SANDIA REPORT
SAND2008-0068
Unlimited Release
Printed January 2008

A Survey of Routing Techniques in Store-and-Forward and Wormhole Interconnects

David M. Holman and David S. Lee

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Sandia National Laboratories

SAND2008-0068
Unlimited Release
Printed January 2008

A Survey of Routing Techniques in Store-and-Forward and Wormhole Interconnects

David Michael Holman
Student Intern, Embedded Sensor Systems, 2623

David S. Lee
Wireless and Event Sensing Applications, 2664

Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-0530

Abstract

This paper presents an overview of algorithms for directing messages through networks of varying topology. These are commonly referred to as routing algorithms in the literature that is presented. In addition to providing background on networking terminology and router basics, the paper explains the issues of deadlock and livelock as they apply to routing. After this, there is a discussion of routing algorithms for both store-and-forward and wormhole-switched networks. The paper covers both algorithms that do and do not adapt to conditions in the network. Techniques targeting structured as well as irregular topologies are discussed. Following this, strategies for routing in the presence of faulty nodes and links in the network are described.

CONTENTS

LIST OF FIGURES

EXECUTIVE SUMMARY

Networked node-based architectures are being examined for use in modern satellite payloads. These networks require design attention to fundamental network concepts, one of which is routing. Routing is the process that determines the path of a packet through a network. As satellite missions have limited resources and often time-critical objectives, deterministic and reliable delivery of data is paramount. Poor routing causes movement of data across non-optimal paths, increasing power utilization, and potentially causing loss of data. Conversely, good routing will optimize data flow and reduce latency. This study presents methods used to develop routing algorithms, as well as some inherent dangers of routing and methods used to mitigate the effects of failures of network elements.

Routing algorithms are typically classified using a number of different criteria. For the purpose of this study, the important factors to distinguish are whether algorithms are adaptive or deterministic, and if they are minimal or non-minimal. Adaptive algorithms make routing decisions at runtime, while deterministic algorithms determine all packet routes prior to runtime. Minimal algorithms require that packets always end up closer to their destination after they traverse a link. Non-minimal algorithms do not impose this “positive-progress” requirement and may route packets in any direction.

Another important classification involves the switching technology utilized in the system. This study focuses on two popular switching technologies: store-and-forward and wormhole switching. Store-and-forward switching first stores incoming packets into memory. Once a packet is entirely received, the switch examines the header of the packet and queues the packet to send via the appropriate output port. Wormhole-based switching is quite different and operates by dividing a packet into blocks (called flits) and sends them as an uninterruptible stream through the network. When a router receives a header flit, it immediately examines it and re-transmits the flit out the appropriate port. Subsequent flits are sent as soon as they are received until the packet is complete, and no other packets may utilize the node until the packet is complete. If the next router is busy, the flit must stall, as will all trailing flits in their respective routers.

Deadlock is arguably the most common and complex pitfall in routing algorithms. A network experiences deadlock when routers in the network are stalled in a cyclic fashion such that no router can transmit. An example would be where router A is attempting to send to B, B is attempting to send to C, and C is attempting to send to A. Livelock is another issue that occurs when a packet is continually routed, but never reaches its destination. Since minimal algorithms ensure that progress is made at each step of the routing process, livelock will only occur in non-minimal routing algorithms.

The discussion of routing algorithms begins with deterministic algorithms. Deterministic store-and-forward algorithms obtain information of the network prior to determining routes. For example, a shortest-path algorithm uses topology data and cost information for links and utilizes Dijkstra’s algorithm to determine routes prior to runtime. For wormhole networks, Zakrevski et al. [1] demonstrates a technique that imposes a logical structure onto topologies through spanning trees. The structure allows only certain paths to be utilized for routing, which eliminates the possibility of deadlock.

Adaptive algorithms introduce some additional difficulties. Since decisions are made at runtime, two packets sent to the same destination may take different paths and arrive out of order. Despite such complications, adaptive algorithms are often more robust than deterministic algorithms. For example, some algorithms can manage live topology changes for hot-pluggable nodes. Others provide for congestion control. One example of this is a store-and-forward algorithm utilizing shortest-path routing until congestion is encountered. At this point the algorithm chooses an alternate "escape path" in an attempt to route the packet down a different, less congested path. Wormhole routers can also similarly avoid congestion by generating virtual channels for use as escape paths [2].

One quality often desired in network architectures is fault tolerance. For fault tolerance, a routing algorithm must route traffic around failed portions of the network. Simply using adaptive algorithms is not a good solution as many are not "fully adaptive" and have measures (such as conservative deadlock protection) that disallow certain paths from being taken that would be needed for fault tolerance. Also, failures that bisect the network change the network topology, causing deadlock with some adaptive algorithms.

While fault-tolerant routing is difficult, a few adaptive, non-minimal algorithms do exist. Typically these algorithms also contain measures to prevent livelock. One example is backtracking [3], which routes using shortest paths until a fault is encountered. At that point, the algorithm first attempts to route down an alternate minimal path, then down a non-minimal path. If that still fails, the packet backtracks to the previous node and tries to find a new route from there. This is one simple example of how routing algorithms may cope with failures in the network.

Ultimately, there are a number of factors that need to be taken into account when selecting a routing algorithm. There is an abundance of routing algorithms already in use, and selecting one involves choosing an algorithm that accomplishes design goals without compromising any desired features, such as fault tolerance. As such, the ideas, techniques, and pitfalls described in this paper provide a number of considerations for selecting a routing algorithm for system development.

1 INTRODUCTION

This paper provides the results of a study into current methods for routing in networks of varying topology and switching methods. Routing as described in this paper is the process by which a message in a network determines its path. There are many issues that a quality routing algorithm has to address in addition to making sure the packet arrives at the correct destination. The issues this paper addresses are deadlock, livelock, and faults in the network.

The remainder of the paper is divided into six sections. First, information is presented that is required to understand the rest of the paper. This includes definitions of key terms that are used throughout the paper, as well as an overview of modern switching techniques. Next, the issue of deadlock in computer networks is explained and methods are presented that can prevent deadlock in different types of networks. The following section includes a brief description of livelock in computer networks and its relevance to the routing techniques that are presented in the paper. Next, routing techniques are presented. This section is divided into to main subsections. One section presents fixed routing techniques that do not adapt to conditions in the network and the other section contains techniques that can use local information to adjust their behavior. The second to last major section then expands on the routing section to show how routing algorithms can be used to deal with faults in networks. The paper then concludes with a summary and list of references that were cited in the paper.

2 BACKGROUND

This section provides definitions of terms used both in routing literature and throughout the paper. It also contains an overview of several switching techniques and router details that are relevant to the design and implementation of routing algorithms.

