Short Notes

Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels

William J. Dally and Hiromichi Aoki

Abstract— **The use of adaptive routing in a multicomputer interconnection network improves network performance by making use of all available paths and provides fault tolerance by allowing messages to be routed around failed channels and nodes. This paper describes two deadlock-free adaptive routing algorithms. Both algorithms allocate virtual channels using a count of the number of *dimension reversals* a packet has performed to eliminate cycles in resource dependency graphs. The *static algorithm* eliminates cycles in the network channel dependency graph. The *dynamic algorithm* improves virtual channel utilization by permitting dependency cycles and instead eliminating cycles in the packet *wait-for graph*. We prove that these algorithms are deadlock-free and give experimental measurements of their performance. For nonuniform traffic patterns, these algorithms improve network throughput by a factor of three compared to deterministic routing. The dynamic algorithm gives better performance at moderate traffic rates but requires source throttling to remain stable at high traffic rates. Both algorithms allow the network to gracefully degrade in the presence of faulty channels.**

Index Terms—**Communication networks, concurrent computing, flow control, interconnection networks, multicomputers, packet routing, parallel processing.**

I. Introduction

A. Interconnection Networks

Interconnection networks are used to pass messages containing data and synchronization information between the nodes of concurrent computers [2], [28], [42], [16]. The messages may be sent between the processing nodes of a message-passing multicomputer [2] or between the processors and memories of a shared-memory multiprocessor [28]. The interconnection network is often the critical component of a large parallel computer because performance is very sensitive to network latency and throughput and because the network accounts for a large fraction of the cost and power dissipation of the machine. To be cost effective, a network should provide throughput approaching network capacity to keep the costly network wires and router pins productive.

Multicomputer networks are characterized by regular topologies, large numbers (10^2 to 10^5) of nodes, small node size, and high speed. The speed and node size constraints limit buffer storage to a few flow-control digits (flits) per channel requiring the use of wormhole routing [14] rather than store-and-forward or virtual cut-through [29]. To achieve high-performance, the routing decision must be reduced to a combinational logic function so it can be made in a single clock cycle. Routing tables that result in quasi-static routing and grow quadratically with machine size are unsuitable in this context. On the positive side, regular topology permits algorithmic routing and short link distances eliminate transmission errors and hence reduce protocol overhead. The algorithms described here are suitable for implementation in a multicomputer routing controller. They require minimal storage, can be used with wormhole routing, and are simple enough to be reduced to a few levels of combinational logic.

An interconnection network is described by its topology, routing, and flow control. The topology of a network is the arrangement of its nodes and channels into a graph. Routing determines the path chosen by a message in this graph. Flow control deals with the allocation of channel and buffer resources to a message as it travels along this path. This paper deals with routing and flow control. It describes two deadlock-free, nonminimal adaptive routing algorithms that select paths in a network, N, using information about the current state of N. These algorithms improve network throughput for nonuniform traffic patterns by balancing network load along alternate paths. The methods described here are applicable to any topology; however, the examples in this paper consider their application to k-ary n-cube interconnection networks [14].

B. The Problem

Most existing multicomputer routing networks [28], [42], [16] use deterministic routing. With deterministic routing, the path followed by a packet is determined solely by its source and destination. If any channel along this path is heavily loaded, the packet will be delayed. If any channel along this path is faulty the packet cannot be delivered. A common deterministic routing algorithm is dimension-order routing, where the packet is routed in one dimension at a time, arriving at the proper coordinate in each dimension before proceeding to the next dimension.

Adaptive routing improves both the performance and fault tolerance of an interconnection network. Fig. 1 shows a 8-ary 2-cube in which the node at $(i, 0)$ sends a packet to the node at $(7, i)$ (for $i \in [0, 7]$). With dimension order deterministic routing [Fig. 1(a)], seven of the eight packets must traverse the channel from (6, 0) to (7, 0). Thus only one of these seven packets can proceed at a time. With adaptive routing [Fig. 1(b)] all of the packets can proceed simultaneously using alternate paths. For the traffic pattern shown in this example, adaptive routing increases throughput by a factor of seven.

Fig. 2 shows the same network with a faulty channel from (3, 4) to (4, 4). With dimension-order deterministic routing, packets from node $(i, 4)$ to node (j, k) where $i \leq 3 < j$ cannot be delivered. With adaptive routing, all messages can be delivered by routing around the faulty channel. To deliver a packet from $(i, 4)$ to $(j, 4)$ where $i \leq 3 < j$, it is necessary for the packet to be *misrouted* away from the destination resulting in a nonminimal distance route.

C. Adaptive Routing with Virtual Channels

Adaptive routing must be performed in a manner that is deadlock-free. Deadlock in an interconnection network occurs whenever there is a cyclic dependency for resources: buffers or channels. Networks that use dimension order routing avoid deadlock by ordering channels so that messages travel along paths of strictly increasing channel

Manuscript received October 30, 1990; revised June 30, 1991 and November 20, 1991. This work was supported in part by the Defense Advanced Research Projects Agency under Contracts N00014-88K-0738 and N00014-87K-0825 and in part by a National Science Foundation Presidential Young Investigator Award, Grant MIP-8657531, with matching funds from General Electric Corporation and IBM Corporation.

The authors are with the Artificial Intelligence Laboratory and Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139.

IEEE Log Number 9206275.

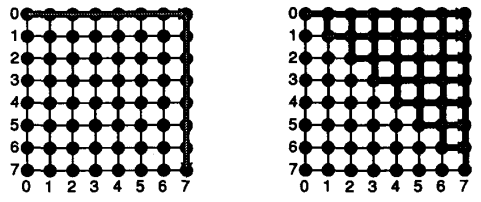

Fig. 1. Routing packets in an 8-ary 2-cube from $(i, 0)$ to $(7, i)$ (for $i \in [0, 7]$). (a) Using dimension order routing, seven packets must traverse the channel from (6, 0) to (7, 0). (b) Using adaptive routing, all packets proceed simultaneously increasing throughput by a factor of 7.

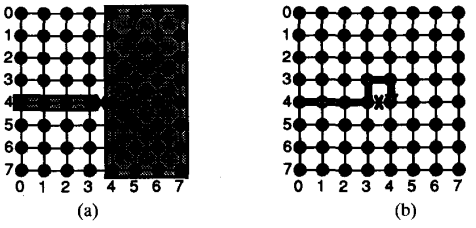

Fig. 2. An 8-ary 2-cube network with a faulty channel from (3, 4) to (4, 4). (a) With dimension order routing, packets from the shaded area on the left to the shaded area on the right cannot be delivered. (b) Using adaptive routing packets can be delivered between all pairs of nodes.

numbers [18]. Channels are ordered so that all of the channels in each dimension are greater than all of the channels in the preceding dimension. This ordering eliminates cycles in the channel dependency graph and thus prevents deadlock. The ordering, however, results in a unique path from source to destination and thus does not allow adaptive routing.

This paper presents two deadlock-free adaptive routing algorithms. Both algorithms permit misrouting (routing a packet along a nonshortest path), avoid deadlock using virtual channels to eliminate cyclic dependencies, and introduce the use of *dimension reversal (DR) numbers* to break dependency cycles.

