Adaptive Fault Tolerant Routing in Interconnection Networks: A Review

Mayank Gupta (mayank@synopsys.com)
Alok Kuchlous (**alokk@synopsys.com)**

1. Introduction

The current multiprocessor systems can have as many as 4K components attached to the interconnection network. Even with the current levels of very high reliability, the large number of components and an even larger number of interconnects increase the probability that one or more of them will fail. In some critical applications (such as defense, space applications or even consumer networks which guarantee a high degree of reliability and quality of service) even such probabilities might not be acceptable. It is very desirable in such circumstances to have routing algorithms and router implementations that can work in spite of such failures. We review two [1, 4] such schemes in sections 2 and 3. We also look at the implementation details of one reliable router [2] in section 4. We end each section with a brief review where we critique the solution and compare it with the other solutions. We also provide some directions for further work.

1.1 Problem Definition

The problem is to route the packets in an interconnection network in the presence of faults. The assumption is that the faults leave the network connected, i.e. with best possible routing the packet can still reach from any node to any other node. It is also assumed that the presence of a fault can be detected – we do not present any techniques for detection of link or node failure.

1.2 Deterministic vs. Adaptive Routing

Routing algorithms can be classified as either *deterministic* or *adaptive* [3]. *Deterministic* algorithms pick out only one route given the source and destination pair (or incoming channel and the destination). On the other hand, *adaptive* algorithms decide the path based on the conditions of the network (they generally have more than one option to choose from at any given time). Obviously *deterministic* algorithms cannot be fault tolerant. Any fault in the unique path chosen by the algorithm will make it fail.

1.3 Fault Models

The faults in the interconnection network can be of many kinds. The main types of faults discussed in the literature are of two kinds. Either the entire processing element (the router) may fail, or any communication channel may fail. The former is referred to as *node faults* and the latter as *link faults*. On a *node fault,* all the links incident onto it are also supposed to be faulty. The faults may also be classified as *static* or *dynamic*. In case of static faults the faults remain the same throughout the operation of the interconnection network, while dynamic faults may change while the network is in operation.

2. Adaptive Routing through Dimension Reversals: Dally and Aoki [1]

Quite a few networks that use dimension ordered routing because it is simple and deadlock-free. Deadlock is avoided by ordering channels so that messages travel along paths of strictly increasing channel numbers. Channels are ordered so that all of the channels in each dimension are greater than all the channels in the preceding dimension. This ordering, however, results in a unique path from a source to destination and thus does not allow adaptive routing. As already noted, adaptive routing is a necessity for fault tolerance.

The authors present two variations on the dimension ordered routing that are deadlock free and adaptive. Both algorithms permit misrouting (routing a packet along a non-minimal path). Both of them avoid deadlock using virtual channels and the concept of dimension reversal (*DR*) to eliminate cyclic dependencies.

2.1 Dimension Reversal

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

i. All packets are initialized with a *DR* of 0.
ii. Each time a packet routes from a channel $\mathbf{c_i} \in \mathbf{C'_p}$ to a channel $\mathbf{c_j} \in \mathbf{C'_q}$ where $\mathbf{p} > \mathbf{q}$ the *DR* of the packet is incremented.

2.2 Static Algorithm

The static algorithm divides the virtual channels of each physical channel in to nonempty classes numbered from 0 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 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, after 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 (dimension order routing).

Note that the algorithm is deadlock free because the virtual channels are fully ordered in the order of increasing *DR* number so there are no indirect dependencies between virtual channels of one *DR* to virtual channels with another *DR*. Within each *DR* the routing is dimension ordered so there are no cyclic dependencies.

The static assignment of *DR* numbers to virtual channels restricts the maximum number of DR's to the number of virtual channels available on a router. This 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 next algorithm solves this problem.

2.3 Dynamic Algorithm

The dynamic algorithm divides the virtual channels of each physical channel into two non-empty 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 *DR* of **p** cannot wait on a channel labeled with a *DR* of **q** if **p** ≥ **q**. A packet that reaches a node where all output channels are occupied by packets with equal of lower *DR*'s must switch to deterministic class of virtual channels. When a packet enters the deterministic channels, it must be routed dimension order, i.e. in the lowest dimension in which the current node address and the destination address differ. Once on the deterministic channels, the packet cannot reenter adaptive channels.

Assertion: 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 $DR(p_{max}) \geq DR(q)$ for all $q \in P$. 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 $DR(p_{max}) \geq DR(pj) \geq r$.

The basic idea here is that there can be no deadlock because in any cycle the packet with the maximum direction reversals (and hence largest *DR*) will always revert to the deterministic channel breaking the cycle.

