)$ where $E_i = \{(\mathrm{vc}_{ij}, \mathrm{vc}_{ik}) \mid \text{a connection can exist from } \mathrm{vc}_{ij} \text{ to } \mathrm{vc}_{ik}\}$. Thus, D_i is a graph with a node for every channel in VI_i and edges to define what connections can be made between the channels. A channel is not allowed to connect to itself, so there will be no 1-cycles in D_i.

Definition 5: A *routing function* $R_i : \mathrm{VC}_{ij} \times \{\text{all possible } \textit{states} \text{ of } \mathrm{VI}_i\} \rightarrow \mathrm{VC}_{ik}$ for a virtual interconnection network VI_i maps a channel vc_{ij} and the present state of VI_i to another channel vc_{ik}. When the first flits of a message arrive at the head of some channel queue, vq_{ij}, the routing function determines the channel, vc_{ik}, that vc_{ij} should be connected to so that the flits in vq_{ij} can advance (possibly indirectly) toward their destination. The state of the network can encompass varying quantities of information about the network depending on how "smart" the routing should be. Simple routing may only care about the destination of the message in vq_{ij} while more complex routing may need to know the entire configuration of VI_i or even how long a message has traveled in the network and what its path was. Note that the definition of the routing function could be relaxed some to allow it to specify a set of channels where the channel finally taken would be the first one to become free. However, from the standpoint of deadlock, we must consider the worst case where a message can only go to a particular channel next.

Definitions 4 and 5 represent a subtle difference between the treatment given in this paper and earlier research. Previously, a routing function was defined for a virtual interconnection network, and then a channel dependency graph was derived from this routing function by using it as an implicit definition of all possible channel connections. In contrast, we take the channel dependency graph as the definition of what connections are possible, and any routing functions developed for the network are restricted to use only those connections. This approach highlights a separation between the structure of the virtual interconnection networks and the adaptive routing algorithms that indicate how messages will travel. Thus, instead of showing each new routing function is deadlock-free, the virtual interconnection network is shown to be deadlock-free for a given channel dependency graph, and then new routing functions are shown to obey the restrictions of the channel dependency graph. To illustrate these concepts, an analogy can be drawn between virtual interconnection networks and traffic systems. The virtual interconnection network is a map that tells what channels (roads) exist, and the channel dependency graph tells what connections (turns) are legal. Then, if all messages (cars) obey these restrictions and the restrictions were properly developed, no deadlock (accidents) should occur. A routing function (the driver) just guides a message from node A to node B while obeying all restrictions.

Deadlock results from flits waiting for other flits to move. If one flit is forced to wait on another to move, its motion is dependent on the motion of the other flit. That is why D_i, which specifies channel connections, is called a channel *dependency* graph. If a connection is possible from vc_{ij} to vc_{ik}, then the motion of flits in vc_{ij} may be dependent on the motion of flits in vc_{ik}. It is important that once VI_i is proven to be deadlock-free based on D_i, no other additional dependencies are introduced. These could result from the multiplexing of several virtual channels over a single physical channel. If queue space becomes available in channel vc_{ik} where $\mathrm{vc}_{ik} = R_i(\mathrm{vc}_{ij}, \text{ state of } \mathrm{VI}_i)$, then a flit at the head of vq_{ij} must be allowed to cross the physical channel $\mathrm{cmap}(\mathrm{vc}_{ik})$ within a finite time regardless of the flit traffic on any other virtual channel mapped to $\mathrm{cmap}(\mathrm{vc}_{ik})$. It is permissible to slow flits down while other flits finish transferring, but making a flit wait until other flits on different virtual networks have found queue space will create an additional dependency that may produce deadlock.

It was shown in [6] that deadlock is impossible if the channel dependency graph is acyclic (of course, in [6] it was proven that a *routing function* was deadlock free and not a *virtual network* but the same proof would apply). This result will be used in the

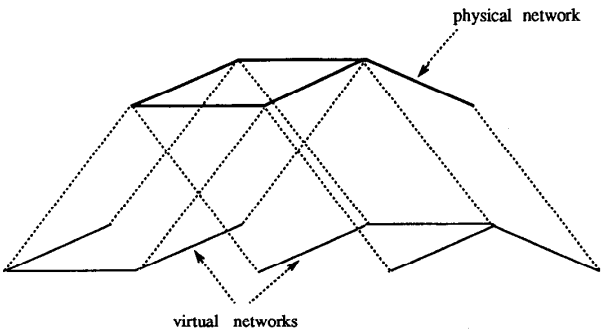


Fig. 1. Multiple virtual networks underlying a single physical network.

following sections to show that the proposed virtual networks are deadlock-free.

Before discussing the details of different interconnection networks, it is necessary to define two convenience functions. First, mod_k(expression) is equal to the value of the expression calculated with modulo k arithmetic. Second, $\mathrm{dig}(N, d)$ is equal to the dth digit of N. For example, $\mathrm{mod}_4(2+3) = 1$, $\mathrm{mod}_4(2-3) = 3$, $\mathrm{dig}(567, 0) = 7$, and $\mathrm{dig}(567, 2) = 5$.

A. Unidirectional k-ary n-cubes

There is only one channel between adjacent nodes in a unidirectional k-ary n-cube. The physical interconnection network for a 4-ary 2-cube is shown in Fig. 2. The physical nodes are identified by a label pn_N (i has been replaced with a single integer N) where N is the *node position.* It is an n digit, radix k integer that indicates the node's position in each dimension, i.e., $\mathrm{dig}(N, d)$ is equal to the position of the node in dimension d.

The physical channels are identified by a label $\mathrm{pc}_{\mathrm{SC,DIM}}$ (i has been expanded into two integers) where SC is the node position N of the *physical source node of the channel,* i.e., the node where flits enter the channel. DIM is the *channel dimension.* This tells what dimension the channel lies in as it exits the node. That is, the node positions of nodes $\mathrm{ps}_{\mathrm{SC,DIM}}$ and $\mathrm{pd}_{\mathrm{SC,DIM}}$ differ in digit DIM. DIM is an integer with range 0 to $n-1$. The destination node of $\mathrm{pc}_{\mathrm{SC,DIM}}$ is $\mathrm{pd}_{\mathrm{SC,DIM}} = \mathrm{pn}_N$ where $\mathrm{dig}(N, \mathrm{DIM}) = \mathrm{mod}_k(\mathrm{dig}(\mathrm{SC}, \mathrm{DIM}) - 1)$ and $\mathrm{dig}(N, d) = \mathrm{dig}(\mathrm{SC}, d)$ for $d \neq \mathrm{DIM}$.

The end-around channels can lead to dependency loops and deadlock if only one queue is used per physical channel and message movement is unrestricted. This section describes a virtual interconnection network, UVI, for the unidirectional k-ary n-cube that is not only deadlock-free but also adaptable (UVI is substituted for the generic VI since several virtual interconnection networks are described in this paper).