2.1 Definitions

Adaptive routing refers to algorithms where decisions are made at runtime in routers using local information to determine the next hop of a given packet. It is easy to see that if two consecutive packets are allowed to follow separate paths through the network, then their order of arrival at the destination could be non-deterministic. A *deterministic* algorithm, on the other hand, routes based only on information that is available before the algorithm begins; this means that all packet routes are determined prior to runtime and each packet has only one path per source-destination pair [4, p. 141]. A *quasi-static* routing algorithm is similar to a deterministic algorithm, except that it will regenerate the fixed parameters it uses to route periodically. This gives the algorithm some adaptability to conditions in the network [5]. *Minimal* routing algorithms never allow packets to be routed away from the destination. *Non-minimal* routers can allow hops in any direction [4, p. 141]. *Deadlock* is used to refer to a state of the network where packets cannot make progress in the network due to a cyclic wait dependency. *Livelock* refers to the situation where a packet is injected into the network and is constantly routed, but never reaches its destination [4, p. 83]. In this paper the term *switching* will be used to indicate the mechanism that a router uses to move a packet from its input to output ports. *Routing*, on the other hand, will indicate the process of selecting a path that a packet will follow through the network. I make this distinction to prevent confusing the reader since the terms seem to be used in quite a few ways in routing literature.

2.2 Key Switching Techniques

As will be discussed throughout this paper, the switching technique that is implemented in the router hardware can greatly affect which routing strategies will work correctly on the router. In this paper, the focus will be on routing strategies that are appropriate for *store-and-forward* switching and *wormhole* switching as they are implemented by RapidIO and Spacewire hardware, respectively [6, 7].

A store-and-forward router simply reads data from its input ports into one (or possibly several) internal buffer(s). After a packet has been completely read from an input port, the router saves the packet in a queue. The router simultaneously reads a packet from the head of the queue and sends it out of the appropriate output port. This technique is relatively simple to implement, but packet delay accrues at each hop in a network since the packet must wait to be saved into the internal queue before being forwarded [8].

Wormhole routing is a circuit-oriented technique where the packet is broken up into equally sized blocks called *flits*. The packet is then sent as an uninterruptible stream of flits through the network. When the leading flit, called the *header*, reaches a node it is forwarded to the next hop, and no other packet in the network can use that node until all of the trailing flits have been through that router. If the header cannot be routed because the next hop is busy, then the header waits in that router and all of the trailing flits also wait in their respective routers, creating a path of temporarily unusable routers in the network [8].

The diagram in Figure 1, borrowed from [8], illustrates graphically the difference in latency between wormhole and store-and-forward switching. The pipelined forwarding of the wormhole routers causes the delay at each router to effectively be only the time required to read and forward the header. The delay in the store-and-forward router, on the other hand, is the time required to read and forward the entire packet. The pipelined forwarding of the wormhole router also allows great overlap between routers, resulting in much of the time spent forwarding the data being hidden. Store-and-forward, on the other hand, leaves routers idle as a packet traverses the network.

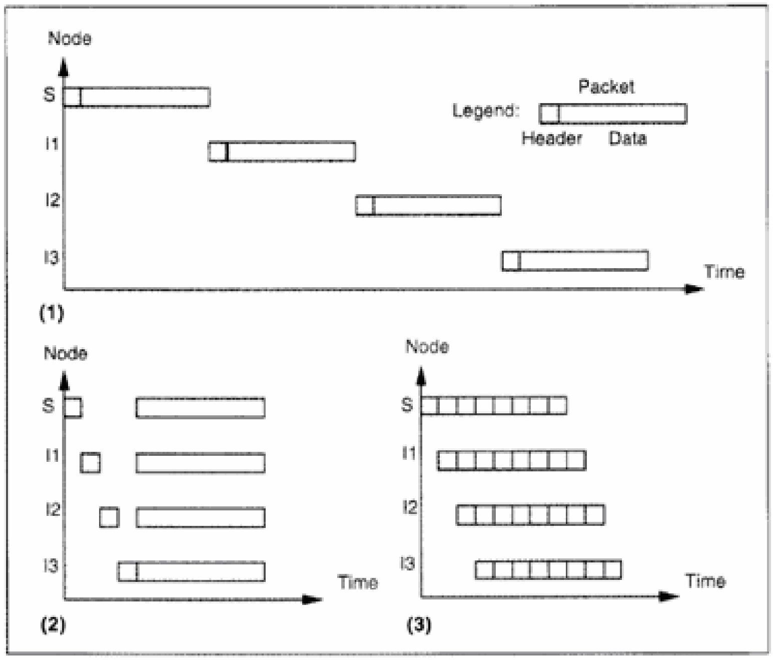

Figure 1. (1) Store-and-Forward; (2) Circuit-Switched; (3) Wormhole-Switched [8]

2.3 Other Switching Techniques

Virtual cut-through is an extension to store-and-forward. This technique also reads buffers packets into a buffer, but only when packets arrive while the router is busy. If the router is idle and a packet arrives, then the router begins forwarding the packet as soon as it has read the header [9]. This technique is favorable because it decreases routing latency when there is low congestion but simply implements store-and-forward routing in the presence of congestion, making it amenable to routing strategies that are designed for store-and-forward [10].

There exist some switching techniques that do not use buffers in the router or otherwise rely heavily on misrouting packets. The more common techniques that I came across are mad-postman [11], deflection routing [12], and chaotic routing [13]. They are typically implemented at a very low level, making them less interesting for a high-level routing study than the other techniques described in this paper. Key references are included on these topics.

2.4 Internal Queues

Where packets (or flits) are stored in a node is an important consideration for the implementation of a routing algorithm. Having a single central queue for packets is simpler and can potentially require less total memory. It, however, can present a bottleneck when the packet at the head of the queue is blocked.

One key advantage that wormhole routing has over other methods that makes it amenable to very large scale integration (VLSI) implementation is its small memory requirements. Since the router only has to buffer a single flit per channel, much less space in the router has to be dedicated to memory. If area considerations are not important, then virtual cut-through routing has been shown to produce greater throughput in a network than wormhole routing [10].

2.5 Issues/Design Considerations

There are some caveats that should be mentioned regarding these routing techniques. First, wormhole routing is described originally in a patent [14] belonging to the California Institute of Technology. It is unclear whether this restricts the use of wormhole routing in computer systems as the technique seems to be used quite pervasively. Another issue is that each paper presented here usually describes its own model of a router. While they are usually quite similar and reasonable, care should be taken to make sure that the router model does not make assumptions that cannot be kept in a higher-level implementation. One especially common variable is whether packets in a router are stored in a central queue or in edge buffers. Another aspect that seems to vary between papers is whether mathematical representations of networks use directed or undirected graphs.

3 DEADLOCK

The potential for deadlock arises in networks when cyclic buffer dependencies develop from the topology and routing algorithm of the network. As buffers fill, packets begin waiting. If a cycle of waiting packets develops, then no progress can be made and the packets wait forever. Preventing deadlock is the focus of the papers outlined in the following subsections. First, strategies for avoiding deadlock in routers using store-and-forward routing are examined. Next, different strategies for avoiding deadlock in wormhole-routed networks are presented.