This algorithm is similar to Duato's algorithm because it maintains a set of dimension ordered deterministic channels as an escape path. The slight difference is that here the packets are allowed to wait on some channels, while in Duato's algorithm the packets never wait on a busy channel.

2.4 Routing Policy

Progress: To guarantee progress toward a destination the algorithms have to be constrained. The amount of misrouting is limited by placing an upper limit on the number of steps a message may take away from its destination.

Throttling: The adaptive virtual channel pool of the dynamic algorithm can be monopolized by eager sources unless some form of throttling is used. If messages are injected faster than the network can deliver them, most of the messages will be forced to use 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** ≥ **q**. This method divides the adaptive virtual channel pool into classes to prevent new messages from consuming the entire pool.

Selection Functions: At each stage the routing function tries to pick an output channel in the following order:

i. Pick a productive channel if available.
ii. Pick any other legal (according to the conditions mentioned above) adaptive channel, if it is busy wait on it.
iii. Pick the deterministic channel.

Note that for (i) and (ii) the algorithms may have a choice of many channels. The authors evaluate three different ways of selecting. Minimum congestion (pick the direction with the most available virtual channels), maximum flexibility (pick the direction in which the largest distance is to be traveled), and straight line (pick the dimension closest to the current dimension).

2.5 Results

All the results presented in this section are for 256 node, 16-ary 2-cube mesh networks with 16 virtual channels per physical channel.

Latency is measured by applying a constant rate source to each input. Adaptive routing gives a significant performance advantage for traffic patterns that load channels non-uniformly. Figure 1 shows latency as a function of throughput for the three routing strategies under bit reversal traffic. Adaptive routing achieves three times the throughput of deterministic routing for bit reversal traffic. For uniform traffic (not shown here) the adaptive algorithms do not do much better than the deterministic one.

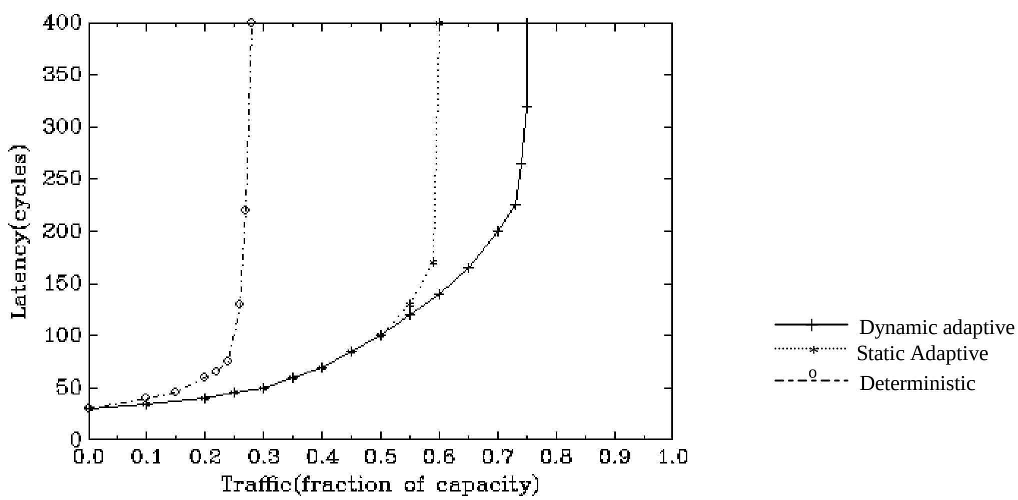

Figure 1: Latency versus accepted traffic for a 16-ary 2-cube under bit reversal traffic. Deterministic dimension order routing is compared with static and dynamic adaptive routing. Deterministic routing saturates at about 25% capacity. Static and dynamic adaptive routing achieve three times this performance (saturating at 60% and 75% capacity, respectively) by routing to distribute the network load.

Fault tolerance: Figure 2 and Figure 3 illustrate the graceful degradation of an adaptive network as channels fail. Figure 2 shows the latency of the network at 50% capacity of the random traffic as function of the percentage of the faulty channels. Latency increases only by a factor of 2.3 from a fault free network to network with 8% faulty channels. Figure 3 shows network throughput as a function of faulty channels. Network throughput also degrades gracefully, dropping from 66% capacity to 54% capacity when 8% channels are faulty.