The precise description of the virtual interconnection network has two components. First, the virtual topology will be defined by identifying the virtual nodes and channels that make up the network and specifying which nodes are the sources and destinations of each channel. For the unidirectional k-ary n-cube topology there will be only one virtual interconnection network mapped to the physical network so the i subscript used in the definitions of virtual objects will be dropped in this case. Second, the edges of the channel dependency graph will be listed to specify the connections that routing functions may use.

For any unidirectional k-ary n-cube, the virtual nodes in the single virtual interconnection network are identified by a label $\mathrm{vn}_{N,L}$ (i has been dropped, and j has been expanded into two integers) where L is the *level number.* To eliminate deadlock due to the end-around connections, the physical network is split into multiple levels. Each time a flit travels through an end-around connection, it starts traveling in a new level (i.e., L is reduced by 1). L is an integer with a minimum range of 0 to n (this range is discussed shortly). Periodically moving to different sets of storage structures is reminiscent of previous store-and-forward buffer structuring methods [3]–[5]. A message always starts in a level $\geq m$ where m is the number of end-around channels it will move through. Each virtual node is mapped to the physical node with the same node position, i.e., $\mathrm{nmap}(\mathrm{vn}_{N,L}) = \mathrm{pn}_N$.

The virtual channels are identified by a label $\mathrm{vc}_{\mathrm{SC,DIM},L}$ (i has been dropped, and j has been expanded into three integers) where SC is the node position N of the *virtual source node of the channel,* i.e., the node where flits enter the channel. DIM is the *channel dimension.* This tells what dimension the channel lies in as it exits the node. That is, the node positions of nodes $\mathrm{vs}_{\mathrm{SC,DIM},L}$ and $\mathrm{vd}_{\mathrm{SC,DIM},L}$ differ in digit DIM. DIM is an integer with range 0 to $n-1$. The destination node of $\mathrm{vc}_{\mathrm{SC,DIM},L}$ is

$$\mathrm{vd}_{\mathrm{SC,DIM},L} = \begin{cases} \mathrm{vn}_{N,L} & \text{if } \mathrm{vc}_{\mathrm{SC,DIM},L} \text{ is not an end-around channel} \\ \mathrm{vn}_{N,L-1} & \text{if } \mathrm{vc}_{\mathrm{SC,DIM},L} \text{ is an end-around channel} \end{cases}$$

where $\mathrm{dig}(N, \mathrm{DIM}) = \mathrm{mod}_k(\mathrm{dig}(\mathrm{SC}, \mathrm{DIM}) - 1)$ and $\mathrm{dig}(N, d) = \mathrm{dig}(\mathrm{SC}, d)$ for $d \neq \mathrm{DIM}$. Note that $\mathrm{vc}_{\mathrm{SC,DIM},L}$ is an end-around channel if $\mathrm{dig}(\mathrm{SC}, \mathrm{DIM}) = 0$. Also, each virtual channel is mapped to the physical channel that is between the same nodes, i.e., $\mathrm{cmap}(\mathrm{vc}_{\mathrm{SC,DIM},L}) = \mathrm{pc}_{\mathrm{SC,DIM}}$.

Fig. 3 shows the virtual interconnection network for a 4-ary 2-cube. The number of levels depends on how many times a message might need to travel through end-around channels for a given routing function. The minimum number of levels for a message to be able to travel between any pair of nodes is $n+1$. For example, a message traveling from pn_{00} to pn_{11} travels through two end-around channels and thus, needs two extra levels in addition to the starting level.

The set of edges E in the channel dependency graph D is

$$E = \{(\mathrm{vc}_{\mathrm{SC1,DIM1},L1}, \mathrm{vc}_{\mathrm{SC2,DIM2},L2}) \mid \mathrm{vd}_{\mathrm{SC1,DIM1},L1} = \mathrm{vs}_{\mathrm{SC2,DIM2},L2}\}.$$

That is, any channel with destination virtual node A can be connected to any channel with source virtual node A.

Note that the definition of the channel dependency graph is very simple only because the previous discussion has developed the concept of virtual nodes. Theoretically, virtual nodes could

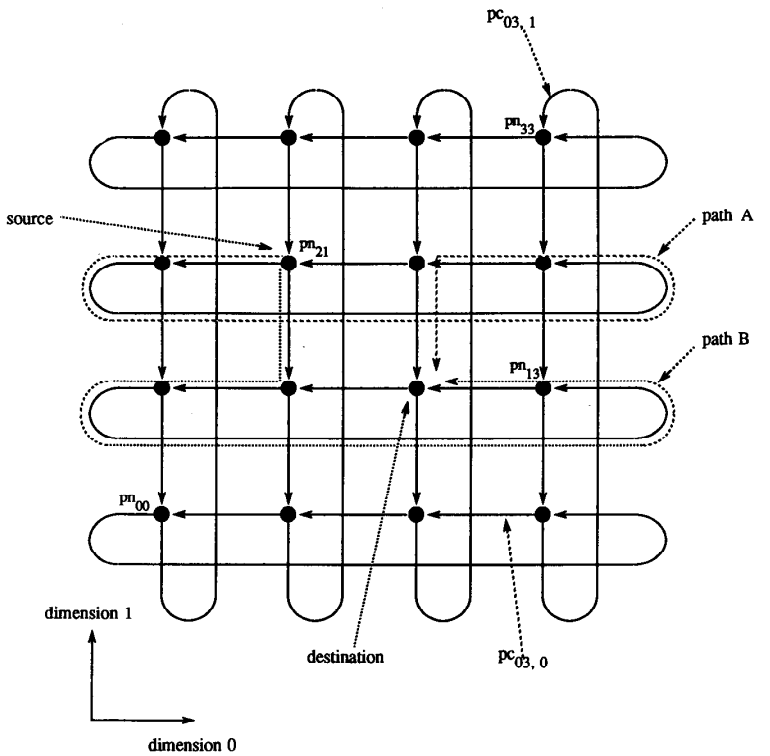


Fig. 2. Physical interconnection network for a unidirectional 4-ary 2-cube.

be eliminated leaving only virtual channels. The virtual channels would be defined by giving them distinct labels and mapping them to appropriate physical channels. Then the channel dependency graph would have the expanded task of defining the possible connections for this mass of channels so that the levels would be implied. We believe, however, that introducing virtual nodes as an intermediate step allows easier visualization of message traffic and thus aids the understanding and development of virtual topologies. It is the channel dependency graph, however, that gives the virtual nodes meaning and forces the additional constraints needed for the more complex systems to follow.

Before a virtual interconnection network with its associated channel dependency graph can be of any use, it must be shown to be deadlock-free.

Assertion 1: The virtual interconnection network, UVI, with its associated channel dependency graph D defined for a unidirectional k-ary n-cube is deadlock-free.