3.1 Store-and-Forward

This section begins with a trivial example of how deadlock can occur in store-and-forward networks. The following section contains some theoretical results on deadlock in store-and-forward networks. After that, the paper presents results from a paper showing that deadlock-avoidant algorithms can be implemented even with cyclic buffer dependencies. The final section describes techniques to avoid deadlock by providing a monotonic ordering of the buffers to remove cyclic dependencies.

3.1.1 Deadlock Example

Figure 2 shows a very simple network topology. Suppose that each node has a buffer that can hold a single packet and the buffer is full with a packet destined for a different node. At each node, we see that the router is waiting for the buffer to become free in its neighboring router so that it can send its packet. Since each router is waiting for the other, no progress is ever made and the system is deadlocked [15].

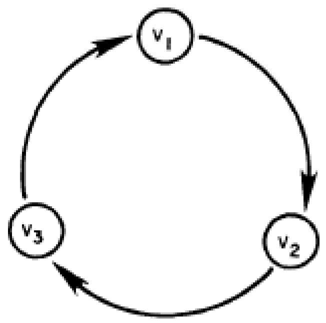

Figure 2. A simple deadlock example [15].

3.1.2 Deadlock Theory

Toueg and Steiglitz present a set of simple theoretical results in [15] for store-and-forward networks. Here, they show that determining whether a set of routes on a given network topology is vulnerable to deadlock can be done in O(|E|) time since it is equivalent to finding a cycle in the routing graph. Determining whether a deadlocked state can actually be reached can also be done with an algorithm described in the paper in O(|E|) time. They also show that one can also determine if there exists a set of routes in a network that is deadlock-free, but this problem is Non-deterministic Polynomial time (NP) complete.

Duato extends the ideas presented in Toueg and Steiglitz's paper by proving a necessary and sufficient condition for deadlock-freedom in networks that utilize adaptive routing [10]. The main idea is that since packets choose from multiple routes at each router, they have the ability to route around buffers that are full. This means that a cyclic buffer dependency in itself does not guarantee that deadlock will occur.

Duato's proof relies on the concept of an *escape path*, which is a path the network packets can use that is guaranteed to be deadlock-free. The dependencies in the escape path are formalized with the idea of an extended buffer dependency graph. This graph is generated by making each buffer in a network a node and adding edges between each pair of nodes that have a *direct* or *direct-cross dependency*. Next, if a sub-graph of the extended buffer-dependency graph can be found that is connected to all of the nodes and is acyclic, then the network is deadlock-free. The proof is relatively complicated, but the key point is that having an acyclic extended buffer dependency graph allows a monotonic ordering of the buffers in the escape path to be created. Duato then uses these ideas to show that a sufficient condition for deadlock free routing R is a deadlock-free routing sub-function R_1 if R_1 can be constructed using a subset of the virtual channels or buffers of R with their associated next-hop sets.

This gives way to a relatively straightforward design methodology for deadlock-free adaptive routing algorithms in store-and-forward networks. Simply begin with a known deadlock-free algorithm, which can be static or dynamic (algorithms that are guaranteed to be deadlock-free are presented later in the paper). At each node where additional alternative paths should be available, add a virtual channel, and allow routing to the extra paths on the new channel. This maintains the deadlock-freedom of the original algorithm and adds new adaptivity.

Schwiebert extends this idea to networks using any routing technique that make routing decisions based only on local information in [16]. Cypher and Gravano present an interesting piece of deadlock-avoidance theory in [17] that proves any deadlock-free adaptive routing strategy for store-and-forward or virtual cut-through switching can be reduced to a deadlock-free deterministic algorithm.

3.1.3 Buffer Ordering

Creating deadlock-free routers for arbitrary topologies without performing analysis ahead of time is shown in [18] by Toueg and Ullman. They present the *Forward-Count Controller*, which only accepts packets that have fewer hops to go than there are free buffer slots in the controller. This technique requires a bounded path length but appears to be suitable for deterministic or adaptive routing.

Gunther explains a technique using buffer ordering for preventing deadlock in store-and-forward (SAF) networks in [19]. The key idea is to have a monotonic ordering assigned to buffers that packets must follow. Gunther describes a descending buffer number intuitively as causing packets to continuously “drain” from the network. Gopal presents one of the more practical versions of this technique in [20]. The technique employs graph coloring to select buffer numbers for each node. From the network topology, each node is assigned a number that is different from its neighbors’ numbers. When a packet is injected into the network, it begins in buffer 0. Each time it is routed to a node with a lower graph number, its buffer number is increased. This idea is illustrated in Figure 3. The number of buffers required is shown to be bounded by the degree of the network; specifically, $B \leq \left\lceil \frac{D}{D+1} * H \right\rceil + 1$ where B is the maximum number of required buffers, D is the maximum degree of the network, and H is the maximum number of hops in the network.

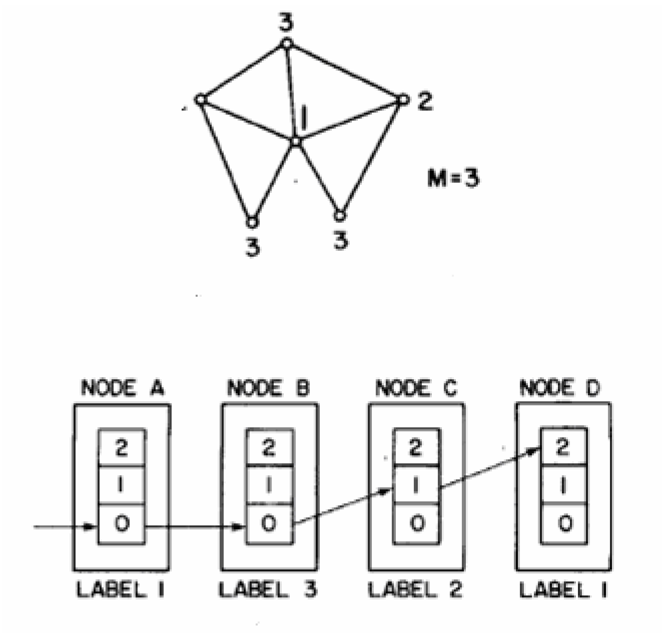

Figure 3. A graph that has been colored and a packet being routed through buffers of the network [20].

3.2 Wormhole

When a header flit is blocked, all of the nodes that are occupied by trailing flits will become unusable until the header is allowed to make progress. This increases the likelihood of stalling other headers that the original header may be waiting on, which makes deadlock more likely. There have been many papers published on providing deadlock-free wormhole routing.