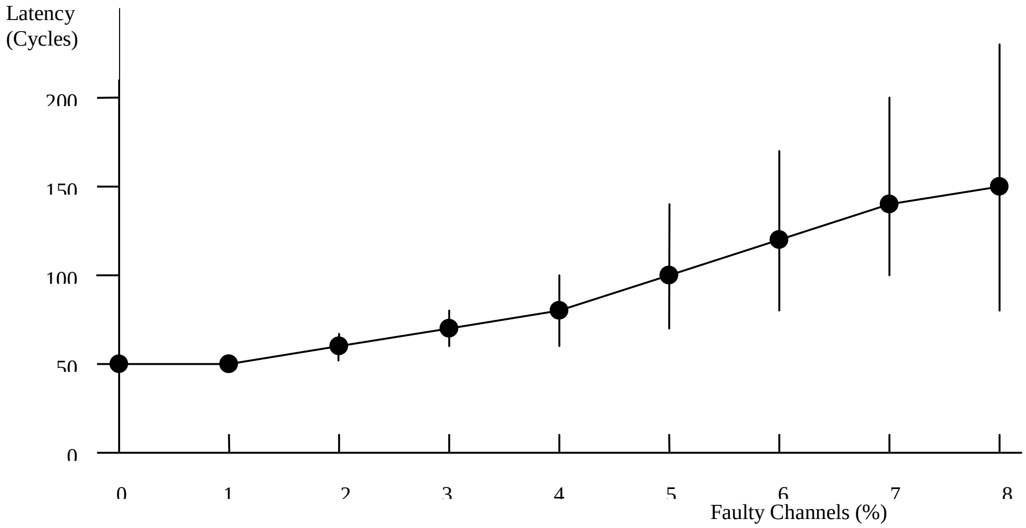

Figure 2. Latency versus percent faulty channels for a 16-ary 2-cube network operating at 50% capacity with random traffic. Each dot gives the mean latency of 20 randomly generated faulty networks. The ends of the vertical error bars represent the 1σ points of each latency

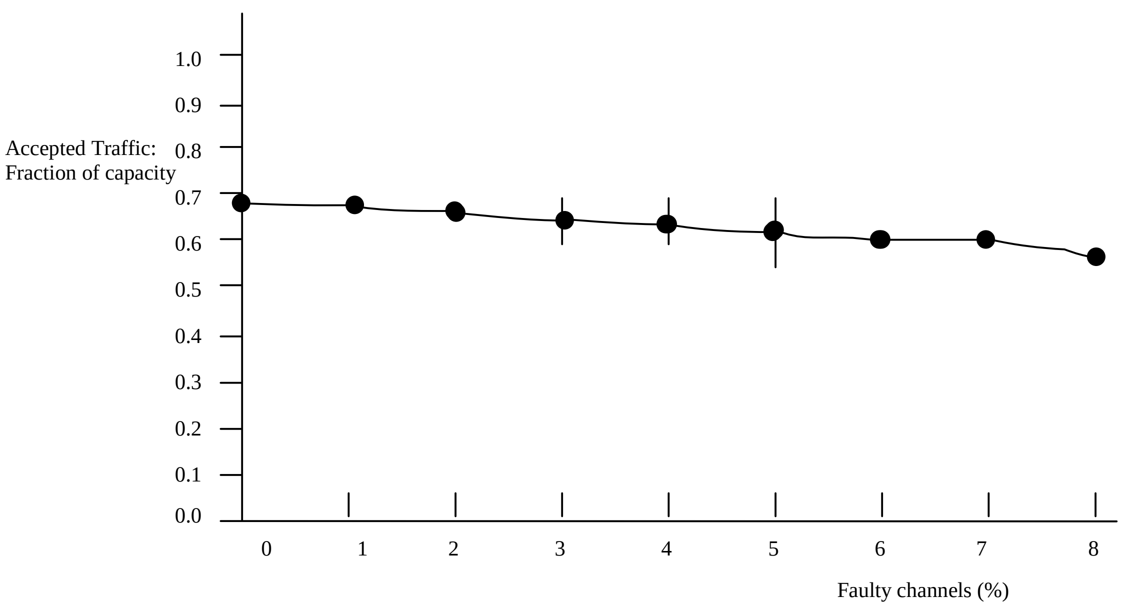

Figure 3. Throughput versus percent faulty channels for a 16-ary 2-cube network with random traffic and throttling with a single entry lane. Each dot gives the mean throughput of 20 randomly generated faulty networks. The error bars show the 1σ points of each distribution.