Proof: To show that UVI is deadlock-free, it must be proven that D has no cycles. By definition, D has no 1-cycles. It can be shown that if a number x can be assigned to each node in a graph such that an edge from node A to node B implies $x_A > x_B$, then the graph is acyclic [10]. Assign the number

$$x_{\mathrm{SC,DIM},L} = (L \times k^n) + \sum_{i=0}^{n-1} [\mathrm{dig}(\mathrm{SC}, i) \times k^i]$$

to virtual channel $\mathrm{vc}_{\mathrm{SC,DIM},L}$. Now if $(\mathrm{vc}_{\mathrm{SC1,DIM1},L1}, \mathrm{vc}_{\mathrm{SC2,DIM2},L2})$ is an element of E, then there are two possible cases for the numbers assigned to these channels.

1) If $\mathrm{vc}_{\mathrm{SC1,DIM1},L1}$ is not an end-around channel, then $L2 = L1$ and $\mathrm{dig}(\mathrm{SC2}, \mathrm{DIM1}) = \mathrm{dig}(\mathrm{SC1}, \mathrm{DIM1}) - 1$. Thus, $x_{\mathrm{SC1,DIM1},L1} > x_{\mathrm{SC2,DIM2},L2}$ since the summation is reduced by k^{DIM1}.
2) If $\mathrm{vc}_{\mathrm{SC1,DIM1},L1}$ is an end-around channel, then $L2 = L1 - 1$ and $\mathrm{dig}(\mathrm{SC2}, \mathrm{DIM1}) = k - 1$. Thus, $x_{\mathrm{SC1,DIM1},L1} > x_{\mathrm{SC2,DIM2},L2}$ since $x_{\mathrm{SC2,DIM2},L2}$ is increased by $(k-1)^{\mathrm{DIM1}}$ but it is decreased by $k^n > (k-1)^n \geq (k-1)^{\mathrm{DIM1}}$.

Either case yields $x_{\mathrm{SC1,DIM1},L1} > x_{\mathrm{SC2,DIM2},L2}$ so D is acyclic, and thus, UVI is deadlock-free. □

The key feature of UVI is the adaptability it provides for the physical network. Previous schemes have forced messages to move through the dimensions in a fixed order. Since, in UVI, a flit can exit a virtual node in any dimension, a message can move through dimensions at will. Fig. 2 shows a message taking two different paths, A and B, in the physical interconnection network. Fig. 3 illustrates this motion in UVI. These two paths are called *shortest paths* since they use the least number of channels in moving between the two nodes. UVI allows messages to take *any* shortest path between two nodes. However, a message can also take other paths as long as the number of end-around channels it travels through does not exceed the number of levels in UVI minus one. If very convoluted paths are desirable, then more levels can be added.

B. Torus-Connected Bidirectional k-ary n-cubes

There are two channels between adjacent nodes in a bidirectional k-ary n-cube—one for each direction. A bidirectional cube can take advantage of locality of communication. If two adjacent nodes need to exchange messages, the path is only one channel

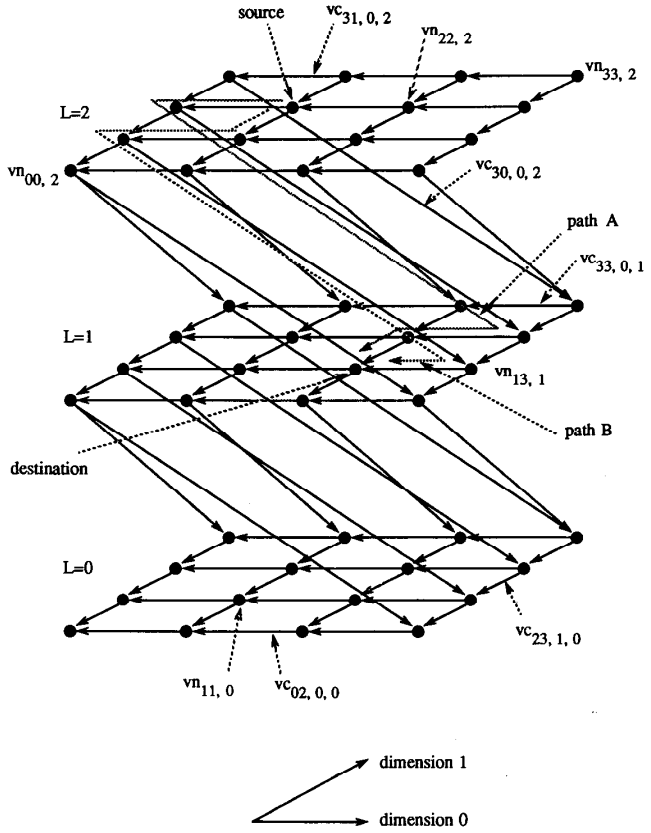


Fig. 3. Virtual interconnection network, UVI, for a unidirectional 4-ary 2-cube.

long for both messages. In a unidirectional cube, however, one path would be $k-1$ channels long. For bidirectional cubes, the end-around channels are not essential. If they are present, the cube is called torus connected, and if not, it is called mesh connected. It will be shown in Section II-C that eliminating the end-around channels greatly simplifies the virtual interconnection networks. The physical interconnection network for a torus-connected 4-ary 2-cube is shown in Fig. 4. The physical nodes are identified by a label pn_N (i has been replaced with a single integer N) where N is the same *node position* defined earlier.

The physical channels are identified by a label $\mathrm{pc}_{\mathrm{SC,DIM,DIR}}$ (i has been expanded into three integers) where DIR is the *channel direction.* DIR is 0 if the channel is in the direction of decreasing subscripts and 1 if the channel is in the direction of the increasing subscripts. The next equation states this more precisely. The destination node of $\mathrm{pc}_{\mathrm{SC,DIM,DIR}}$ is $\mathrm{pd}_{\mathrm{SC,DIM,DIR}} = \mathrm{pn}_N$ where

$$\mathrm{dig}(N,\mathrm{DIM}) = \begin{cases} \mathrm{mod}_k(\mathrm{dig}(\mathrm{SC},\mathrm{DIM})-1) & \text{if DIR} = 0 \\ \mathrm{mod}_k(\mathrm{dig}(\mathrm{SC},\mathrm{DIM})+1) & \text{if DIR} = 1 \end{cases} \tag{1}$$

and $\mathrm{dig}(N,d) = \mathrm{dig}(\mathrm{SC},d)$ for $d \neq \mathrm{DIM}$.

For this topology, there will be several virtual interconnection networks mapped to the physical network. Thus, the i subscript used in the definitions of virtual objects will not be dropped. Instead, i will be represented by an $n-1$ digit, binary integer VN which is called the *virtual network* of the object. VN is defined as

$$\mathrm{dig}(\mathrm{VN},d) = \begin{cases} 0 & \text{if the channels in dimension } d+1 \text{ of this network point exclusively in the direction of decreasing subscripts} \\ 1 & \text{if the channels in dimension } d+1 \text{ of this network point exclusively in the direction of increasing subscripts.} \end{cases}$$

That is, there will be 2^{n-1} virtual networks for a torus-connected k-ary n-cube identified by $\mathrm{TVI}_{\mathrm{VN}}$ and each will have different restrictions on the directions of its channels. Levels will also be used here because the end-around channels are still present, but the bidirectional nature of the physical network presents a new source of the dependency loops—multidimensional loops. To eliminate this problem, we will split the bidirectional network into several virtual networks that are very similar to the unidirectional networks examined in Section II-A. Each network will have channels in only one direction in $n-1$ dimensions and both directions in the 0th dimension. Allowing the 0th dimension to be bidirectional while the rest are unidirectional will complicate our analysis, but it does result in reducing the number of virtual networks by half.