The first key paper in the area was by Dally and Seitz (inventors of the wormhole-switching method) [14] [21]. They describe a system capable of routing statically in any q-ary n-cube without deadlock. In a method similar to buffer ordering, they provide multiple buffers in each router and multiplex sending time on the physical communication links to provide *virtual channels*. By providing a monotonic ordering of these virtual channels, they create a static routing algorithm where cyclic buffer dependencies do not exist. They also show that having this acyclic set of buffer dependencies is a necessary and sufficient condition for deadlock avoidance when routing statically with wormhole-switched networks.

Upadhaya et al. provide an example of applying Dally and Seitz's underlying idea to existing routing algorithms in [22]. In this paper, they provide a method for converting deadlock-free routing algorithms for meshes into deadlock-free routing algorithms for tori by using virtual channels to remove the cyclic dependencies created by the wraparound links on tori.

Duato formalizes the requirements on the escape path for wormhole routing in [2]. Unlike in store-and-forward networks, once the header flit leaves a router, the router is not free to route additional headers until every flit in the message has passed through the router. This means that if a header flit routes through another message's escape path, it may continue to block that header until it reaches its destination. Duato's theorems capture this increased dependence between messages by defining two new types of dependencies in the extended buffer dependency graph called *indirect dependencies* and *indirect cross dependencies*. These dependencies represent the waiting that can occur when headers hold parts of the escape path while they block. This allows a routing sub-function with an acyclic extended buffer dependency graph to again be a necessary and sufficient condition for deadlock-free in adaptive algorithms.

Some form of providing monotonic ordering on virtual channels or acyclic escape routes is present in most published wormhole routing algorithms. How specific routing algorithms are implemented will be discussed in the next section.

3.3 Issues/Design Considerations

One issue to consider for providing a practical, deadlock-free system is the complexity of the deadlock-avoidance technique used. Store-and-forward networks have a clear advantage in this regard. When routing statically, the two techniques that appear simplest to implement are using the forward-count router from Toueg's paper [18] or providing an ordering on the buffers using Gopal's graph coloring technique [20].

While both of these techniques will work with adaptive routing as well, adding adaptivity in the case of store-and-forward switching actually makes deadlock avoidance simpler. In fact, a deadlock-free sub-function is all that is required to prevent deadlock [10]. If deterministic routing is strongly preferred, then the routes can also be checked statically for deadlock using Toueg and Steiglitz's methods [15], which may show that buffer ordering is unnecessary.

Overall, deadlock avoidance for wormhole routing is more complicated. Static routing algorithms generally require careful buffer ordering as in [21], and there is less flexibility in designing deadlock-free adaptive routing algorithms [2].

4 LIVELOCK

The following subsections present the theory of livelock and practical techniques to avoid creating livelock when routing.

4.1 Theory and Techniques

Livelock occurs when a packet is continually routed, but never reaches its destination. Since minimal algorithms ensure that progress is made at each step of the routing process, it is clear that livelock is impossible in a minimal routing algorithm; however, livelock does become possible when non-minimal routing is allowed [23].

Several techniques exist to deal with livelock when allowing non-minimal routing paths. One method is misrouting randomly [13], which provides a probabilistic guarantee against livelock. Ngai and Seitz illustrate two other techniques for preventing livelock in [23]. One method assigns precedence to packets according to how close they are to their destinations. This guarantees system-wide progress, but does not ensure livelock-freedom for an individual packet. To guarantee that every packet is livelock-free, they instead assign precedence based on the length of time that a packet has been in the network. Thus, as a packet becomes older, it is more likely to be selected to be routed on its preferred channel.

4.2 Issues/Design Considerations

Livelock is generally not an issue in the routing algorithms presented in the next section because they are largely minimal routing algorithms. When attempting to correct faults, it is much more likely for the routing algorithm to need to misroute packets, increasing the risk of livelock. The algorithms that are reliant on misrouting typically have guarantees to prevent livelock built into the algorithm, so verifying that all packets will eventually arrive should not be an issue. It is, however, important to be aware of the possibility when using non-minimal routing algorithms, in case they are modified to suit a certain application.

5 ROUTING

This section is split into two main subsections. First, deterministic routing techniques are presented. Next, adaptive routing techniques are described. Both subsections are split further into sections based on the underlying switching technique that the routing algorithm is intended to run above. The section closes with a discussion of implementation issues.

5.1 Deterministic Routing

Deterministic routing in this paper refers to techniques where the routes taken by packets are determined solely by their starting and ending nodes. This allows all packets to arrive at their destinations without the possibility of livelock or change of order.

5.1.1 Wormhole

Dally and Seitz created a routing algorithm capable of routing in q-ary n-cubes that was discussed previously. This is commonly referred to as “dimension-ordered” routing because the algorithm makes adjustments along strictly increasing or decreasing dimensions of the network until the source address has been transformed into the destination address [21].

Zakrevski et al. adapt a strategy (originally described in [24]) for adaptive routing in regular topologies to arbitrary networks to prevent deadlocks when using wormhole switching in [1]. While the original algorithm relies on the structure of the topology to determine which moves must be prohibited, Zakrevski et al. impose a structure on arbitrary topologies through spanning trees. They can then determine which turns are safe and which need to be removed to prevent deadlock. The method requires that no more than one third of the total turns be prohibited. The running time of the algorithm is O(N^3) where N is the number of nodes in the network, but it only has to be run when the topology of the network changes, so it is a relatively inexpensive algorithm.

5.1.2 Store-and-Forward

Shortest Path First (SPF) is a routing algorithm developed for the ARPANET [25]. It is typically implemented as what Gallager would call *quasi-static* [5], which means that the algorithm provides a means to adapt to changes in the network periodically rather than while routing each packet. Every node constructs a complete graph of the network using information that is obtained through control packets that are periodically broadcast by all nodes in the network. The topology graph could also be used to calculate routing tables before runtime, making the routing algorithm completely static. The network topology information that is broadcast also includes a cost for each edge. The cost values can then be input into Dijkstra's algorithm to calculate a minimum-cost (usually minimum-delay or minimum-hop) path from the node to all other destinations in the network. Following this, the next hops from the calculated routes are filled into routing tables.

One clear issue with this strategy is that while the paths are all minimal, there is no guarantee that there are no cyclic dependencies between paths; thus, the algorithm can produce deadlock. Another issue with this algorithm when applied quasi-statically is that while routes are being reconfigured, packets are still being routed through the network. This creates the possibility of temporary loops forming as new cost information is propagated through the network. To reduce the impact of reconfiguration, each node broadcasts its link information at a different offset in time so that the network does not become flooded with configuration packets when changes occur in the network.

5.2 Adaptive Routing

This section outlines routing strategies that use local information in a router (typically the current node location and congestion information) to decide the next hop in the path taken by the packet. By making these decisions at run-time, two packets destined to the same final node can take different routes and, as a result, arrive out of order. This section is broken down into two subsections: One for adaptively routing strategies in structured topologies and one for routing in arbitrary topologies. The strategies used in each situation can vary quite a bit.