2.6 Review

The solution is proposed for adaptiveness and does quite well on bad traffic patterns. . The algorithms are simple and can be implemented quite efficiently. As the authors show, the performance degradation in the presence of faults is quite graceful. But the algorithms are not provably fault tolerant; that is, situations exist where a single fault will lead to a failure in the delivery of packets. If there is a fault on the deterministic route then any packet that has to take that route will fail to reach its destination. This case was not seen during simulations because of the low traffic volume used. The simulations are done at 50% of the network capacity; it is unlikely that too many packets would need to use the deterministic routes at such a low traffic volume. This solution should be acceptable where the simplicity and efficiency of implementation is important and there is no need for guarantees on the fault tolerance.

2.7 Future Work

The adaptiveness of the algorithm is a direct function of the number of virtual channels available. The authors give the results for the case of 16 virtual channels per physical channel. It would be interesting to see how the performance varies with the number of virtual channels. Also the traffic injection at the source nodes is always uniform. It is likely that different algorithms will behave differently when the traffic has different time profile (bursty v/s uniform). The solution could also be changed to make it provably fault tolerant. One way would be to make the deterministic routing use the modified turn model (rather than the dimension ordered routing) described in Section 3.

3. Fault Tolerant Routing in Meshes using the Turn Model: Glass and Ni [4]

This paper shows how to modify the routing algorithm produced by the *turn model* [5] to handle static and dynamic faults. It first describes how to modify the negative first routing algorithm to make it one fault tolerant. Then, it describes how to modify the negative first routing algorithm to make it *(n-1)*-fault tolerant where *n* is the number of dimensions of the mesh.

3.1 Assumptions

The paper assumes a node fault model i.e. the faults are assumed to occur on the routers, not on the channels. When a node is faulty all the channels incident on it are assumed to be faulty. The other assumption is that error recovery is done at a packet level. So if a packet does not reach the destination, the source resends the whole packet. We will look at the option of flit level error recovery in section 4 when we review the reliable router.

3.2 One-Fault-Tolerant Routing

The algorithms reviewed in section 2 used virtual channels to make the network adaptive and fault tolerant. The algorithm presented here uses *turn model* to make fault tolerant meshes without virtual channels. The turn model involves identifying the directions that packets can travel in a network, and prohibiting just enough turns to break every cycle. The routing algorithms the turn model produces are deadlock free, maximally adaptive for the network, minimal or non-minimal, and livelock free.

The one-fault-tolerant routing algorithm is a modification of the *negative first* routing algorithm produced by the turn model for two-dimensional meshes. The *negative first* routing algorithm routes message packets first in the negative directions (west and south) and then in the positive directions (east and north). The minimal version of the *negative first* routing algorithm can route packets adaptively except in the cases where the destination is northwest or southeast of the source. The non-minimal version of the negative first routing algorithm can usually route packets adaptively during the both its negative and positive phases. The only time during the negative phase that the routing is non-adaptive is when a packet is at a negative (west or south) edge of the mesh. Then the packet can only be routed along the edge in negative direction. The only time during the positive phase that the routing is non-adaptive is when a packet needs to travel along only one dimension in order to reach the destination. Simple modifications listed below, of the negative first routing algorithm, remove these few cases of non-adaptiveness.

- Route the packet west and south to the destination or farther west and south than the destination, avoiding routing the packet to a negative edge for as long as possible. If a faulty node on a negative edge blocks the path along the edge, route the packet one hop perpendicular to the edge.
- Route the packet east and north to the destination, avoiding routing the packet as far east or north as the destination for as long as possible. If a faulty node on a negative edge of the mesh blocks the path to a destination on the edge, route the packet one hop perpendicular to the edge, two hops toward the destination, and one hop back to the edge.

Figure 4 illustrates these modifications with a few examples of the new paths.