A message will use a particular network depending on where it wishes to go. For example, if the source node of a message traveling in a 5-ary 3-cube were at position 114 and the destination node were at position 341, the message would use the virtual network identified with VN = 10. That is, the message would want to move in the direction of increasing subscripts

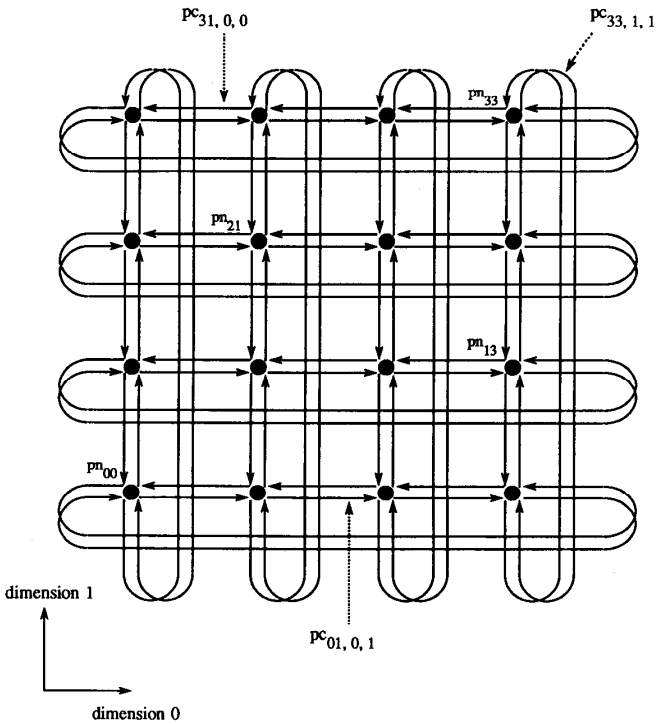


Fig. 4. Physical interconnection network for a torus-connected bidirectional 4-ary 2-cube.

(i.e., $1 \rightarrow 2 \rightarrow 3$) for dimension 2 (hence, $\mathrm{dig}(\mathrm{VN}, 1) = 1$ and the direction of decreasing subscripts (i.e., $1 \rightarrow 0 \rightarrow 4$) for dimension 1 (hence, $\mathrm{dig}(\mathrm{VN}, 0) = 0$.

For any torus-connected bidirectional k-ary n-cube, the virtual nodes in the virtual interconnection network $\mathrm{TVN}_{\mathrm{VN}}$ are identified by a label $\mathrm{vn}_{\mathrm{VN,N},L}$ (j has been expanded into two integers) where VN is the *virtual network* described above. Each virtual node is mapped to the physical node with the same node position, i.e., $\mathrm{nmap}(\mathrm{vn}_{\mathrm{VN,N},L}) = \mathrm{pn}_N$.

The virtual channels are identified by a label $\mathrm{vc}_{\mathrm{VN,SC,DIM},L,\mathrm{DIR}}$ (j has been expanded into four integers) where DIR is the *channel direction.* DIR is defined as

$$\mathrm{DIR} = \begin{cases} 0 & \text{if DIM} \neq 0 \\ 0 & \text{if DIM} = 0 \text{ and the channel points in the direction of decreasing subscripts} \\ 1 & \text{if DIM} = 0 \text{ and the channel points in the direction of increasing subscripts.} \end{cases} \quad (2)$$

Thus, DIR has no meaning outside dimension 0 since these dimensions are unidirectional. In dimension 0, however, there are two channels originating at each node and they point in opposite directions. The destination node of $\mathrm{vc}_{\mathrm{VN,SC,DIM},L,\mathrm{DIR}}$ is

$$\mathrm{vd}_{\mathrm{VN,SC,DIM},L,\mathrm{DIR}} = \begin{cases} \mathrm{vn}_{\mathrm{VN},N,L} & \text{if } \mathrm{vc}_{\mathrm{VN,SC,DIM},L,\mathrm{DIR}} \text{ is not an end-around channel} \\ \mathrm{vn}_{\mathrm{VN,N},L-1} & \text{if } \mathrm{vc}_{\mathrm{VN,SC,DIM},L,\mathrm{DIR}} \text{ is an end-around channel} \end{cases}$$

where

$$\mathrm{dig}(N, \mathrm{DIM}) = \begin{cases} \mathrm{mod}_k(\mathrm{dig}(\mathrm{SC}, \mathrm{DIM}) - 1) & \text{if } (\mathrm{DIM} \neq 0 \wedge \mathrm{dig}(\mathrm{VN}, \mathrm{DIM}-1) = 0) \vee (\mathrm{DIM} = 0 \wedge \mathrm{DIR} = 0) \\ \mathrm{mod}_k(\mathrm{dig}(\mathrm{SC}, \mathrm{DIM}) + 1) & \text{if } (\mathrm{DIM} \neq 0 \wedge \mathrm{dig}(\mathrm{VN}, \mathrm{DIM}-1) = 1) \vee (\mathrm{DIM} = 0 \wedge \mathrm{DIR} = 1) \end{cases} \quad (3)$$

and $\mathrm{dig}(N, d) = \mathrm{dig}(\mathrm{SC}, d)$ for $d \neq \mathrm{DIM}$. Note that $\mathrm{vc}_{\mathrm{VN,SC,DIM},L,\mathrm{DIR}}$ is an end-around channel if it points in the direction of increasing subscripts and $\mathrm{dig}(\mathrm{SC}, \mathrm{DIM}) = k - 1$ or it points in the direction of decreasing subscripts and $\mathrm{dig}(\mathrm{SC}, \mathrm{DIM}) = 0$. Also, each virtual channel is mapped to the physical channel that is between the same nodes, i.e.,