5.2.1 Structured Topologies

This section discusses methods to route adaptively in networks that have some sort of structured topology. The majority of the techniques apply to q-ary n-cubes.

5.2.1.1 Store-and-Forward

As discussed in Section 4, a deadlock-free routing sub-function is all that is required to ensure deadlock-freedom in store-and-forward networks. Combining Dally and Seitz's dimension ordered routing [21] with any adaptive routing strategy will always give a deadlock-free routing algorithm. Since it is relatively straightforward to create a useful adaptive routing algorithm in this manner, the search was focused on store-and-forward routing techniques for structured topologies.

5.2.1.2 Wormhole

Mohapatra's wormhole routing survey [26] contains a discussion of various routing techniques in structured networks. The techniques in [27] are presented as an example of a fully adaptive algorithm that requires a large set of virtual channels to prevent deadlock. The large number of buffers required to implement the techniques described therein make them less practical, but they are mentioned here because they are cited frequently in the literature as examples of preventing deadlock through addition of virtual channels.

Dally and Aoki present a pair of algorithms for routing in wormhole-switched networks that allow as many *dimension reversals* as there are virtual channels in the network [28]. A *dimension reversal* occurs when a header flit is routed between two nodes with decreasing node address as expressed in [21]. One algorithm only allows as many reversals as there are virtual channels in the network, then the header must be routed according to dimension-ordered routing as described previously in the paper. The second, more flexible algorithm keeps a counter for the number of dimension reversals that a given header has gone through. If the header encounters a node where all outgoing channels contain waiting packets with lower dimension reversal counters, then the header must move to a set of buffers where only dimension-ordered routing is allowed. In both algorithms, once dimension-ordered routing begins, the adaptive routing cannot resume. This guarantees the existence of a deadlock-free escape route for all headers. Though this algorithm can require a large number of virtual channels to provide high adaptability, it is attractive because it allows routing in any direction and still provides deadlock-freedom.

One of the initial draws of wormhole switching is the small buffer space it requires to work efficiently, but introducing a large number of virtual channels into the routers increases the memory required for buffers and the complexity of the switching logic inside the router. The following algorithms address the issue of large buffer requirements in wormhole-switched networks to simplify router implementation.

Glass and Ni describe a set of techniques for preventing deadlock in wormhole-switched networks that have a mesh topology by reducing which direction the header flit can turn in the network. This reduces the overall adaptivity of the algorithm, but prevents deadlock without requiring a large number of virtual channels. It is an important paper because it illustrates the idea of preventing deadlock by limiting the paths of flits through the network rather than creating monotonic buffer ordering [24].

Glass and Ni's adaptive algorithm can be applied to n-dimensional meshes or q-ary n-cubes, but Demaine and Srinivas show another method to apply the idea to q-ary n-cubes to provide partially adaptive routing in [29]. They simply compare the current address of a message with that of its destination. Then, they apply direction-first routing on the lowest two dimensions that differ until they are corrected. Next, apply the direction-first routing algorithm again on the lowest two dimensions with differing values. Repeat this until the packet reaches its destination. This technique of applying routing across two dimensions at a time is actually based on the ideas in the paper described next.

Chien and Kim present an algorithm to limit the number of virtual channels that are required to route adaptively in n-meshes using wormhole switching as the dimension of the topology increases. This is accomplished by routing in a single plane of the network at a time. This technique also reduces the total adaptivity of the algorithm in order to provide deadlock-freedom with relatively few virtual channels in high-dimensional topologies. Applying this technique to q-ary n-cubes seems relatively straightforward. Simply choose an adaptive algorithm for low dimensional q-ary n-cubes, and then apply it to a single plane at a time as described in the paper for meshes [30].

5.2.2 Arbitrary Topologies

Routing adaptively with wormhole switching in an unstructured network is much more difficult than with store-and-forward switching because deadlock is much easier to introduce into wormhole-based systems. However, Silla et al. present an algorithm that can convert a static, deadlock-free wormhole-based routing algorithm into an adaptive routing algorithm that is also deadlock-free [31]. This is accomplished through the necessary and sufficient conditions for deadlock in wormhole-switched networks proved by Duato in [2]. The essence of the algorithm is to add virtual channels to the existing set of channels. These new channels can be used as adaptive escape paths when the normal, original channels are blocked. In order to prevent deadlock in these new channels, flits can escape to the original channels that are known to be deadlock-free. Once routed on the original channels, the flits must stay in original channels to prevent introducing deadlock into the deadlock-free escape path. One clear drawback of this is that you must already have a deadlock-free static routing strategy to use on your topology. An example of how this could be used would be to create a deadlock-free, deterministic routing algorithm using the methods shown in [1], and then add virtual channels to allow routing on any alternative, minimal paths that may exist.

Shortest Path First with Emergency Exist is a routing strategy built on top of the original SPF algorithm [32]. This adds alternate paths to the routing table based on adjacent nodes that fall on different branches in the sink tree [33, p. 352-353]. This prevents packets from oscillating between two nodes under congestion. Like the SPF algorithm, there is no built-in method to deal with deadlock, so extra work would be required to build paths that did not cause deadlock when using wormhole switching.

5.3 Issues/Design Considerations

There appears to be large difference in the amount of research time invested in routing in wormhole networks versus store-and-forward networks. This is in all likelihood a result of the difficulty of routing in wormhole networks without introducing deadlock. As a result, it is very important to consider the routes being taken by packets in a network that is built onto a wormhole-switched protocol.

Q-ary n-cubes are the target of most of the research that is intended for non-arbitrary networks. Since these algorithms can be implemented in both hypercube and torus topologies, they are also what has been included in the paper.

6 FAULT-TOLERANT ROUTING

This section provides an overview of the techniques found on fault-tolerant routing. The first section explains some theory relating to fault-tolerant routing. Next, algorithms are illustrated that are designed to be used in networks with structured topologies.

6.1 Fault-Tolerant Routing Theory

In [34], Duato again describes his techniques for preventing deadlock in arbitrary wormhole-switched networks. His theorems are reliant on the idea of a connected routing sub-function for a given network. It is clear that if a given routing sub-function is connected (and has an acyclic extended buffer dependency graph), then all channels that are not part of the sub-function can be removed and the network will still function correctly. However, if a channel in the sub-function is removed, then it is possible that the network is no longer connected and deadlock-free. If a given network contains two such routing sub-functions and they are completely disjoint, then removing communication links from only one of the sub-functions will not prevent the network from being connected and deadlock-free. Similarly, if there are three such sub-functions that are completely disjoint, then two faulty nodes alone cannot disconnect the network or create deadlock. This is the key idea of Duato's paper; specifically, that when $r + 1$ removed links from the network is the minimal set that has non-empty intersection with all of the network's routing sub-function, the network can tolerate r faults. This is because you cannot remove r links from the network that will intersect with all of the routing sub-functions. The routing sub-function(s) with empty intersection will provide connected, deadlock-free routing for the network.