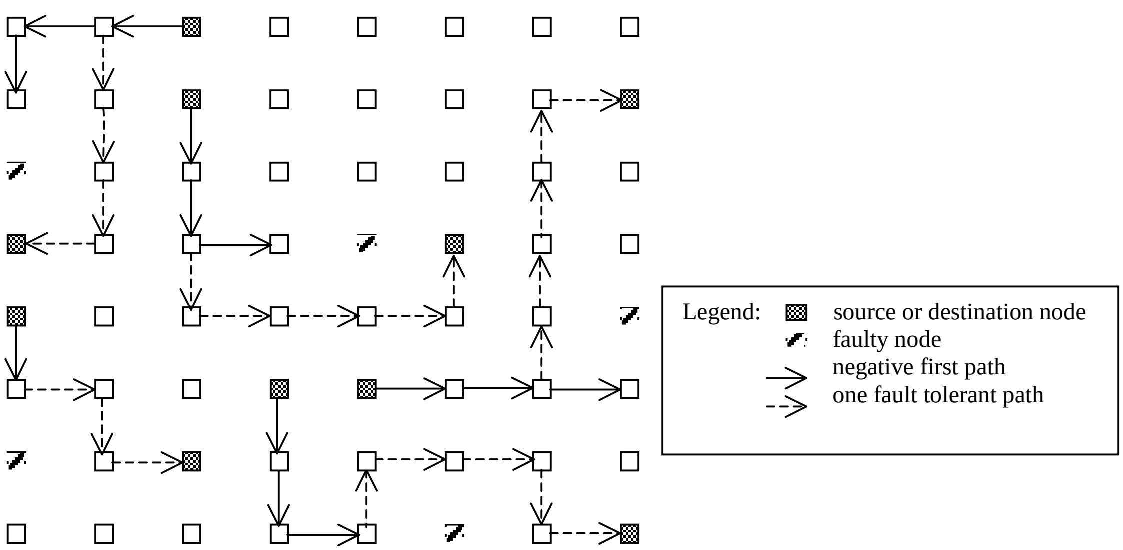

Figure 4. Examples of the paths allowed by the one fault tolerant routing algorithm for two dimensional meshes.

The idea behind the modifications is to avoid the cases where the *negative first* algorithm becomes non-adaptive. The algorithm does it by overshooting the destination when travelling in the negative directions and avoiding the south edges as much as possible.

If a node ever finds it impossible to route the packet farther, it discards the packet and possibly returns an acknowledgement of the error to the sender. The algorithm is one-fault-tolerant and deadlock free for two-dimensional meshes. The freedom from deadlock is proved by showing an order on the channels.

3.3 Results

Figure 5 and 6 compare the minimal, non-minimal, and one-fault-tolerant versions of the negative first routing algorithms for uniform and matrix-transpose traffic. These simulations were done on a faultless 10 X 10 mesh.

These results suggest that when contention is high, shorter path lengths (minimal routing) are more important than greater adaptiveness (non-minimal). The advantage of non-minimal routing, however, is that it can greatly increase the fault tolerance. To be practical, non-minimal routing should be used only to avoid faulty nodes.

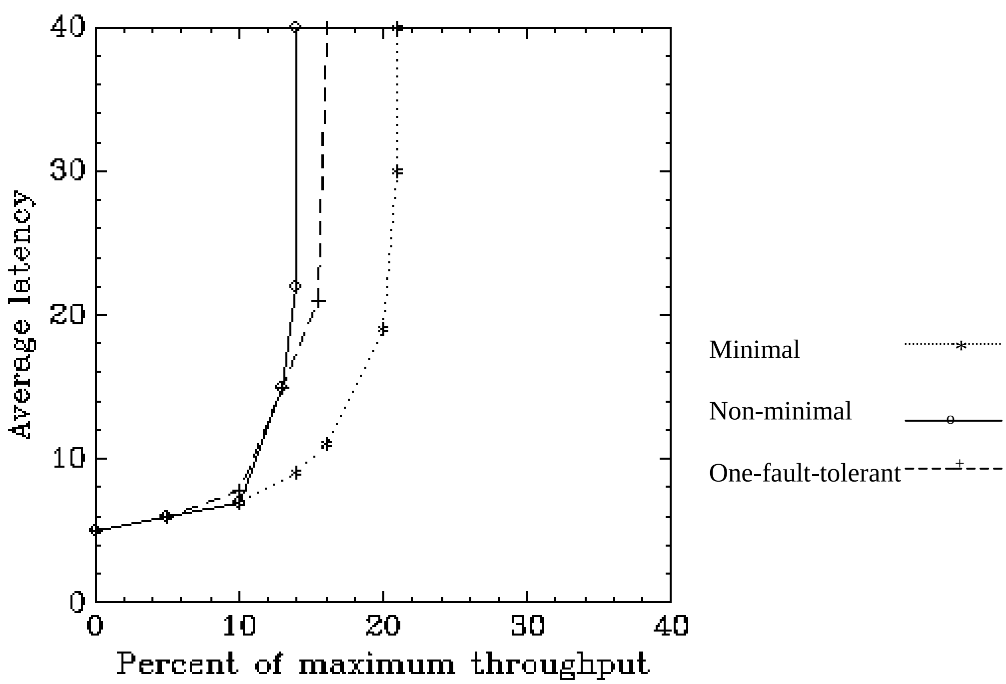

Figure 5. Comparison of the minimal, non-minimal and one-fault-tolerant versions of the negative-first routing algorithm for uniform traffic.