The *static algorithm* applies DR numbers to the method of [18] to eliminate cycles in the channel dependency graph by numbering virtual channels and routing packets to traverse virtual channels in increasing order. Each packet is labeled with a DR number that is initialized to zero. When a packet performs a dimension reversal, routing from a higher dimension to a lower dimension, its dimension reversal number is incremented and it is routed on a *class* of virtual channels used only by packets with the same dimension reversal number. The number of classes of virtual channels places an upper limit on the maximum number of dimension reversals permitted. Once a packet has made this number of dimension reversals, it is restricted to dimension-order routing.

This static assignment of DR numbers to virtual channels restricts the number of DR's permitted and makes inefficient use of virtual channels. A packet may be blocked waiting for a virtual channel in its DR class, while other virtual channels for the same physical channel remain idle.

The *dynamic algorithm* overcomes these limitations by permitting packets to use any available virtual channel. Deadlock is avoided by eliminating cycles from the packet *wait-for graph*. Packets are labeled with their dimension reversal number and are not permitted to wait for a virtual channel held by a packet with a lower dimension reversal number. If all available virtual channels are occupied by packets with lower dimension reversal numbers, the packet reverts to dimension-order routing on an additional set of virtual channels reserved for this purpose. The dynamic algorithm places no restrictions on the number of dimension reversals permitted.[1]

D. Related Work

Adaptive routing has been extensively studied in the context of wide-area networks (WAN's) and local-area networks (LAN's) [24], [5], [43], [4] and proposals have been made to apply these LAN/WAN techniques to multicomputers [23], [41], [30]. However, as described above, LAN/WAN routing algorithms are not directly applicable to multicomputer networks for three reasons. First, most LAN and WAN routing algorithms are table driven making them impractical for large multicomputers as the total table storage grows as N^2. The regular topology of multicomputer networks allows routing without tables. Second, most LAN/WAN routing algorithms (e.g., [24]) assume quasi-static network traffic while traffic in multicomputer networks changes rapidly. Finally, most WAN's/LAN's use deadlock avoidance strategies based on packet buffer assignment [27], [35], [25] that use too much storage to be implemented in single-chip routers.

Several algorithms have been developed for permutation routing [37], [38] in which the traffic pattern is fixed and known *a priori*. These algorithms are not applicable to a multicomputer network where a single routing algorithm must support many traffic patterns that are not known *a priori* and may not be permutations and where routing must be done on-line using only local information.

Virtual channels were introduced in [18] for deadlock avoidance. Virtual channels have also been used to support multiple virtual circuits [6], and to increase network throughput [15]. Minimal, deadlock-free adaptive routing algorithms based on virtual channels are described in [12], [13], [34], and [31]. These algorithms do not permit misrouting and thus cannot route around certain network faults. An adaptive wormhole routing algorithm that permits misrouting is described in [36]; however, this algorithm is not deadlock-free. Store-and-forward adaptive routing algorithms have been developed based on packet exchange, [40], [39], promotion [1], and randomization [33], [32]. These three algorithms require that entire packets be buffered and thus cannot be used with wormhole routing. Circuit-switched networks have used adaptive routing algorithms based on tree search [26], [10]. Chen and Shin develop and analyze adaptive routing algorithms for hypercubes in [7], [8]. However, they consider only the problem of finding routes and do not address issues of deadlock, livelock, or contention. Duato has proposed a method for nonminimal adaptive routing using virtual channels by ensuring that a connected subset of the channels is acyclic [21], [20]. A random oblivious adaptive routing scheme is proposed in [44] and [45]; however this scheme destroys any locality present in communications and doubles average communication distances.

E. Outline

The next section introduces the notation, terminology, and assumptions that will be used throughout this paper. Section III describes the two deadlock-free adaptive routing algorithms in more detail, proves that they are deadlock free, and discusses routing policy. These algorithms are evaluated experimentally in Section IV. The experiments measure network performance on uniform and nonuniform traffic patterns, evaluate the effect of selection functions and throttling, and determine how network performance degrades as network channels are failed.

[1] As a practical matter, the number of dimension reversals will be limited by the size of the packet header field used to hold the packet's dimension reversal number.

II. Preliminaries

Topology: An interconnection network is a strongly-connected, directed graph, $I = G(N, C)$. The vertices of I are a set of *nodes*, N. The edges are a set of *channels*, $C \subseteq N \times N$. Each channel is unidirectional and carries data from a source node to a destination node. A bidirectional network is one where $(n1, n2) \in C \Rightarrow (n2, n1) \in C$.

Flow Control: Communication between nodes is performed by sending *messages*. A message may be broken into one or more *packets* for transmission. A packet is the smallest unit of information that contains routing and sequencing information. A packet contains one or more flow control digits or *flits*. A flit is the smallest unit on which flow control is performed. Information is transferred over physical channels in physical transfer units or *phits*. A phit is usually the same size or smaller than a flit.

Each physical channel, $c_i \in C$, in the network is composed of one or more virtual channels, $c_{ij} \in C'$. The virtual channels associated with a single physical channel share physical channel bandwidth, allocated on a flit-by-flit basis. However, each virtual channel contains its own queue and is allocated on a packet-by-packet basis independently of the other virtual channels. For purposes of deadlock analysis, each virtual channel is logically a separate channel.

Routing: A packet is assigned a route through the network according to a *routing relation*, $R \subseteq C' \times N \times C'$. Given the virtual channel occupied by the head of the packet and the destination node of the packet, the routing relation specifies a (possibly singleton) set of virtual channels that may be used for the next step of the packet's route.

A selection function, $\rho(P(C'), \alpha) \mapsto C'$, is used to pick the next channel of the route from the elements of this set using some additional information, α. This additional information may include the occupancy and/or operational status of channels in the network. The next channel selected for a packet, p_i, is denoted $next(p_i)$.

The *channel dependency graph* for an interconnection network, I, and routing relation, R, is a directed graph, $D = G(C', E)$. Its vertices are C', the virtual channels of I, and its edges are given by the projection of the routing relation onto $C' \times C'$:

$$E = \{(c_i, c_j) | (c_i, n, c_j) \in R \text{ for some } n \in N\}. \quad (1)$$

Consider a network, I, occupied by a set of packets, P, where each packet, $p_i \in P$, occupies a particular set of virtual channels, $\text{occ}(p_i) \subseteq C'$. The *wait-for graph* of I is a directed graph, $W = G(P, E_W)$. The vertices of W are P, the set of packets in the network at a given instance of time. There is an edge of W, $e_i \in E_W$, for each packet that is waiting on another packet to acquire a resource:

$$E_W = \{(p_i, p_j) | next(p_i) \in \text{occ}(p_j)\}. \quad (2)$$

Performance: The performance of a fault-free network is measured in terms of its *latency*, T, and its *throughput*, λ_{sat}. The latency of a message is the elapsed time from when the message is initiated until the message is completely received. Network latency is the average message latency under specified conditions. Network throughput is the number of messages the network can deliver per unit of time. The latency and throughput of a network degrade as channels fail. The rate at which they degrade gives a figure of merit for the network.

k-ary n-cube Networks: k-ary n-cube is a radix k cube with n dimensions, having $N = k^n$ nodes. The radix implies that there are k nodes in each dimension. The nodes in each dimension may be connected in a linear array giving a mesh network [Fig. 3(a)] or in a ring giving a torus network [Fig. 3(b)]. A k-ary 1-cube [Fig. 3(a), (b)] is a k-node ring or linear array. A k-ary 2-cube [Fig. 3(c)] is constructed by taking k k-ary 1-cubes and connecting like elements.