Duato proposes a design method for creating r-fault-tolerant routing algorithms using existing routing techniques. The method is less constructive than Duato's design methodology for deadlock-free adaptive routing algorithms in store-and-forward networks, but it still could be useful when designing networks that are expected to have failing nodes or channels. The design method begins by applying an existing fully adaptive non-minimal routing algorithm to the network. Then, a connected routing sub-function has to be extracted with an acyclic extended buffer dependency graph. After this, at a given node in the routing sub-function, for each outgoing link c_i, remove the link and $r - 1$ other outgoing links to determine if the sub-function becomes disconnected. If so, add a new outgoing link to the node that causes the sub-function to be connected again without adding a cycle to the extended buffer dependency graph. This should not simply replace one of the removed links. If the structure of the topology is regular, then the added links for each outgoing link should be replicable across all nodes in the network, creating r-fault tolerance in the network.

6.2 Structured Topologies

This section describes techniques to deal with faults in networks that have a structured topology. First, the fault tolerance of regular adaptive algorithms is described, and then techniques are presented that extend existing adaptive routing algorithms to provide fault tolerance.

6.2.1 Fault Tolerance Inherent in Adaptive Routing

In minimal, fully adaptive routing, faulty channels can frequently be routed around since a packet typically has multiple minimal paths it can follow from each router. For example, consider the illustration in Figure 4 versus Figure 5. This figure was originally used by Chien and Kim in [30] to show the benefit of adaptive routing when the network is under heavy load. All of the paths in Figures 4 and 5 are minimal, yet they clearly use largely different nodes. This illustrates that if the "overloaded channels" in Figure 4 were actually failed channels, many of the nodes would still be able to deliver their packets using alternate, minimal routes.

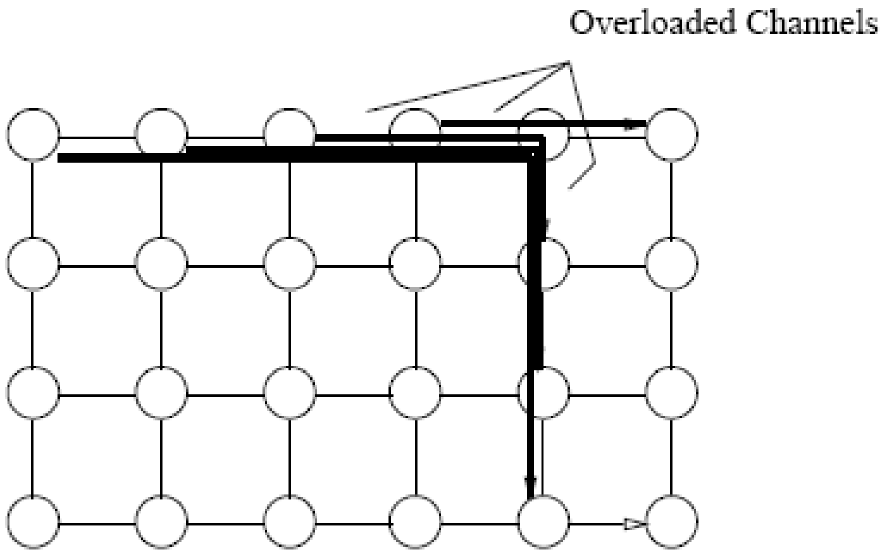

Figure 4. A congested network [30].

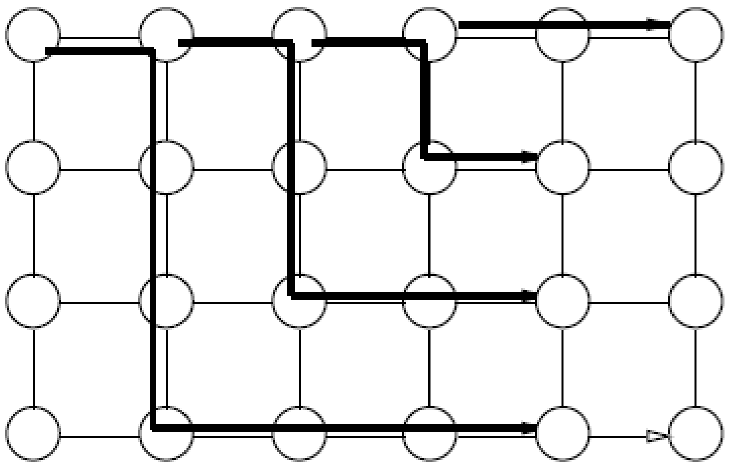

Figure 5. Same source/destination pairs with alternate minimal paths being taken [30].

This is a direct result of the degree of connectivity that most regular topologies show. There are, however, a few issues with simply relying on the fault tolerance of adaptive algorithms to deal with faulty channels or routers in the network. First, many adaptive algorithms are not fully adaptive. This means that while a non-faulty minimal path may be available in the topology, the routing algorithm may have conservative deadlock prevention that disallows many minimal paths from being taken. Also, when nodes or channels begin to fail, the original assumptions that were used to construct the routing algorithm have changed, so the routing graph may actually become disconnected or prone to deadlock.

A specific example of fault tolerance created from a highly adaptive routing algorithm is the algorithm proposed by Dally and Aoki in [28] (described in Section 5.2.1.2). Dally and Aoki allow their algorithm to route packets non-minimally to increase the fault tolerance it provides. With non-minimal routing comes the risk of livelock, so they present simple techniques to prevent continuously routing packets away from their destination. Since each packet can only execute a certain number of dimension reversals, it is eventually forced into deterministic routing. This guarantees it will not circulate infinitely in the network.

The main problem with the technique is that if a packet is forced into deterministic routing, then it no longer has any ability to route around faults. One other issue that is addressed by papers in the next section is the shape of faults. Well-placed faults could create a situation where a packet would need to misroute more times than it has dimension reversals available to it. Such faults would obviously not be tolerated by this routing scheme.

6.2.2 Block Faults

Since adaptive routing can frequently route around simple faults as described above, a question that some researchers chose to address was how to deal with more complex faults using the same methods. In order to apply the adaptive routing techniques to more complex faults, the idea of converting complex faults to simple faults was introduced and is described in the papers in the following section.

6.2.2.1 Planar Adaptive Routing

As described previously, planar adaptive routing is a technique for routing in high-dimensioned topologies with a small number of virtual channels [30]. This is achieved by limiting how many dimensions a packet can be adaptively routed in at a time. The paper also addresses the possibility of providing fault tolerance with the planar adaptive algorithm.

The technique is designed to route around what are called *convex* fault regions. The authors chose to deal with convex fault regions because it can be shown that routing around a convex region will only require misrouting in a single dimension in a given routing plane. This greatly simplifies deadlock and livelock prevention when routing around faults.