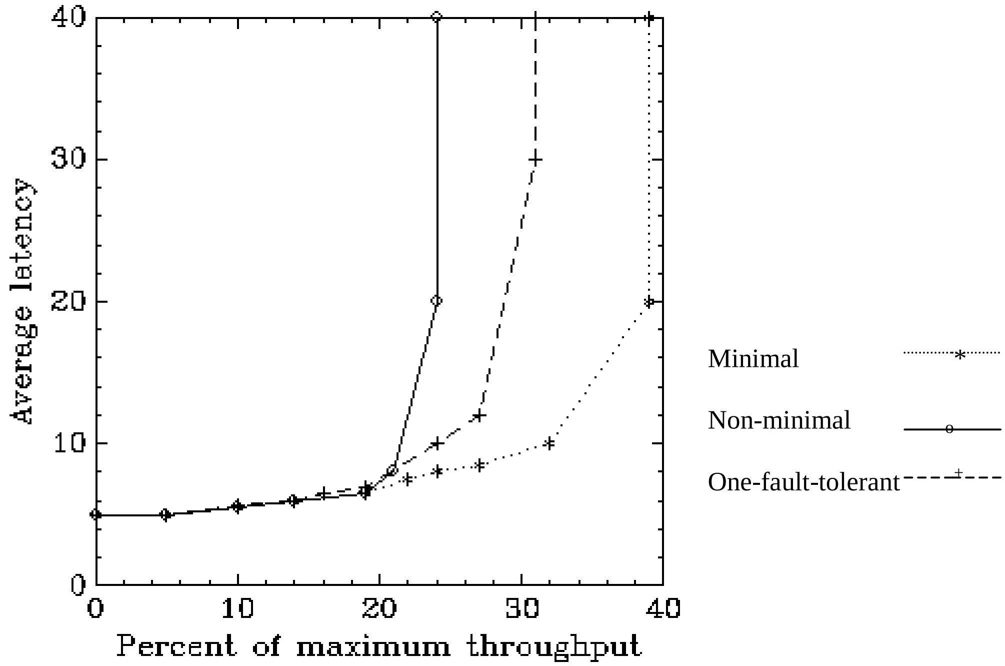

Figure 6. Comparison of the minimal, non-minimal and one-fault-tolerant versions of the negative-first routing algorithm for matrix-transpose traffic.

3.4 Review

The paper presented a modification of the *negative first* routing algorithm for meshes and proved that the resulting algorithm is one-fault tolerant. The algorithm is also deadlock and livelock free. The paper also presented extensions of the algorithm for multiple dimensions. As the dimensions increase the fault tolerance of the algorithm also increases. The most attractive feature of the algorithm is that it does not use virtual channels (or can be implemented on just one set of virtual channels). Virtual channels are not free. They add to the cost in terms of extra control lines, extra buffer space and additional switching hardware. This algorithm could have been used (instead of dimension order routing) on the last set of channels, in the solution sketched in section 2. In that case the solution would have been very adaptive as well as provably fault tolerant. In isolation, this algorithm inherits the inefficiencies of the *negative first* algorithm and leads to load imbalance (it tends to route the traffic into the corners). With this algorithm the network saturates at around 40% even for uniform traffic, while the algorithm is section 2 saturates at reaches 70% even for non-uniform traffic patterns. The effect of the complexity of algorithm on the router delay and architecture is also not clear. The algorithm is also non-minimal. As the simulations show this has a very bad effect on the latency. It is not clear how this could be modified so that the algorithm behaves minimally in the non-faulty case and routes non-minimally only in the presence of faults. The authors assume a message level retransmission protocol for error recovery, which can be quite expensive and can be replaced with flit level error recovery, as we shall see in the next paper.

4. Reliable Router: Dally et al. [2]

We finally look at one implementation of the reliable router [2], an adaptive fault tolerant switching element. The reliable router (RR) exploits adaptive routing for both performance and reliability purposes. It also has mechanisms for continuous link monitoring and link-level retransmission when a link parity error is detected. It employs a forwarding protocol at the flit level that facilitates packet reconstruction and duplicate detection at the receiving end when a fault occurs. This protocol is called Unique Token protocol (UTP).

The coupling of these features (adaptive routing, link monitoring, link-level retransmission and the UTP) enable the RR to handle a single node or link failure anywhere in the network without interruption of service.

4.1 Fault Tolerance and Adaptive Routing

The RR routing algorithm minimizes resource requirements and message state by using Duato's algorithm. The fault-handling properties of the routing algorithms are decoupled from the adaptive properties by using a different set of virtual channels and a different adaptive algorithm – *the turn model* – for fault handling. One can think of a network of RRs as the superposition of three separate virtual networks.