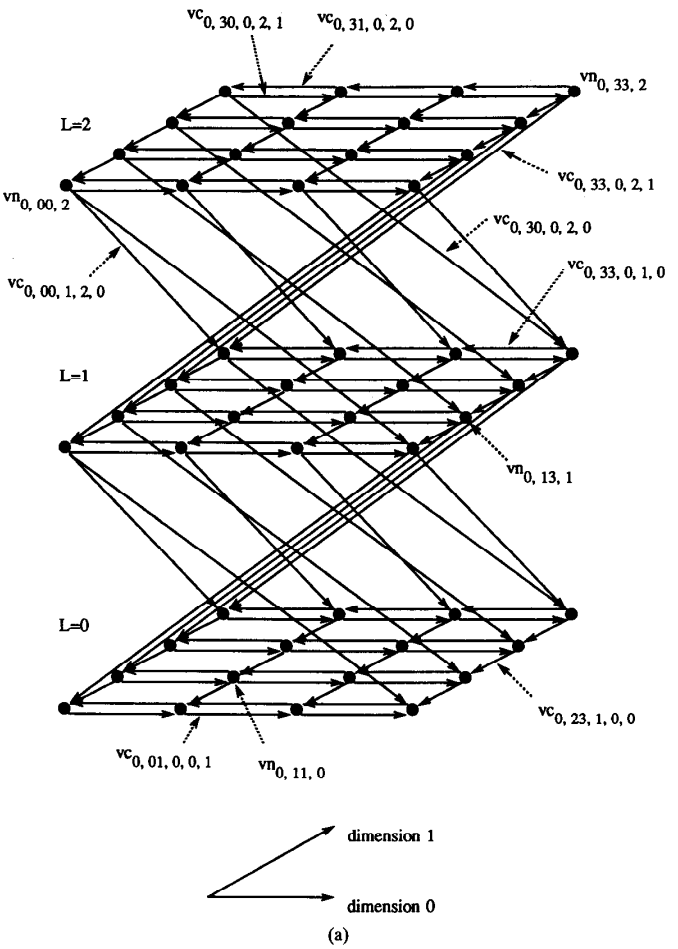
$$\mathrm{cmap}(\mathrm{vc}_{\mathrm{VN,SC,DIM},L,\mathrm{DIR}}) = \begin{cases} \mathrm{pc}_{\mathrm{SC,DIM,DIR}} & \text{if DIM} = 0 \\ \mathrm{pc}_{\mathrm{SC,DIM,dig(VN,DIM-1)}} & \text{if DIM} \neq 0. \end{cases} \quad (4)$$

Note that these equations show that the virtual interconnection networks, $\mathrm{TVI}_{\mathrm{VI}}$, are completely separate, i.e., have no channels in common. Thus, they can be treated separately for deadlock analysis.

Fig. 5 shows the two virtual interconnection networks, TVI_1 and TVI_0, for a 4-ary 2-cube. Again, the number of levels depends on how many times a message might need to travel through end-around channels for a given routing function (the minimum number of levels is $n + 1$).

The set of edges E_{VN} in the channel dependency graph D_{VN} for $\mathrm{TVI}_{\mathrm{VN}}$ is

$$E_{\mathrm{VN}} = \{(\mathrm{vc}_{\mathrm{VN,SC1,DIM1},L1,\mathrm{DIR1}}, \mathrm{vc}_{\mathrm{VN,SC2,DIM2},L2,\mathrm{DIR2}}) \mid \mathrm{vd}_{\mathrm{VN,SC1,DIM1},L1,\mathrm{DIR1}}$$



(a)

Fig. 5. Virtual interconnection network for a torus-connected bidirectional 4-ary 2-cube. (a) TVI_0. (b) TVI_1.

$$= vs_{VN,SC2,DIM2,L2,DIR2} \wedge$$
$$\text{if DIM1} = \text{DIM2} = 0 \text{ then DIR1} = \text{DIR2}\}. \quad (5)$$

That is, any channel with destination virtual node A can be connected to any channel with source virtual node A with the restriction that the channels are not in opposite directions in dimension 0.

Assertion 2: The virtual interconnection network, TVI_{VN}, with its associated channel dependency graph D_{VN} defined for a torus-connected bidirectional k-ary n-cube is deadlock-free.

Proof: To show that TVI_{VN} is deadlock-free, it must be proven that D_{VN} has no cycles. By definition, D_{VN} has no 1-cycles. Following the proof used in assertion 1, assign the number

$$x_{VN,SC,DIM,L,DIR} = (L \times k^n) + \sum_{i=1}^{n-1} \begin{cases} \text{dig(SC}, i) \times k^i & \text{if dig(VN}, i-1) = 0 \\ [(k-1) - \text{dig(SC}, i)] \times k^i & \text{if dig(VN}, i-1) = 1 \end{cases} + \begin{cases} 0 & \text{if DIM} \neq 0 \\ \text{dig(SC, 0)} \times 1 & \text{if DIM} = 0 \wedge \text{DIR} = 0 \\ [(k-1) - \text{dig(SC, 0)}] \times 1 & \text{if DIM} = 0 \wedge \text{DIR} = 1 \end{cases} \quad (6)$$

to virtual channel $vc_{VN,SC,DIM,L,DIR}$. We have effectively defined new node position digits that always decrease in the direction of the channels. The only anomaly is that the 0th digit of the node position for channels not in dimension 0 is ignored. This is done because a channel in the 0th dimension must be able to enter a channel in another direction regardless of its position in the 0th dimension. Now, if $(vc_{VN,SC1,DIM1,L1,DIR1}, vc_{VN,SC2,DIM2,L2,DIR2})$ is an element of E_{VN}, there are several possible cases for the numbers assigned to these channels.

1) If $vc_{VN,SC1,DIM1,L1,DIR1}$ is not an end-around channel, then $L2 = L1$ and:

   a) If $(\text{DIM1} \neq 0 \wedge \text{dig(VN, DIM1} - 1) = 0) \vee (\text{DIM1} = \text{DIM2} = 0 \wedge \text{DIR1} = 0)$ then $\text{dig(SC2, DIM1)} = \text{dig(SC1, DIM1)} - 1$ which reduces the summation by k^{DIM1}. If the first channel is not in dimension 0 but the second is, there can also be a change in that part of the sum associated with the 0th digit of SC. However, the maximum increase would be $k-1$ while the summation is reduced by $k^{\text{DIM1}} > k-1$ if $\text{DIM1} \neq 0$.

   b) If $(\text{DIM} \neq 0 \wedge \text{dig(VN, DIM1} - 1) = 1) \vee (\text{DIM1} = \text{DIM2} = 0 \wedge \text{DIR1} = 1)$ then

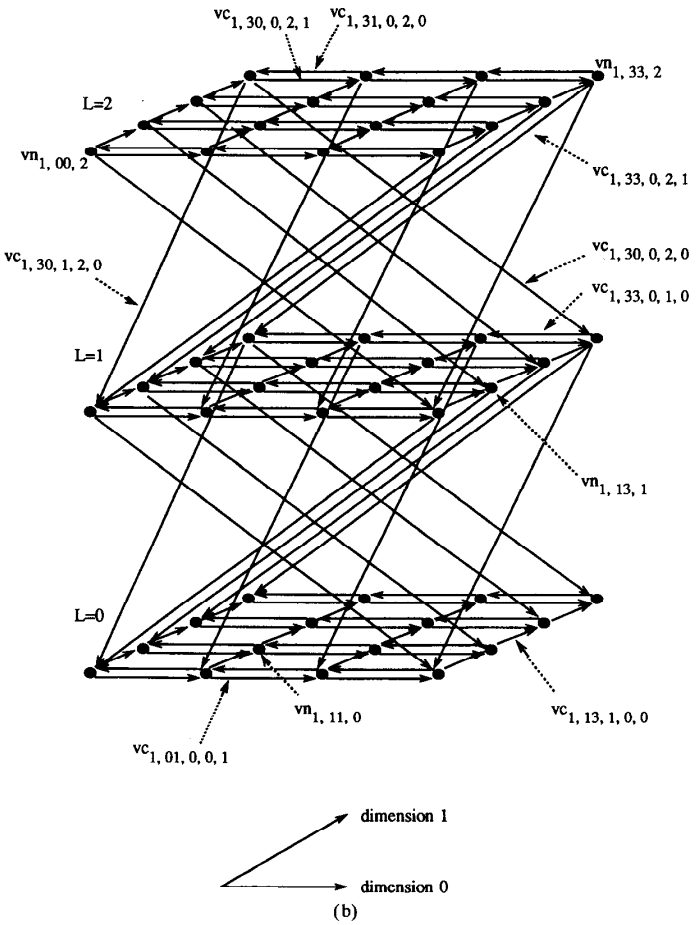


Fig. 5. (*Continued*)