Creating convex fault regions is performed locally. If a node finds that on a given plane it is connected to faulty nodes in two dimensions, then it marks all of its channels in that plane as faulty. The result is that all reachable nodes will have only one faulty dimension in a given plane that must be routed around.

As shown in Figure 6, once the convex fault regions have been created, misrouting around the region is very simple. If a packet cannot route toward the destination in one dimension, it chooses a random direction on the other dimension to route along. Once there is no longer an obstruction in the original routing dimension, the packet will resume moving in its originally intended direction until it clears the convex fault region. Then it moves back along the second dimension to the rank where it began misrouting.

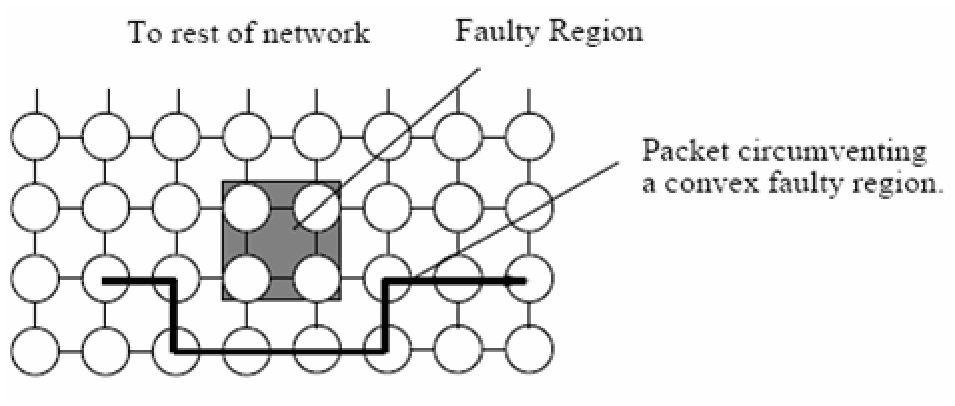

Figure 6. Convex fault region and a misrouting packet [30].

6.2.2.2 Adaptive Extensions

Chalasani and Boppana present a method for extending an existing deadlock-free fully adaptive wormhole routing scheme to tolerate block faults [35]. This paper also targets convex fault blocks, though in n-dimensional tori. This allows messages to be easily routed around the faulty areas without large movements away from their destinations. The movement of messages around faulty nodes is very similar to that of the planar-adaptive scheme, but since the movement is not limited to two dimensions at a time, the virtual channel requirements are higher than that for planar-adaptive routing. In addition to the information in this paper, a more detailed description of building block fault regions is available in [36].

6.2.3 Backtracking

Chen and Shin present an algorithm for routing in hypercubes when using store-and-forward switching in [3]. This algorithm simply routes along shortest paths when there are no faults. If a fault is encountered, then the algorithm first attempts to follow an alternate minimal path. If that fails, then the algorithm attempts to route the packet along a non-minimal path. If all outward paths from a given node are faulty, then the packet will move to a node it had previously visited (backtrack) and try to find a new route.

This algorithm provides no guarantee for deadlock freedom. It does, however, keep track of where the packet has been, and this information is used to prevent packets from looping endlessly when they encounter faults. This implies that the algorithm does provide livelock-freedom, though this is not proven in the paper.

6.3 Issues/Design Considerations

Though Duato's design methodology [34] for fault-tolerant routing is not as straightforward as his design methodology for adaptive routing algorithms in store-and-forward networks, the theorems he presents in his paper are very relevant to designing fault-tolerant networks. They provide a means to prove that a given topology and routing algorithm will remain deadlock-free and fully connected for up to *r* faults. While many of the algorithms presented in this section can protect against multiple faults, several do not have guarantees for how many faults they can tolerate before the network begins failing. Having Duato's theorems available to independently calculate the performance of these algorithms is a very valuable tool.

One particularly important aspect of [35] is that it shows how any adaptive algorithm can be augmented to provide fault tolerance with a simple strategy and a few extra virtual channels. The extension for fault tolerance in [30] should be similarly applicable to adaptive algorithms.

7 CONCLUSION

This paper presented an overview of some of the key issues in designing routing functions for interconnects, including deadlock, livelock, and faulty components. Additionally, techniques for dealing with these issues were presented and discussed with commentary on possible implementation issues. The techniques that were presented could be applied to networks utilizing store-and-forward or wormhole switching. Techniques were also presented that were applicable to irregular network topologies as well as highly regular topologies such as q-ary n-cubes.

8 REFERENCES

[1] L. Zakrevski, S. Jaiswal, L. Levitin, and M. Karpovsky, A New Method for Deadlock Elimination in Computer Networks with Irregular Topologies, *Proc. IASTED Int. Conf. Parallel and Distributed Computing and Systems*, 1999.

[2] J. Duato, A Necessary and Sufficient Condition for Deadlock-Free Adaptive Routing in Wormhole Networks, *IEEE Trans. Parallel and Distributed Systems*, Vol. 6, No. 10, pp. 1055-1067, October 1995.

[3] M. Chen and K. G. Shin, Depth-First Search Approach for Fault-Tolerant Routing in Hypercube Multicomputers, *IEEE Trans. Parallel and Distributed Systems*, Vol. 1, No. 2, April 1990.

[4] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks*. San Francisco, CA: Morgan Kaufmann, 2003, pp. 83, 141.

[5] R. G. Gallager, A Minimum Delay Routing Algorithm Using Distributed Computation, *IEEE Trans. On Communications*, Vol. 25, No. 1, pp. 73-85, January 1977.

[6] G. Shippen, Enter the Inner Sanctum of RapidIO: Part 2, *EE Times*, http://www.eetimes.com/story/OEG20031124S0020 (Current July 25, 2007).

[7] Spacewire – Links, nodes, routers, and networks, *European Cooperation for Space Standardization*, ECSS-E-50-12A, pp. 33, 82-83, 89, January 24, 2003.

[8] L. M. Ni and P. K. McKinley, A Survey of Wormhole Routing Techniques in Direct Networks, *IEEE Computer*, Vol. 26, no. 2, pp. 62-76, February1993.

[9] P. Kermani and L. Kleinrock, Virtual Cut-Through: A New Computer Communication Switching Technique, *Computer Networks*, Vol. 3, pp. 267-286, 1979.

[10] J. Duato, A Necessary and Sufficient Condition for Deadlock-Free Routing in Cut-Through and Store-And-Forward Networks, *IEEE Trans. On Parallel and Distributed Systems*, Vol. 7, No. 8, pp. 841-854, 1996.

[11] C. Izu, R. Beivide, and C. Jesshope, Mad-Postman: a Look-ahead Message Propagation Method for Static Bidimensional Meshes, *Proc. of Second Euromicro Workshop on Parallel and Distributed Systems*, pp. 117-124, January 1994.

[12] U. Feige and P. Raghavan, Exact Analysis of Hot-Potato Routing, *Proc. of the 33rd Annual Symp. on Foundations of Computer Science*, pp. 553-562, October 1992.