A minimally adaptive network: A packet using this network is able to route to any productive channel – a channel that will bring it closer to its destination. This virtual network tries to exploit adaptivity for performance reasons and is susceptible to deadlocks. The RR allocates two virtual channels per physical link to the adaptive network.

A dimension-ordered network: Packets in this network are routed in strict dimension order. Dimension ordered routing is provably deadlock free. This virtual network exists in order to break deadlocks introduced in the previous network as suggested by Duato's algorithm for breaking a deadlock. The RR allocates two virtual channels per physical link to this dimension ordered network.

A fault-handling network: This network permits non-minimal adaptive steps and it is used to exploit the fault handling properties of adaptive routing. The RR allocates one channel per physical link to this network.

All three virtual networks share the same physical network by using different sets of virtual channels. So there are 5 virtual channels per physical link. The routing algorithm consists of three separate computations occurring in parallel and takes one clock cycle. Each computation results in a next step virtual channel from one of the three virtual networks describes above.

Adaptive computation: The switching node attempts to find a non-busy minimally adaptive channel. If such a channel is found, then the message will use it as the next step of its path.

Dimension ordered Computation: If no minimal adaptive channel is available, the packet is routed to the unique dimension ordered channel corresponding to its current position and its destination. If this channel is busy, the packet is blocked for one cycle and tries again to find a channel using the adaptive computation.

Fault-Handling Computation: If the dimension ordered channel is faulty, the packet is routed to a fault-handling channel. This can be any channel, productive or unproductive, except for a channel that will cause the message to make a 180-degree turn. After the packet has been forwarded to the next switching node the algorithm either reverts to picking channels from the adaptive computation or sticks to fault-handling computation channels only. The idea is that if the packet ever misrouted from the *y* dimension in the dimension ordered network, it cannot come back to it else it will break the dimension order.

4.2 Router Architecture

The reliable router is designed for two-dimensional mesh topologies. Its organization is shown in Figure 7. There is one input controller and one output controller for every direction. Moreover, there is a processor input/output and a diagnostic input/output. Communication between an input and an output port occurs through a crossbar switch. The switch is a full crossbar and allows each input controller to connect to every output controller.

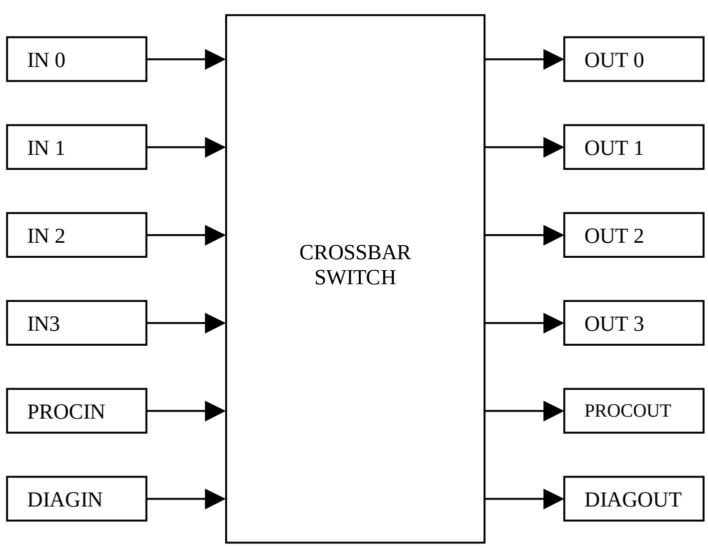

Figure 7. Organization of the Reliable Router

4.2.1 Major Object Types

Three types of objects are handled by the architecture. The packet is the main unit of information exchange between the sending and receiving end. The RR can handle packets of arbitrary size. Virtual channels are allocated on a packet basis. Every packet is broken into 75-bits flits. Buffering, forwarding and flow control within the system is performed at the flit level (wormhole routing). Crossbar bandwidth and the physical channels are also allocated at the flit level. The first flit of each packet called head flit and contains the address of the packet destination. The format of RR head flit is shown in Table 1. Subsequent flits are of type data and carry user data. The final flit of the packet if of type tail and marks the end of packet. There is also a flit of type token that is injected at the end of each a packet to implement the UTP. Each flit contains 64bits of user data, 8 byte-parity bits for end-to-end error detection and 3 bits that indicate flit type.