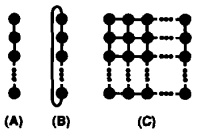

Fig. 3. k-ary n-cube is a cube of n dimensions with k nodes in each dimension. (a) Meshes are cubes with each dimension connected in a linear array. (b) Connecting each dimension in a ring gives a Tori. (c) Higher dimensional cubes are constructed by combining like elements of lower dimensional cubes.

In general, a k-ary n-cube is constructed from k k-ary $(n-1)$-cubes by connecting like elements into rings or linear arrays.

Every node has an address that is an n digit, radix k number, $a_{n-1} \cdots a_0$. Each address digit, a_d, represents a node's coordinate in dimension d and can take on values in the range $[0, k-1]$. In a torus network, nodes are connected to all nodes with an address that differs in only one digit by ± 1 mod k. In a mesh, nodes are connected to all nodes with an address that differs in one digit by ± 1 where the result is in the range $[0, k-1]$.

The dimensions and directions of the cube partition the set of virtual channels, C', into subsets for each dimension: $C'_{00}, C'_{01} \cdots, C'_{(n-1)0}, C'_{(n-1)1}$. A channel, c_i, with source, n_s, and destination, n_d, whose addresses differ in the dth position is said to be a channel in the dth dimension. If $n_s > n_d, c_i \in C'_{d0}$. If $n_s < n_d, c_i \in C'_{d1}$. This definition partitions the two directions of a given dimension into distinct channel sets.

Many networks are included in the family of k-ary n-cubes. At the extreme of $k = 2$, we have a binary n-cube. At the extreme of $n = 1$ we have a ring or linear array. For $n = 2$ we have a torus or 2-D mesh. These networks have been used in several message-passing computers [42], [11], [16]. For the remainder of this paper we consider only mesh-connected k-ary n-cubes.

III. Adaptive Routing Algorithms

This section describes two deadlock-free adaptive routing algorithms that use a packet's dimension reversal number to avoid cycles in resource dependency graphs. First, dimension reversals are defined, and the static and dynamic algorithms for assigning virtual channels to packets are presented and proved deadlock free. Finally, strategies for selecting among admissible channels that guarantee progress are described.

A. Dimension Reversals

The dimension reversal number of a packet is the count of the number of times a packet has been routed from a channel in one dimension, p, to a channel in a lower dimension, $q < p$. Dimension reversal (DR) numbers are assigned to packets as follows:

1) All packets are initialized with a DR of 0.
2) Each time a packet routes from a channel $c_i \in C'_p$ to a channel $c_j \in C'_q$ where $p > q$ the DR of a packet is incremented.

B. The Static Algorithm

The static algorithm divides the virtual channels of each physical channel into nonempty classes numbered zero to r, where r is the maximum number of dimension reversals permitted. Packets with $DR < r$ may be routed in any direction but must use only virtual channels of class DR. Once a packet has DR $= r$, it must use dimension-order routing on the virtual channels of class r. Thus, when a packet makes its final dimension reversal, it must start routing in

the lowest dimension in which its current node address differs from the destination address.

Assertion 1: The static algorithm is deadlock free.

Proof: The channel dependency graph is acyclic. Assign a number, $\mathrm{num}(c_i)$, to each channel, c_i, in each dimension, C_{dx}, so channel numbers increase in the direction of routing.[2] Now order all virtual channels according to their class, dimension, and number. With this ordering, packets using the static algorithm will always traverse channels in ascending order. Thus the channel dependency graph is acyclic and the routing is deadlock free. □

C. The Dynamic Algorithm

The dynamic algorithm divides the virtual channels of each physical channel into two nonempty classes: adaptive and deterministic. Packets originate in the adaptive channels. While in these channels, they may be routed in any direction without a maximum limit on the number of dimension reversals a packet may make. Whenever a packet acquires a channel it labels the channel with its current DR number. To avoid deadlock, a packet with a DR of p cannot wait on a channel labeled with a DR of q if[3] $p \geq q$. A packet that reaches a node where all output channels are occupied by packets with equal or lower DR's must switch to the deterministic class of virtual channels.[4] When a packet enters the deterministic channels, it must be routed in the lowest dimension in which the current node address and the destination address differ. Once on the deterministic channels, the packet must be routed in dimension order and cannot reenter the adaptive channels.

Assertion 2: The dynamic algorithm is deadlock free.

Proof: By contradiction. If the network is deadlocked, then there is a set of packets, P, waiting on virtual channels held by other packets in P. There exists a packet, $p_{\max}$, such that the $DR(p_{\max}) \geq DR(q) \forall q \in P$. Let r be the DR label of $c_n = next(p_{\max})$. Then $r \leq DR(p_j)$ where $c_n \in occ(p_j)$. However, $p_{\max}$ is not permitted to wait on $next(p_{\max})$ since $DR(p_{\max}) \geq DR(p_j) \geq (r)$. □

The dynamic algorithm is deadlock free even though it permits cycles in the network's channel dependency graph. This does not contradict Theorem 1 of [18] as that theorem assumes deterministic routing. In [18], R is a function, not a relation, so if a cycle exists in the channel dependency graph, a packet is required to follow the cycle. With adaptive routing, R is a relation. There may be many channels available to route a packet. Deadlock can be avoided by choosing a channel that does not create a cycle in the packet wait-for graph.

D. Routing Policy

Progress: The static and dynamic algorithms allow a packet to be routed deadlock-free along an arbitrary path in a k-ary n-cube network. These algorithms by themselves, however, give no guarantee that a packet will ever reach its destination.

To guarantee progress toward a destination, misrouting is limited by placing an upper limit on the number of steps a message may take away from its destination. With this limit, a weighted sum of the distance to the destination and the number of misrouting steps remaining for all messages in the network is strictly decreasing. Thus the network is livelock free. A variant on this scheme is to limit the ratio of misrouting steps to progress steps—e.g., no more than one step back for each two steps forward.

[2] This method can be extended to tori by using the method given in [18].

[3] Since any cycle contains at least one DR, it suffices to prohibit waits only when $p > q$.

[4] A packet may wait a finite amount of time before resorting to dimension-order routing.

Throttling: The adaptive virtual channel pool of the dynamic algorithm can be monopolized by eager sources unless some form of throttling [9] is used. If many sources attempt to inject messages into the network faster than the network is able to handle them, these new messages will consume all available virtual channels in the adaptive pool. Older messages will be forced to revert to deterministic routing using the deterministic pool.