$\mathrm{dig}(\mathrm{SC2}, \mathrm{DIM1}) = \mathrm{dig}(\mathrm{SC1}, \mathrm{DIM1}) + 1$. Thus, $[(k-1) - \mathrm{dig}(\mathrm{SC1}, \mathrm{DIM1})] - \{(k-1) - [\mathrm{dig}(\mathrm{SC1}, \mathrm{DIM1}) + 1]\} = 1$ which reduces the summation by k^{DIM1}. If the first channel is not in dimension 0 but the second is, there can also be a change in that part of the sum associated with the 0th digit of SC. However, the maximum increase would be $k-1$ while the summation is reduced by $k^{\mathrm{DIM1}} > k-1$ if $\mathrm{DIM1} \neq 0$.

c) If $(\mathrm{DIM1} = 0 \wedge \mathrm{DIM2} \neq 0)$ then the part of the sum associated with the 0th digit of SC is 0. This reduces the total sum by at least 1 since if $\mathrm{dig}(\mathrm{SC1}, 0) = 0$ then $\mathrm{DIR1} = 1$ and if $\mathrm{dig}(\mathrm{SC1}, 0) = k-1$ then $\mathrm{DIR1} = 0$.

2) If $\mathrm{vc}_{\mathrm{VN,SC1,DIM1},L1,\mathrm{DIR1}}$ is an end-around channel, then $L2 = L1 - 1$ and:

a) If $(\mathrm{DIM1} \neq 0 \wedge \mathrm{dig}(\mathrm{VN}, \mathrm{DIM1}-1) = 0) \vee (\mathrm{DIM1} = \mathrm{DIM2} = 0 \wedge \mathrm{DIR1} = 0)$ then $\mathrm{dig}(\mathrm{SC2}, \mathrm{DIM1}) = k-1$. Since $\mathrm{dig}(\mathrm{SC1}, \mathrm{DIM1}) = 0$, the summation is increased by $(k-1)k^{\mathrm{DIM1}}$. If the first channel is not in dimension 0 but the second is, there can also be a change in that part of the sum associated with the 0th digit of SC. However, the maximum increase would be $k-1$; so the total increase in x would be no more than $(k-1)k^{\mathrm{DIM1}} + k - 1$.

b) If $(\mathrm{DIM} \neq 0 \wedge \mathrm{dig}(\mathrm{VN}, \mathrm{DIM1}-1) = 1) \vee (\mathrm{DIM1} = \mathrm{DIM2} = 0 \wedge \mathrm{DIR1} = 1)$ then $\mathrm{dig}(\mathrm{SC2}, \mathrm{DIM1}) = 0$. Since $\mathrm{dig}(\mathrm{SC1}, \mathrm{DIM1}) = k-1$, the summation is still increased by $(k-1)k^{\mathrm{DIM1}}$. If the first channel is not in dimension 0 but the second is, there can also be a change in that part of the sum associated with the 0th digit of SC. However, the maximum increase would be $k-1$; so the total increase in x would be no more than $(k-1)k^{\mathrm{DIM1}} + k - 1$.

c) If $(\mathrm{DIM1} = 0 \wedge \mathrm{DIM2} \neq 0)$ then the part of the sum associated with the 0th digit of SC is 0. This does not increase the total sum.

Thus, $x_{\mathrm{VN,SC1,DIM1},L1,\mathrm{DIR1}} > x_{\mathrm{VN,SC2,DIM2},L2,\mathrm{DIR2}}$ since $x_{\mathrm{VN,SC2,DIM2},L2,\mathrm{DIR2}}$ is increased by at most $(k-1)k^{\mathrm{DIM1}} + k - 1$ but it is decreased by $k^n = (k-1)k^{n-1} + k^{n-1} > (k-1)k^{\mathrm{DIM1}} + k - 1$ if $n > 1$. If $n = 1$, the additional $k-1$ term would never be present and the inequality would still hold. All the cases yield $x_{\mathrm{VN,SC1,DIM1},L1,\mathrm{DIR1}} > x_{\mathrm{VN,SC2,DIM2},L2,\mathrm{DIR2}}$ so D_{VN} is acyclic, and thus, $\mathrm{TVI}_{\mathrm{VN}}$ is deadlock-free. □

As with the UVI, the key feature of the TVI_{VN} virtual interconnection networks is the adaptability they provide for the physical network. Since, in TVI_{VN}, a flit can exit a virtual node in any dimension, a message can move through dimensions at will taking any shortest path (other paths can be taken as long as the number of end-around channels it travels through does not exceed the number of levels minus one). In addition, since all the networks are complete unidirectional k-ary n-cubes, a message can travel to its destination in any of the virtual networks although some will allow shorter paths than others.

C. Mesh-Connected Bidirectional k-ary n-cubes

A bidirectional k-ary n-cube in which the end-around channels have been eliminated is called mesh connected. Unfortunately, the penalty for removing the end-around channels is that the longest path a message might have to travel is doubled (i.e., the *diameter* of the network is doubled) and symmetry is lost resulting in unevenly distributed traffic. The physical interconnection network for a mesh-connected 4-ary 2-cube is identical to the network in Fig. 4 except that the end-around connections are dropped. The physical nodes and channels are identified in the same manner as for the torus-connected cube. The destination node of $\text{pc}_{\text{SC,DIM,DIR}}$ is $\text{pd}_{\text{SC,DIM,DIR}} = \text{pn}_N$ where $\text{dig}(N, \text{DIM})$ is defined the same as in (1) except that the modulo functions are dropped since there are no end-around channels.

For this topology, there will be 2^{n-1} virtual networks for a mesh-connected bidirectional k-ary n-cube identified by MVI_{VN} and each will have different restrictions on the directions of its channels. These different networks are required to eliminate multidimensional loops, but each network is simplified because no levels are required to break up loops resulting from end-around channels. A message will still use a particular network depending on where it wishes to go. For example, if the source node of a message traveling in a 5-ary 3-cube were at position 114 and the destination node were at position 341, the virtual network identified with VN = 11 would be used.