[13] S. Konstantinidou and L. Snyder, The Chaos Router, *Symp. on Parallel Algorithms and Architectures*, Vol. 19, No. 1, pp. 79-88, March 1991.

[14] W. J. Dally and C. L. Seitz, "Torus Routing Chip," U.S. Patent 4 933 933, June 12, 1990.

[15] S. Toueg and K. Steiglitz, Some Complexity Results in the Design of Deadlock-Free Packet Switching Networks, *SIAM Journal on Computing*, Vol. 10, No. 4, pp. 702-712, November 1981.

[16] L. Schwiebert and D. N. Jayasimha, A Universal Proof Technique for Deadlock-Free Routing in Interconnection Networks, *Proc. 7th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pp. 175-184, 1995.

[17] R. Cypher and L. Gravano, Requirements for Deadlock-Free, Adaptive Packet Routing, *Symp. Principles of Distributed Computing*, pp. 25-33, 1992.

[18] S. Toueg and J. Ullman, Deadlock-Free Packet Switching Networks, *Proc. 11th Annual ACM Symp. Theory of Computing,* pp. 89-98, 1979.

[19] K. D. Gunther, Prevention of Deadlocks in Packet-Switched Data Transport Systems, *IEEE Trans. Communications*, Vol. COM-29, No. 4, pp. 512-524, April 1981.

[20] I. Gopal, Prevention of Store-and-Forward Deadlock in Computer Networks, *IEEE Trans. Communications*, Vol. COM-33, No. 12, pp. 1258-1264, December 1985.

[21] W. J. Dally and C. L. Seitz, Deadlock-Free Message Routing in Multiprocessor Interconnection Networks, *IEEE Trans. Computers*, Vol. C-36, No. 5, pp. 547-553, May 1987.

[22] J. Upadhyay, V. Varavithya, and P. Mohapatra, Routing Algorithms for Torus Networks, *Intl. Conf. High Performance Computing*, pp. 743-748, 1995.

[23] J. Y. Ngai and C. L. Seitz, A Framework for Adaptive Routing in Multicomputer Networks, *Symp. Parallel Algorithms and Architectures*, Vol. 19, No. 1, pp. 6-14, March 1991.

[24] C. J. Glass and L. M. Ni, The Turn Model for Adaptive Routing, *Journal of the ACM*, Vol. 41, No. 5, pp. 874-902, September 1994.

[25] J. M. McQuillan, I. Richer, and E. C. Rosen, The New Routing Algorithm for the ARPANET, *IEEE Trans. Communications*, Vol. 28, No. 5, pp. 711-719, May 1980.

[26] P. Mohapatra, Wormhole Routing Techniques for Directly Connected Multicomputer Systems, *ACM Computing Surveys*, Vol. 30, No. 3, pp. 374-410, 1998.

[27] D. H. Linder and J. C. Harden, An Adaptive and Fault Tolerant Wormhole Routing Strategy for k-ary n-cubes, *IEEE Trans. Computers*, Vol. 40, No. 1, pp. 2-12, January 1991.

[28] W. J. Dally and H. Aoki, Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels, *IEEE Trans. Parallel and Distributed Systems*, Vol. 4, No. 4, pp. 466-475, April 1993.

[29] E. D. Demaine and S. Srinivas, A Novel Routing Algorithm for k-ary n-cube Interconnection Networks, presented at *High Performance Computing Symposium*, Montreal, Canada, July 1995.

[30] A. A. Chien and J. H. Kim, Planar-Adaptive Routing: Low-cost Adaptive Networks for Multiprocessors, *Journal of the ACM*, Vol. 42, No. 1, pp. 91-123, 1995.

[31] F. Silla, M. P. Malumbres, A. Robles, P. Lopez, and J. Duato, Efficient Adaptive Routing in Networks of Workstations with Irregular Topology, *Communication, Architecture, and Applications for Network-Based Parallel Computing*, pp. 46-60, 1997.

[32] Z. Wang and J. Crowcroft, Shortest Path First with Emergency Exits, *ACM SIGCOMM Computer Communication Review*, Vol. 20, No. 4, pp. 166-176, 1990.

[33] A. S. Tanenbaum, *Computer Networks 4th Edition*. Upper Saddle River, NJ: Prentice Hall PTR, 2003, pp. 352-353.

[34] J. Duato, A Theory of Fault-Tolerant Routing in Wormhole Networks, *IEEE Trans. Parallel and Distributed Systems*, Vol. 8, No. 8, pp. 790-802, August 1997.

[35] S. Chalasani and R. V. Boppana, Fault-Tolerant Wormhole Routing in Tori, *Proc. 8th Int. Conference on Supercomputing*, pp. 146-155, 1994.

[36] R. V. Boppana and S. Chalasani, Fault-Tolerant Wormhole Routing Algorithms for Mesh Networks, *IEEE Trans. Computers*, Vol. 44, No. 7, pp. 848-864, July 1995.

DISTRIBUTION

1 Leonard G. Burczyk
Mail Stop D440
Los Alamos National Laboratory
Los Alamos, NM 87545

4 Office of Nonproliferation Research & Development
NA-22/GH-068
Attn: Vaughn Standley
W. Randy Bell
Roger Byrd
Philip Cole
1000 Independence Ave. SW
Washington, DC 20585-0420

1	MS0343	Jim B. Woodard, 2600
1	MS0406	Toby O. Townsend, 5713
1	MS0501	Steve B. Rohde, 5337
1	MS0503	Mythi M. To, 5337
1	MS0530	Dan E. Gallegos, 2623
1	MS0530	Eric Olilla, 2623
1	MS0801	Dorothy S. Rarick, 9330
1	MS0806	John M. Eldridge, 9336
1	MS0806	Len Stans, 9336
1	MS0806	Jason S. Wertz, 9336
1	MS0971	Bob M. Huelskamp, 5730
1	MS0971	Jae W. Lee, 5733
1	MS0972	Kurt R. Lanes, 5560
1	MS0986	J. (Heidi) Ruffner, 2664
1	MS0964	Brian C. Brock, 5733
1	MS0980	Matt P. Napier, 5571
1	MS0980	Steve M. Gentry, 5703
1	MS0982	Dan E. Carroll, 5732
1	MS0982	Dan Kral, 5732
1	MS0982	Phil J. Cole, 5732
1	MS0986	Dave M. Bullington, 2664
1	MS0986	Jonathon W. Donaldson, 2664
1	MS0986	John V. Vonderheide, 2660
1	MS0986	Jeff L. Kalb, 2664
1	MS0986	Dave Heine, 2664
1	MS0986	Dave S. Lee, 2664
1	MS1003	Ray H. Byrne, 6473
1	MS1172	Ron J. Franco, 5415
1	MS0899	Technical Library, 9536 (electronic copy)

Sandia National Laboratories