Throttling can be performed by using a hybrid of the static and dynamic algorithms. The virtual channels are divided into classes as in the static algorithm. A packet with a DR of p is permitted to select a channel of class q only if $p \geq q$. This method divides the adaptive virtual channel pool into classes to prevent new messages from consuming the entire pool. In practice, two classes, 0 and 1, are sufficient to limit the channels consumed by injected messages.

Selection Functions: We denote the set of virtual channels that packet p_i is permitted to select for the next step of its route as $\eta(p_i) \subseteq C'$. This set is determined by the routing algorithm: static or dynamic to avoid deadlock. Also, any faulty channels are excluded from η. The single deterministic channel that p_i may route on is denoted $\delta(p_i)$ and is excluded from $\eta(p_i)$. The set of unoccupied channels in $\eta(p_i)$ that move p_i closer to its destination are denoted $\gamma(p_i)$. The selection function, *next*(p_i), chooses the next channel of the route as follows:

$$next(p_i) = \begin{cases} pick(\gamma(p_i)) & \text{if } |\gamma(p_i)| > 0 \\ pick(\eta(p_i)) & \text{if } |\gamma(p_i)| = 0 \text{ and } |\eta(p_i)| > 0 \\ \delta(p_i) & \text{if } |\gamma(p_i)| = 0 \text{ and } |\eta(p_i)| = 0 \end{cases} \quad (3)$$

The selection function reserves an unoccupied, productive channel if possible. If no such channels are available the function picks any legal adaptive channel. This channel may be occupied and/or unproductive. If the channel is occupied, the packet waits on the channel until it becomes free. Only if no legal adaptive channels are available does the packet resort to deterministic routing.

The function *pick*, $P(C') \mapsto C'$, chooses a next channel from a set of productive or permitted channels. In Section IV the selection criteria listed below are evaluated. Other criteria, including random, are possible.

- Minimum congestion: Pick the direction with the most available virtual channels.
- Maximum flexibility: Pick the direction with the greatest distance to travel to the destination. This is similar to the Z^2 routing policy proposed in [3].
- Straight lines: Pick the dimension closest to the current dimension.

IV. Experimental Results

To measure the performance of the adaptive routing algorithms described above, we have simulated a number of k-ary n-cube networks varying the routing relation, selection function, and traffic patterns. Faulty networks were simulated to measure performance degradation.

The simulator is a 9000-line C program that simulates interconnection networks at the flit level. A flit transfer between two nodes is assumed to take place in one time unit. The network is simulated synchronously, moving all flits that have been granted channels in one time step and then advancing time to the next step. The simulator is programmable as to topology, routing algorithm, and traffic pattern.

All of the results in this section are for 256-node 16-ary 2-cube mesh networks with 16 virtual channels per physical channel. This network was chosen because it is representative of existing machines. Simulations were run using dimension-order deterministic routing and

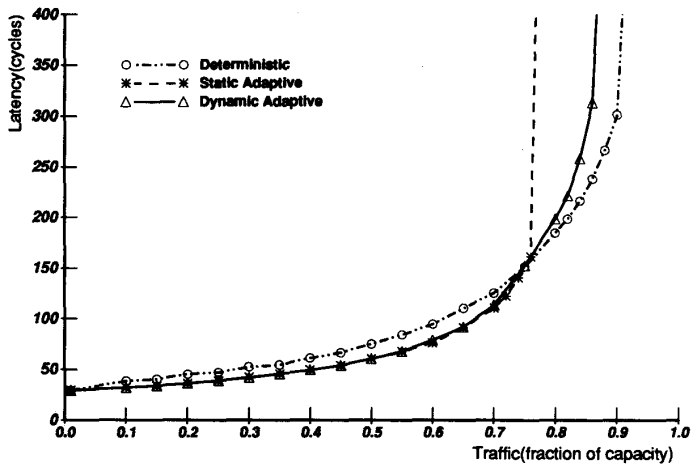

Fig. 4. Latency versus accepted traffic for a 16-ary 2-cube under random traffic. Deterministic, dimension-order routing is compared with static and dynamic adaptive routing. With random traffic, adaptive routing gives slightly lower latency with low traffic than deterministic routing but saturates first.

both static and dynamic adaptive routing. For the adaptive routing cases, the selection functions described in Section III-D are used. Livelock is avoided by placing an upper limit on DR numbers and hence misrouting steps.

A. Latency

Latency is measured by applying a constant rate source to each input and measuring the time from packet creation until the last flit of the packet is accepted at the destination. Source queueing time is included in the latency measurement.

Fig. 4 compares the performance of deterministic dimension-order routing with static and dynamic adaptive routing under uniform random traffic. The figure shows latency as a function of throughput for the three routing strategies. Both adaptive routing strategies use a selection function that favors minimum congestion, and both permit misrouting. For deterministic routing, saturation occurs at 94% capacity. For static and dynamic routing, saturation occurs at 78% and 88%, respectively.[5]

Random traffic places a uniform load on the network channels and buffers. Thus, adaptive routing affects performance only slightly for this traffic pattern. For small loads, adaptive routing slightly reduces latency by moving packets that would otherwise be blocked. However, above 75% capacity, adaptive routing gives a higher latency than deterministic routing. This is because dimension-order routing concentrates traffic on the through channels of each switch. Collision probability and hence expected latency is proportional to the competing traffic rate [13]. Thus concentrating traffic on the through channels results in less contention and lower latency. With adaptive routing, the switch traffic is more uniform, resulting in higher latency.

Dynamic adaptive routing outperforms static adaptive routing at high traffic levels. The dynamic algorithm allows more flexible buffer assignments allowing packets to make progress that would otherwise be blocked waiting on a particular buffer.

Adaptive routing gives a significant performance advantage for traffic patterns that load channels nonuniformly. Fig. 5 shows latency as a function of throughput for three routing strategies under bit-reversal traffic. In this traffic pattern, each node, i, sends messages to node j where j is the bit-reversal of i. For example, node 43_{16} sends messages to node $C2_{16}$. Deterministic routing gives very poor performance for this traffic pattern, saturating at 25% capacity. This saturation occurs because a few channels become bottlenecks as in Fig. 1(a). With adaptive routing, packets are routed around bottleneck channels achieving three times the throughput of deterministic routing. The static algorithm saturates at 60% capacity while the dynamic algorithm saturates at 75% capacity.

[5]These saturation throughputs occur at much higher latencies than those included on the y-axis of Fig. 4 and thus are not shown in the figure.

B. Throttling

Fig. 6 shows the effect of throttling on network throughput. The figure shows throughput (accepted traffic) as a function of offered traffic for a network using dynamic adaptive routing. The simulations were run using random traffic. The curves correspond to no throttling and throttling by restricting packets with a DR of 0 (entry packets) to use only one, two, or four virtual channels (entry lanes) of each physical channel. Throughput values are shown after 10 000 network cycles. At traffic rates in excess of peak throughput, throughput degradation is affected by both the intensity and duration of the traffic burst.