For any mesh-connected bidirectional k-ary n-cube, the virtual nodes in the virtual interconnection network MVI_{VN} are identified by a label $\text{vn}_{\text{VN},N}$. Each virtual node is mapped to the physical node with the same node position, i.e., $\text{nmap}(\text{vn}_{\text{VN},N}) = \text{pn}_N$. The virtual channels are identified by a label $\text{vn}_{\text{VN,SC,DIM,DIR}}$ (j has been expanded into three integers) where DIR is defined the same as in (2). The destination node of $\text{vc}_{\text{VN,SC,DIM,DIR}}$ is $\text{vd}_{\text{VN,SC,DIM,DIR}} = \text{vn}_{\text{VN},N}$ where $\text{dig}(N, \text{DIM})$ is defined the same as in (3) except the modulo functions are dropped. Also, each virtual channel is mapped to the physical channel that is between the same nodes, i.e., $\text{cmap}(\text{vc}_{\text{VN,SC,DIM,DIR}})$ is defined the same as in (4).

Fig. 6 shows the two virtual interconnection networks, MVI_1 and MVI_0, for a 4-ary 2-cube. Note that the lack of levels greatly reduces the number of virtual channels. The set of edges E_{VN} in the channel dependency graph D_{VN} for MVI_{VN} is the same as (5) except the L subscripts are dropped.

Assertion 3: The virtual interconnection network, MVI_{VN}, with its associated channel dependency graph D_{VN} defined for a mesh-connected bidirectional k-ary n-cube is deadlock-free.

Proof: To show that MVI_{VN} is deadlock-free, it must be proven that D_{VN} has no cycles. By definition, D_{VN} has no 1-cycles. Following the procedure used in the proof of Assertion 1, assign the number $x_{\text{VN,SC,DIM,DIR}}$ to virtual channel $\text{vc}_{\text{VN,SC,DIM,DIR}}$ where $x_{\text{VN,SC,DIM,DIR}}$ is defined as in (6) except the term associated with L is dropped. The proof that $x_{\text{VN,SC1,DIM1,DIR1}} > x_{\text{VN,SC2,DIM2,DIR2}}$ is identical to case 1 of the proof for Assertion 2.

III. Fault Tolerance

A message passing communication system is fault tolerant if failures in some nodes do not prevent other nodes from communicating. The adaptability afforded by multiple paths is the key to fault tolerance when a node is truly unusable (i.e., there are no redundant nodes that can be "switched" in to replace the failed node). It may be inconvenient, however, to have some "global" controller in a network that is aware of all faults. A simpler and more distributed system would allow a message to start out along any path to its destination and then, route around faulted nodes as they are encountered. This section will describe additions to the virtual interconnection networks detailed in previous sections that allow them to be tolerant of at least one faulty node.

Consider first the unidirectional k-ary n-cube. Suppose the first flit of a message is presently at a node pn_{N1} and the destination is pn_{N2}. If $N1$ and $N2$ differ in more than one digit, then there is more than one dimension in which the flit can leave pn_{N1} to move toward its destination, and if one is blocked by failure, another can be used. If $N1$ and $N2$ differ in only one dimension, say DIM1, and the channel in this dimension is blocked by a failure in the adjacent node, then the flit will have to temporarily move away from its destination. That is, it can travel through one channel in some other dimension, say DIM2, and then finish its movement in dimension DIM1. To get to its destination, all it has to do is completely traverse dimension DIM2 using an end-around channel. Note that the message was "unaware" when it started that it would have to make an extra movement through an end-around channel in DIM2 to avoid a failed node. Thus, another level should be added to UVI so that a message can make an end-around movement in every dimension plus one more for fault tolerance.

The same fault-avoidance method applies to torus-connected bidirectional k-ary n-cubes. There is an additional method, however, if the dimension that is blocked, DIM, is not dimension 0. In this case, the message can travel through one channel in dimension 0, complete its motion in DIM, and then reach its destination by moving in the opposite direction in dimension 0.

For the mesh-connected bidirectional k-ary n-cubes, if the blocked dimension, DIM, is not dimension 0 then the method just discussed for the torus-connected cube applies. However, when DIM = 0, there is no end-around channel in some other dimension to use. One solution is to make dimension 1 bidirectional with special virtual channels that are switched on only if there is a fault. Fig. 7 illustrates one of these special channels, vc_1, (only dimensions 0 and 1 are shown). They were not considered in the proof of Assertion 3, so the question of deadlock arises. There were no cycles in the channel dependency graph before the special channel vc_1 was added; so if there is a cycle after it is added, the cycle must involve vc_1. We assume the routing system will only allow two new connections involving $\text{vc}_1 : \text{vc}_2$ to vc_1 and vc_1 to vc_3. If there is a cycle, then vc_3 must connect to some channel that connects to others that eventually connect to vc_2. This is impossible since a channel in dimension 0 cannot connect to another channel in dimension 0 that is in the opposite direction. Thus, any attempt to get back to vc_2 would require a movement in some dimension other than dimension 0, and since these are all unidirectional, there is no way to return by moving in the opposite direction. By limiting which channels connect to vc_1, deadlock is avoided.

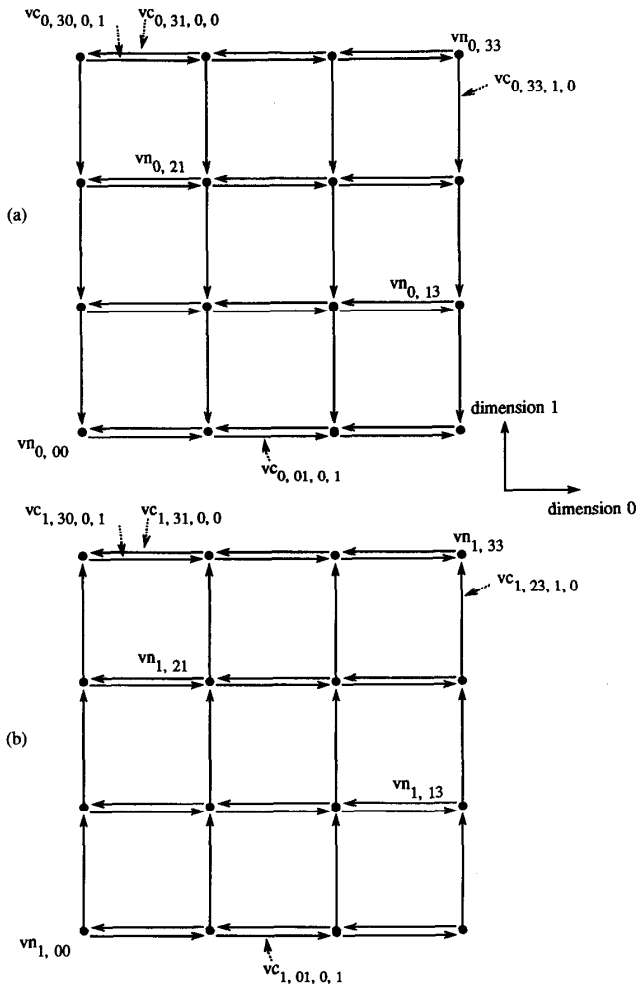


Fig. 6. Virtual interconnection network for a mesh-connected bidirectional 4-ary 2-cube. (a) MVI_0. (b) MVI_1.