Table I shows the throughput and fraction of packets that are forced into deterministic routing for varying degrees of throttling under maximum traffic, when each network node attempts to inject a message into the network on each cycle. For this network, maximum traffic corresponds to four times network capacity and quickly congests all network buffers if throttling is not employed. The table shows that the number of packets forced onto the deterministic virtual channels is reduced by throttling. This improves fault tolerance since once a packet begins deterministic routing, it is vulnerable to a single channel fault.

Without throttling, a network using dynamic adaptive routing is unstable. As offered traffic is increased beyond 86% capacity, throughput decreases. At maximum traffic (all sources sending all the time), throughput is reduced to 11% capacity. Without throttling, the network can be pushed into this unstable state by a short burst of traffic in excess of capacity. Once overloaded in this manner, the network will not recover to normal operation until the offered traffic is reduced below the overload throughput (in this case 11% capacity).

Throttling the network by restricting entry packets to use a single virtual channel increases the overload throughput from 11% to 66% of capacity. With high overload throughput, the network recovers

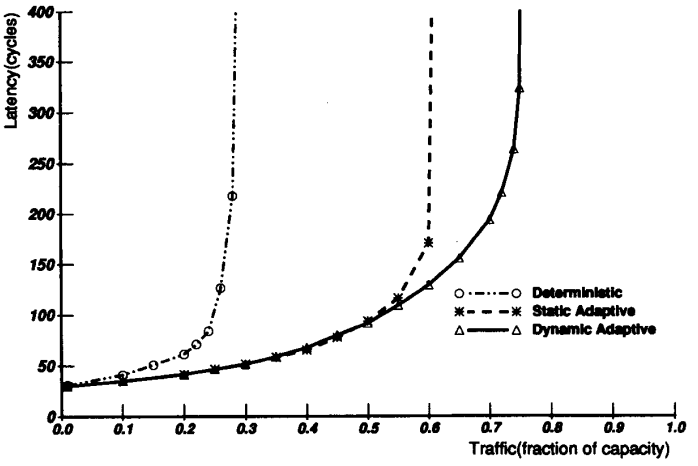

Fig. 5. Latency versus accepted traffic for a 16-ary 2-cube under bit reversal traffic. Deterministic dimension order routing is compared with static and dynamic adaptive routing. This nonuniform traffic pattern causes deterministic routing to perform very poorly, saturating at about 25% capacity. Static and dynamic adaptive routing achieve three times this performance (saturating at 60% and 75% capacity, respectively) by routing to distribute the network load.

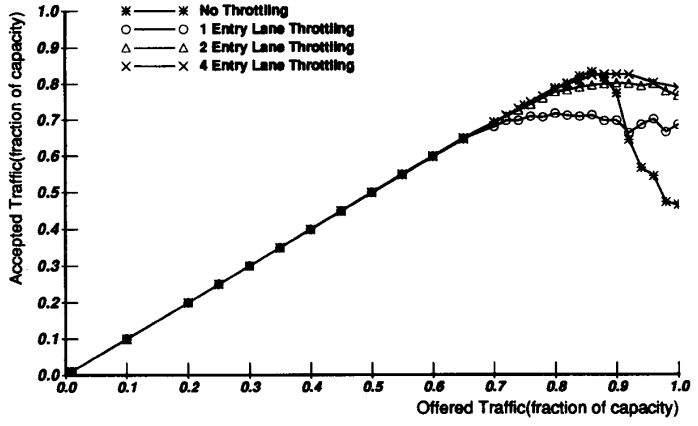

Fig. 6. Throughput as a function of offered traffic for a 16-ary 2-cube network using dynamic adaptive routing with varying degrees of throttling. Throttling reduces peak throughput by restricting the entry of new packets into the network.

TABLE I
THROTTLING REDUCES THE NUMBER OF PACKETS FORCED TO ROUTE ON THE DETERMINISTIC SET OF VIRTUAL CHANNELS. THIS TABLE SHOWS THE THROUGHPUT AND FRACTION OF DETERMINISTICALLY ROUTED PACKETS FOR VARIOUS DEGREES OF THROTTLING UNDER A MAXIMUM LOAD (ALL SOURCES INJECTING EACH CYCLE) OF RANDOM TRAFFIC

Entry Lanes	Throughput	Percent Deterministic
1	.662	0.09
2	.716	1.35
4	.339	13.0
No Throttling	.110	69.1

quickly from a burst of high offered traffic. As soon as the offered traffic drops below 66% capacity the network returns to normal operation. Throttling, however does reduce peak throughput. With a single entry lane peak, throughput is reduced from 86% capacity (no throttling) to 71% capacity because some entry packets are forced to block or turn when they would otherwise be able to make progress. In an unthrottled (and hence unstable) network, however, the peak throughput cannot be sustained since a small variation in traffic would result in network overload and drastically reduced throughput.

Fig. 7 shows the effect of throttling on latency. The figure shows the average packet latency as a function of accepted traffic for a network loaded with random traffic using dynamic adaptive routing. Curves are shown for no throttling and for throttling with 1, 2, and 4 entry channels. The figure shows that throttling increases latency as the curves for throttling approach their lower throughput asymptotes. Most of this added latency is experienced in the queue at the source node as less traffic is allowed into the network.

Figs. 6 and 7 show that throttling slightly degrades latency but stabilizes the network at high traffic rates. Another advantage of throttling is that it reduces the effect of high-traffic sources on low-traffic sources. By restricting the entry of packets from high-traffic sources into the network, throttling reduces congestion, giving the low-traffic sources lower and more predictable latency. The table and figures suggests that throttling with two entry lanes offers a good compromise between peak performance and stability.

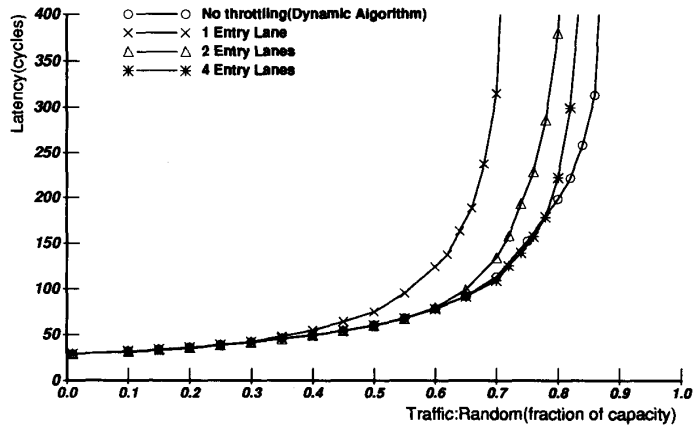

Fig. 7. Latency as a function of accepted traffic for A 16-ary 2-cube network using dynamic adaptive routing with varying degrees of throttling. Throttling increases latency by restricting the entry of new packets into the network.

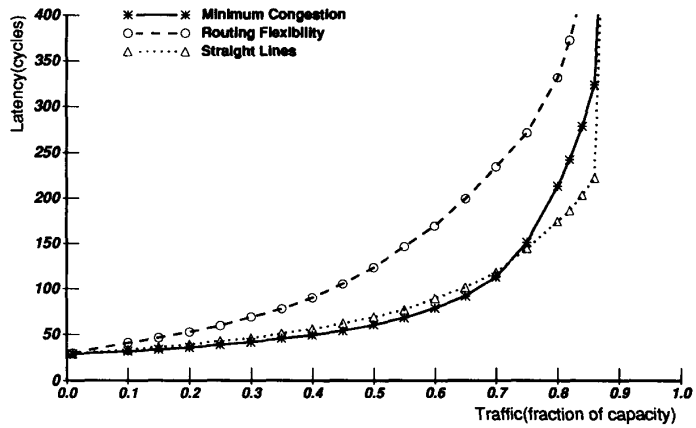

Fig. 8. Latency as a function of offered traffic for three selection functions. Simulations were performed using dynamic adaptive routing and random traffic.

C. Selection Function

Fig. 8 compares the performance of different selection functions in handling nonuniform traffic. The simulation was run with random traffic. The figure shows that minimum-congestion and straight-line selection functions give good performance for this traffic pattern. Straight-line routing gives higher latency than minimum-congestion routing for mid-range traffic levels. This selection function will not begin to adapt until all virtual channels in a given direction are full. Latency is introduced by filling these virtual channels rather than routing over idle physical channels. Maximum-flexibility routing results in much higher latency and saturates at a lower traffic level than the other two functions. The maximum flexibility selection function causes messages to alternate dimensions once a *diagonal* to the destination is reached. This dimension alternation results in high DR numbers and a large number of packets resorting to deterministic routing. Routing along diagonals also results in more uniform loading of switch inputs and hence higher contention [13].

D. Fault Tolerance

Figs. 9 and 10 illustrate the graceful degradation of an adaptive network as channels fail. Fig. 9 shows the latency of the network at 50% capacity for random traffic as a function of the percentage of faulty channels. For each percentage, 20 networks were simulated, each with a different randomly chosen fault set. The asterisks in the figure give the ensemble average latencies. The ends of the vertical error bars represent the 1σ points of each latency distribution. Latency increases only by a factor of 2.3 from a fault-free network to a network with 8% faulty channels (38 faulty channels in a 16-ary 2-cube). In contrast, in a network with deterministic routing, a single faulty channel renders the network inoperable.

Fig. 10 shows network throughput as a function of faulty channels. Network throughput degrades gracefully, dropping from 66% capacity to 54% capacity when 8% of the channels are faulty. This 18% drop in throughput is larger than the 8% that could ideally be achieved but is far better than the 100% drop in throughput that would occur without adaptive routing.

Network throughput falls off faster than the fraction of faulty channels because our algorithms use only local information in making routing decisions. Local routing may waste network capacity by directing many packets into the vicinity of faulty channels where they must then be rerouted around the faults. Traditional global routing methods using routing tables overcome this problem; however they are infeasible in large multicomputer networks because of the table size required. We are investigating the use of hierarchical routing

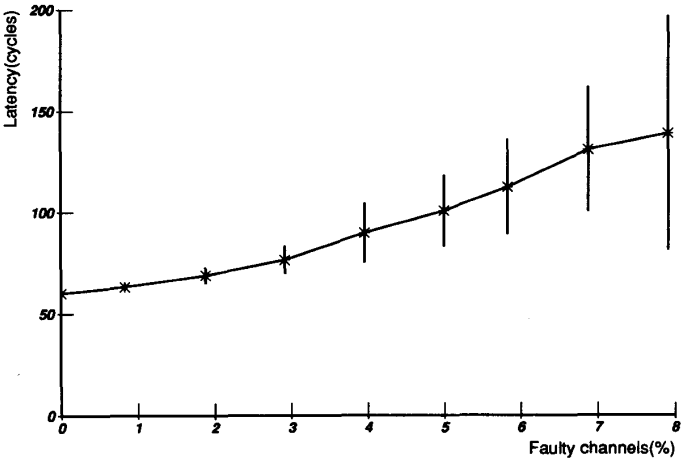

Fig. 9. Latency versus percent faulty channels for a 16-ary 2-cube network operating at 50% capacity with random traffic. Each asterisk gives the mean latency of 20 randomly generated faulty networks. The ends of the vertical error bars represent the 1σ points of each latency distribution.

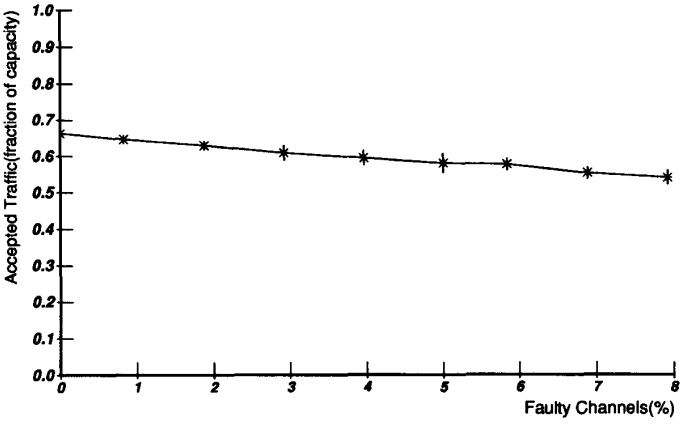

Fig. 10. Throughput versus percent faulty channels for a 16-ary 2-cube network with random traffic and throttling with a single entry lane. Each point gives the mean throughput of 20 randomly generated faulty networks. The error bars show the 1σ points of each distribution.

tables (that store information about regions of the network) as a scalable approach to overcoming the limitations of strictly local routing.

V. Conclusion

Adaptive routing improves the performance and reliability of a multicomputer interconnection network by routing packets around congested or faulty channels. This paper has described two adaptive routing algorithms, proved that they are deadlock free, and evaluated their performance. Both algorithms permit misrouting and avoid deadlock by allocating virtual channels according to the number of dimension reversals a packet has made.

In the static algorithm, there is a fixed mapping between number of dimension reversals and virtual channels. This fixed assignment gives an acyclic channel dependency graph. The dynamic algorithm allows more flexible channel allocation by allocating virtual channels based on occupation to prevent cycles in the packet wait-for graph. Cycles in this graph are eliminated by not allowing a packet to wait on a buffer held by a packet with a lower DR number. In this case, the channel dependency graph is cyclic and adaptive routing is required to avoid deadlock.

These routing algorithms are ideally suited for implementation in a multicomputer network. They use only local information to make routing decisions and can be easily implemented in a small amount of combinational logic. Minimal control storage is required since they do not keep routing tables or global network information. They break deadlock by controlling the allocation of channels (rather than packet buffers [27]) and thus support wormhole routing which requires minimal data storage (one flit per virtual channel). The algorithms are scalable in the sense that their logic and storage requirements per node remain fixed as the size of a machine is increased.[6]

Simulation experiments show that adaptive routing significantly improves throughput for nonuniform traffic patterns but has little effect on performance with random traffic. Adaptive routing improves throughput by a factor of three for bit-reversal traffic in a 16-ary 2-cube network. This traffic pattern causes nonuniform channel loads

[6]Node addresses and the logic to decode them grow logarithmically with the size of the machine; however, a fixed-sized node address (e.g., 32-bits) will handle all currently practical machine sizes.

when dimension-order deterministic routing is employed. Adaptive routing routes around congested channels to balance the load. With random traffic, channels are loaded uniformly, and load balancing is not required.