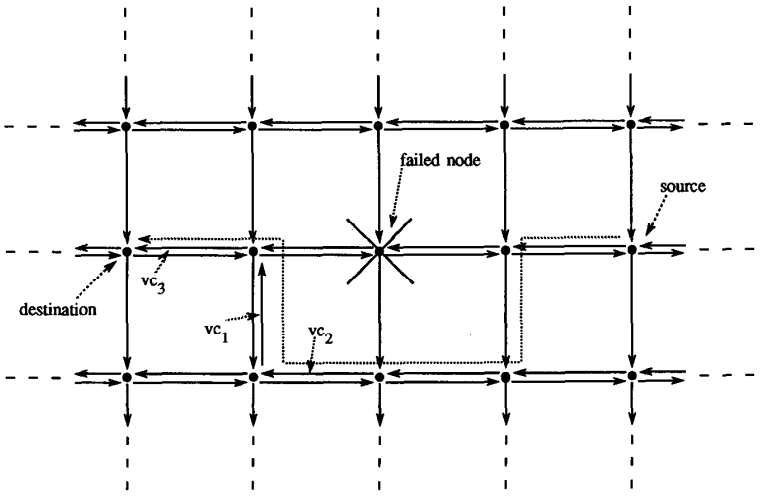


Fig. 7. Avoiding a failed node in a mesh-connected bidirectional k-ary n-cube.

IV. Conclusion

To escape the constraints of a fixed physical interconnection network, virtual interconnection networks can be created that are time multiplexed over the physical channels. Virtual interconnection networks have been described in this paper that provide for adaptability and fault tolerance. Other approaches to adaptability using an exchange model and prioritized messages are under development [12], [13]. These approaches are valid for general topologies but involve more complex decision processes. We believe that the virtual network scheme described here may lead to a simpler implementation for regular topologies like the k-ary n-cube.

Although the physical channels are shared by the virtual channels, virtual channels still have a cost in terms of queue space and switching mechanisms. The unidirectional k-ary n-cube cannot exploit locality of communication, so it is probably undesirable for most purposes. It was studied as an introduction to the more powerful, but complex bidirectional k-ary n-cube. With 2^{n-1} virtual interconnection networks, $n+1$ levels per network, and k^n virtual channels per level for the bidirectional k-ary n-cube, the number of virtual channels rapidly increases with n. However, research has shown that low-dimensional cubes (i.e., $n = 2$ to 4) are likely to be the best networks for general purpose processing [13]. If the routing function and virtual queues were implemented in software, a bidirectional k-ary n-cube of low dimension would certainly be practical and would offer a great degree of adaptability and fault tolerance. The mesh-connected bidirectional k-ary n-cube, especially in two dimensions, costs very little and is probably best suited for hardware implementations of the routing function and virtual queues. Future research could involve the development of routing functions that examine the present state of the message traffic in the network and make optimum use of the adaptability and fault tolerance allowed by the virtual structures described in this paper.

References

[1] W. C. Athas and C. L. Seitz, "Multicomputers," Tech. Rep. 5244:TR:87, Dep. Comput. Sci., California Instit. Technol., 1987.

[2] C. L. Seitz, "The Cosmic Cube," *Commun. ACM*, vol. 28, pp. 22–23, Jan. 1985.

[3] I. S. Gopal, "Prevention of store and forward deadlock in computer networks," *IEEE Trans. Commun.*, vol. COM-33, no. 12, pp. 1258–1264, Dec. 1985.

[4] K. D. Gunther, "Prevention of deadlocks in packet-switched data transport systems," *IEEE Trans. Comput.*, vol. C-29, no. 4, pp. 512–524, Apr. 1981.

[5] S. Toueg and J. D. Ullman, "Deadlock-free packet switching networks," *SIAM J. Comput.*, vol. 10, no. 3, pp. 594–611.

[6] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Comput.*, vol. C-36, no. 5, pp. 547–553, May 1987.

[7] *Submicron Systems Architecture Semiannual Technical Report*, Tech. Rep. 5235:TR:86, Dep. Comput. Sci., California Instit. Technol., 1986.

[8] W. D. Hillis, "The connection machine," *Scient. Amer.*, vol. 256, no. 6, pp. 108–115, June 1987.

[9] W. Welch, "Message-driven solver for Euler fluid dynamics equations," in *Proc. 1988 ACM Southeast Regional Conf.*, Apr. 1988.

[10] B. Carre', *Graphs and Networks*. Oxford, England: Clarendon, 1979, pp. 50–51.

[11] J. Blazewicz, J. Brzezniski, and G. Gambosi, "Time-stamp approach to store-and-forward deadlock prevention," *IEEE Trans. Commun.*, vol. COM-35, no. 5, pp. 490–495, May 1987.

[12] J. Y. Ngai and C. L. Seitz, "A framework for adaptive routing," Tech. Rep. 5246:TR:87, Dep. Comput. Sci., California Instit. Technol., 1987.

[13] W. J. Dally, *A VLSI Architecture for Concurrent Data Structures*. Boston, MA: Kluwer Academic, 1987, pp. 144–161.

Daniel H. Linder (M'89) was born in Baltimore, MD, on November 15, 1962. He received the B.S.E.E. and M.S.E.E. degrees from Mississippi State University, Mississippi State, in 1985 and 1987, respectively.

In 1988 he was employed in the aerospace processing group of Westinghouse, Baltimore, MD. He is presently a Research Engineer with the NSF Engineering Research Center for CFS at Mississippi State University. His research interests include concurrent computing, VLSI, and simulation.

Mr. Linder is a member of Phi Kappa Phi, Tau Beta Pi, and the IEEE Computer Society.

Jim C. Harden (S'82–M'85) was born in Cleveland, MS, on November 3, 1943. He received the B.S.E.E. degree from Mississippi State University, Mississippi State, in 1965, the M.S.E.E. degree from Georgia Institute of Technology, Atlanta, in 1966, and the Ph.D. degree in electrical engineering from Texas A&M University, College Station, in 1985.

Presently, he is an Associate Professor of Electrical Engineering at Mississippi State University. Prior to this appointment, he worked in the telemetry and data acquisition areas in industry, beginning in 1966 at Radiation (now Harris) and then at IBM Federal Systems Division, Huntsville, AL. Later as a founding member of Care Electronics (now Care Monitoring Systems) he was involved in the application of digital telemetric techniques to commercial cardiac monitoring. His current research interests include wafer scale integration, cellular computing structures, and real-time data acquisition systems.

Dr. Harden is a member of the IEEE Computer Society, Sigma Alpha Epsilon, Tau Beta Pi, and Eta Kappa Nu.