Throttling is required to stabilize the dynamic algorithm at high traffic rates. Throttling is easily implemented by restricting new packets to route on a small number of virtual channels, *entry lanes*, until their first dimension reversal. Throttling slightly increases latency for uniform loads but reduces the effect of a hot-spot node on the network latency seen by other nodes. Throttling also reduces the fraction of messages that are forced by resource constraints to resort to deterministic routing. Simulations suggest that throttling with two entry lanes effectively stabilizes the network with only a small affect on latency.

The adaptive routing algorithms presented here can be used in conjunction with many different selection functions. Simulations show that minimum-congestion and straight-line selection functions give good performance. The maximum-flexibility selection function results in higher latency and lower throughput because it forces packets onto network diagonals.

The performance of networks using adaptive routing gracefully degrades as channels fail. Experiments show that with 8% of the channels faulty, latency increases by a factor of 2.3 and throughput is reduced to 81% of its normal level.

With virtual channel flow control [15] and adaptive routing, multicomputer networks achieve performance approaching 90% of their physical capacity. This performance is affected little by nonuniform traffic patterns and degrades gracefully with channel failures.

The use of adaptive routing and virtual channels motivates the use of synchronous router design. Many early routers were asynchronous or self-timed to achieve maximum performance [17], [19], [22]. With deterministic routing, the design of such routers was straightforward, as each dimension could operate independently and no concept of global time was required. To make use of virtual channels, however, the router must maintain timers to distinguish between a blocked channel and one that is waiting for an acknowledgment. In a synchronous router, such timing is implicit. Adaptive routing requires that information from many output channels be collected together to make a routing decision. In an asynchronous router, collecting this information poses a high synchronization overhead.

The application of these high-performance networks extends beyond connecting the processing nodes of multicomputers. Low-dimensional k-ary n-cube networks can also be used as data switches in a local-area or long-haul network and as a general-purpose backplane to connect components of digital systems. They offer a scalable alternative to buses for general-purpose interconnection in digital systems.

ACKNOWLEDGMENT

We thank the members of the MIT Concurrent VLSI Architecture group for their help with and contributions to this paper. Special thanks go to E. Spertus and L. Sardegna for proofreading this manuscript, to J. Keen and D. Wallach for assistance with the network simulations, and to R. Lethin for suggesting the use of hierarchical routing tables. H. Aoki thanks his wife Kuniko for her support during this work.

REFERENCES

[1] J. K. Annot and R. A. H. van Twist, "A novel deadlock free and starvation free packet switching communication processor," in *Parallel Architectures and Languages Europe 1987,* 1987, pp. 68–85.

[2] W. C. Athas and C. L. Seitz, "Multicomputers: Message-passing concurrent computers," *IEEE Comput. Mag.,* vol. 21, no. 8, pp. 9–24, Aug. 1988.

[3] H. G. Badr and S. Podar, "An optimal shortest-path routing policy for network computers with regular mesh-connected topologies," *IEEE Trans. Comput.,* vol. 38, no. 10, pp. 1362–1371, Oct. 1989.

[4] P. R. Bell and K. Jabbour, "Review of point-to-point network routing alorithms," *IEEE Commun. Mag.,* vol. 24, no. 1, pp. 34–38, Jan. 1986.

[5] D. Bertsekas and R. Gallager, *Data Networks.* Englewood Cliffs, NJ: Prentice-Hall, 1987.

[6] S. Borkar *et al.*, "iWARP: An integrated solution to high-speed parallel computing," in *Proc. Supercomput. Conf.,* IEEE, Nov. 1988, pp. 330–338.

[7] M.-S. Chen and K. G. Shin, "Adaptive fault-tolerant routing in hypercube multicomputers," *IEEE Trans. Comput.,* vol. 39, no. 12, pp. 1406–1416, Dec. 1990.

[8] ——, "Depth-first search approach for fault-tolerant routing in hypercube multicomputers," *IEEE Trans. Parallel Distributed Syst.,* vol. 1, no. 2, pp. 152–159, Apr. 1990.

[9] A. Andai Chien, "Congestion control in routing networks," Master's thesis, Massachusetts Instit. Technol., Cambridge, MA, Oct. 1986.

[10] E. Chow, H. Madan, and J. Peterson, "A real-time adaptive message routing network for the hypercube computer," in *Proc. 8th Real Time Syst. Symp.,* IEEE, 1987, pp. 88–96.

[11] Ametek Corp., Ametek 2010 Product Announcement, 1987.

[12] W. J. Dally, "Fine-grain message passing concurrent computers," in *Proc. Third Conf. Hypercube Concurrent Comput.,* vol. 1, Pasadena, CA, Jan. 1988, pp. 2–12. VLSI memo 88-454.

[13] ——, "Network and processor architecture for message-driven computers," in *VLSI and Parallel Computation,* Suaya and Birtwhistle, Eds. Palo, Alto, CA: Morgan Kaufmann, 1990.

[14] ——, "Performance analysis of k-ary n-cube interconnection networks," *IEEE Trans. Comput.,* vol. 39, no. 6, June 1990. Also appears as a chapter in *Artificial Intelligence at MIT, Expanding Frontiers,* edited by P. H. Winston, with S. A. Shellard, MIT Press, 1990, vol. 1, pp. 548–581.

[15] ——, "Virtual-channel flow control," in *Proc. 17th Annu. Int. Symp. Comput. Architecture,* May 1990, pp. 60–68.

[16] W. J. Dally *et al.*, "The J-Machine: A fine-grain concurrent computer," in *Proc. IFIP Congress,* G. X. Ritter, Ed., North-Holland, Aug. 1989, pp. 1147–1153.

[17] W. J. Dally and C. L. Seitz, "The torus routing chip," *Distributed Comput.,* vol. 1, pp. 187–196, 1986.

[18] ——, "Deadlock free message routing in multiprocessor interconnection networks," *IEEE Trans. Comput.,* vol. C-36, no. 5, pp. 547–553, May 1987.

[19] W. J. Dally and P. Song, "Design of a self-timed VLSI multicomputer communication controller," in *Proc. Int. Conf. Comput. Design,* IEEE, Computer Society Press, Oct. 1987, pp. 230–234.

[20] J. Duato, "On the design of deadlock-free adaptive routing algorithms for multicomputers: Design methodologies," in *Parallel Architectures and Languages Europe,* 1991.

[21] —— "On the design of deadlock-free adaptive routing algorithms for multicomputers: Theoretical aspects," in *Proc. 2nd European Distributed Memory Comput. Conf.,* 1991.

[22] C. M. Flaig, "VLSI mesh routing systems," Master's thesis, California Instit. Technol., 1987.

[23] R. M. Fujimoto, "VLSI communication components for multicomputer networks," Tech. Rep. UCB/CSD/83/137, Comput. Sci. Div., Univ. California at Berkeley, Berkeley, CA 94720, Sept. 1983.

[24] R. G. Gallager, "A minimum delay routing algorithm using distributed computation," *IEEE Trans. Commun.,* vol. COM-25, no. 1, pp. 73–85, Jan. 1977.

[25] D. Gelernter, "A DAG-based algorithm for prevention of store-and-forward deadlock in packet networks," *IEEE Trans. Comput.,* vol. C-30, no. 10, pp. 709–715, Oct. 1981.

[26] D. C. Grunwald, "Circuit switched multicomputer and heuristic load placement," Tech. Rep. UIUCDCS-R-89-1514, Dep. Comput. Sci., Univ. Illinois, Dep. Comput. Sci., Univ. Illinois at Urbana–Champaign Urbana, IL 61801-2987, Sept. 1989.

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

[28] BBN Advanced Computers Inc., "Butterfly parallel processor overview," BBN Rep. 6148, Mar. 1986.

[29] P. Kermani and L. Kleinrock, "Virtual cut through: A new computer communication switching technique," *Comput. Networks,* vol. 3, pp.

267–286, 1979.
[30] C.-k. Kim and D. A. Reed, "Adaptive packet routing in a hypercube," in *Proc. Third Conf. Hypercube Concurrent Comput. Appl.,* ACM Press, Jan. 1988, pp. 625–630.
[31] S. Konstantinidou, "Adaptive minimal routing in hypercubes," in *Proc. 6th MIT Conf. Advanced Res. VLSI,* 1990, pp. 139–153.
[32] S. Konstantinidou and L. Snyder, "Chaos router: A practical application of randomization in network routing," in *Proc. Symp. Parallel Architectures Algorithms,* 1990, pp. 21–30.
[33] ——, "Chaos router: Architecture and performance," in *Proc. 18th Annu. Symp. Comput. Architecture,* 1991, pp. 212–221.
[34] J. N. Mailhot, "Routing and flow control strategies in multiprocessor networks," S.B. thesis, May 1988.
[35] P. M. Merlin and P. J. Schweitzer, "Deadlock avoidance-store-and-forward deadlock," *IEEE Trans. Commun.,* vol. COM-28, no. 3, pp. 345–354, Mar. 1980.
[36] W. G. P. Mooij and A. Ligtenberg, "Architecture of a communication network processor," in *Proc. Parallel Architectures and Languages Europe 1989,* Springer-Verlag, 1989, pp. 238–250.
[37] D. Nassimi and S. Sahni, "An optimal routing algorithm for mesh-connected parallel computers," *J. ACM,* vol. 27, no. 1, pp. 6–29, Jan. 1980.
[38] ——, "Optimal BPC permutations on a cube connected SIMD computer," *IEEE Trans. Comput.,* vol. C-31, no. 4, pp. 338–341, Apr. 1982.
[39] J. Y. Ngai, "A framework for adaptive routing in multicomputer networks," Ph.D. dissertation, Caltech, 1989.
[40] J. Y. Ngai and C. L. Seitz, "A framework for adaptive routing in multicomputer networks," in *Proc. ACM Symp. Parallel Algorithms and Architectures,* 1989.
[41] D. A. Reed and R. M. Fujimoto, *Multicomputer Networks: Message-Based Parallel Processing.* Cambridge, MA: MIT Press, 1987.
[42] C. L. Seitz, "The Cosmic cube," *Commun. ACM,* vol. 28, no. 1, pp. 22–33, Jan. 1985.
[43] A. S. Tanenbaum, *Computer Networks,* second ed. Englewood Cliffs, NJ: Prentice-Hall, 1988.
[44] L. G. Valiant, "A scheme for fast parallel communication," *SIAM J. Comput.,* vol. 11, no. 2, pp. 350–361, May 1982.
[45] ——, "Optimality of a two-phase strategy for routing in interconnection networks," *IEEE Trans. Comput.,* vol. C-32, no. 9, pp. 861–863, Sept. 1983.

Optimal Resilient Distributed Algorithms for Ring Election

M. Y. Chan and F. Y. L. Chin

***Abstract*— This paper considers the problem of electing a leader in a dynamic ring in which processors are permitted to fail and recover during election. $\theta(n \log n + k_r)$ messages, when counting only messages sent by functional processors, are shown to be necessary and sufficient for dynamic ring election, where k_r is the number of processor recoveries experienced.**

***Index Terms*— Distributed election, processor failures and recoveries, unidirectional rings.**

Manuscript received June 26, 1989; revised June 20, 1991 and August 2, 1992.

The authors are with the Department of Computer Science, The University of Hong Kong, Hong Kong.

IEEE Log Number 9206276.

I. Introduction

One of the most studied problems in the area of distributed algorithms is distributed leader election. Many papers have been written about distributed leader election especially on rings. To cite just a few references, consider [12], [3], [2], [9], [10], [4], [6], [15], [16], [7], [11], [17], [18], [1], [14], [8]. All of these papers deal with the static ring, with the exception of Goldreich and Shrira's treatment of election in rings with *communication link failures* [8]. The problem of considering rings with *processor failures and recoveries* during election provides a complement to [8], and was first suggested by Filman and Friedman [5] as a problem worthy of research. One of the main assumptions of this problem is that, when a processor leaves the ring (fails), the ring is patched around its place. This property allows for some rather interesting solutions.

This paper considers the problem of electing a leader in a dynamic ring in which processors are permitted to fail and recover during election. $\theta(n \log n + k_r)$ messages, when counting only messages sent by functional processors, are shown to be necessary and sufficient for dynamic ring election, where k_r is the number of processor recoveries experienced.

II. The Model for Dynamic Rings

The objective is to devise an algorithm, to be run on each processor, which will distinguish one of the functional processors as leader. We outline in greater detail the assumptions of our model:

1) We consider a system of n independent processors arranged and connected in a circular fashion by n point-to-point *unidirectional* communication links. Initially, they are all in the "sleep" state (Fig. 1).
2) Each processor is distinguished by a unique identification number. Furthermore, we assume that only comparisons of identity numbers can be made, and the algorithm is not aware of the domain or range to which identities belong.
3) As assumed in [3], [4], [7], and [9], processors may start, or "wakeup" to, the algorithm, i.e., get into the "active/relay" state (Fig. 1), either spontaneously at any arbitrary time of its own free will, or upon receipt of a message of the algorithm. Election begins when at least one processor awakens spontaneously.
4) The network is also assumed to provide both "sequential" and "guaranteed" communications, meaning messages sent across a link will be eventually received, and received in the order sent and received as sent. In other words, communication is reliable and only processors are faulty.
5) When processors fail, they get into the "failed" state (Fig. 1). We consider only "clean" failures, i.e., "failed" processors simply stop participating in the election protocol and do not behave maliciously. In fact, the "clean" failures also imply that the ring structure would not be disrupted by processor faults as messages simply pass through or around "failed" processors. This assumption is now common for ring networks and is made possible by providing a bypass switch for each processor [13].
6) For added flexibility we assume that processors may fail ("active/relay" state → "failed" state), or recover after failing ("failed" state → "sleep" state), at any time, as long as eventually there is at least one functional processor in the ring. And there are no limits to the number of times a processor may fail and then recover during election. Thus, the number of