Bit Field	63:12	11	10	9:5	4:0
Contents	User Info	Priority	Diagnostic	Address in y	Address in x

Table 1. Head Flit Format

Flits are further broken into frames so that they can be transmitted across physical channel links, which are only 23 bits wide. Each flit is decomposed into four separate frames shown in table 2.

Bit field	22	21	20:18	17	16	15:0
Frame 0	PE	USR0	VCI	BP1	BP0	Data [15:0]
Frame 1	Copied Kind		Copied VCI	BP3	BP2	Data [31:16]
Frame 2	U/D	USR1	Kind	BP5	BP4	Data [47:32]
Frame 3	Freed			BP7	BP6	Data [63:48]

Table 2. Frame Format

4.2.2 Functional Description

Most of the functionality in the router has been pushed into the input controller. The input controller supports five separate virtual channels with decoupled resources. The functionality of the input controller can be summarized as follows:

- It buffers flits for each virtual channel. The flits are buffered in 16-flits deep FIFO, which also has some special state and functionality to implement backtracking and retransmission in case of fault.
- It computes the next step route of each packet based on head flit information and current output virtual channel state. The route is stored in dedicated registers so that it can be used by subsequent data flits.
- It picks one of the five virtual channels to drive a flit across the crossbar.

- It keeps track of the virtual channel buffer size in the receiving node and allows transmission across the crossbar only if the virtual channel buffer at the receiving node has space. The protocol followed for getting the information about the receiving node will be described later.

4.3 The Unique Token Protocol

Reliable Router implements link level retransmission in combination with unique-token protocol to guarantee fault-tolerant exactly-once delivery of all packets in the network. This link-level protocol offers significant advantages over end-to-end protocols because it does not require acknowledgement packets and does not keep copies for possible retransmissions at the packet source. In this way, effective network bandwidth is increased and storage requirements at the nodes decrease. These properties allow the protocol resources to scale linearly with the number of network nodes as opposed to end-to-end protocols.

Packet forwarding under UTP ensures that at least two copies exist in the path between the source node and the destination node at all times. This is achieved by first copying the packet forward one node and then allowing the release of the storage in the rearmost node. When the packet is first injected into the network, a token is injected right behind the packet. The nodes give up their copies of the packet only after the token goes through.

The actual implementation of the UTP occurs at the flit level rather than at the packet level. In order to maintain two copies of flits at all times within the network flow control information must make two steps to the back using separate flow control paths. This is shown in figure 9. Let us assume that node C copies a flit across to node D. It sends a copied message to node B using the copied kind and copied VCI fields shown in Table 2. Node B will use this information to invalidate its own copy of the flit. Moreover, reception of a copied field in an incoming frame will make node B send a freed message to node A using the Freed field on outgoing frame. Node A will receive the freed information and use it to decrement the appropriate counter, indicating that the neighboring node has one flit less in that flit queue.

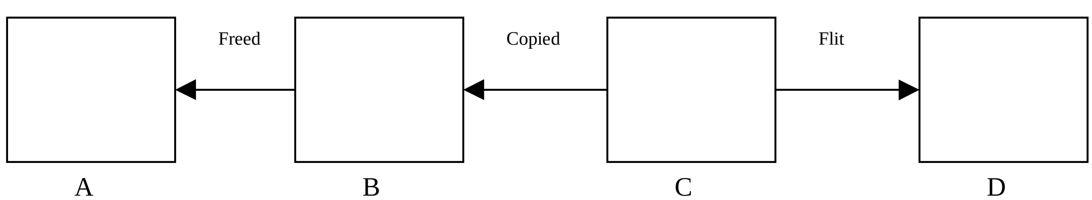

Figure 9: Flow Control in the Reliable Router

Fault Handling: When a node in the network fails, communication between the advance and rear copies of the packet may be severed. Each copy now must make its way to the destination without knowing the fate of the other copy. When packets arrive at the destination they must be marked in such a way so that the destination knows that it needs to look for duplicates. For this reason, two types of token are defined: Unique and Replica. If the network needs to use multiple paths while forwarding the packet, the token is changed to type Replica for all copies of the packet. After the token is changed, forwarding proceeds in the usual way of keeping two copies of the packet per path. Such an example is shown in figure 10. Due to a faulty link, communication between two copies of the packet has been broken. As a result, each copy changes it token to Replica and proceeds to destination using different paths. Every node needs a copy of the head flit to ensure retransmission of the partial packet when a link fails after only part of a packet has been transmitted to the next node. The head flit of the trailing piece if a partial packet is tagged as a special kind of head flit, called a *head:restart* flit as opposed to *head:original* flit. This is necessary for the destination to reconstruct the original packet.

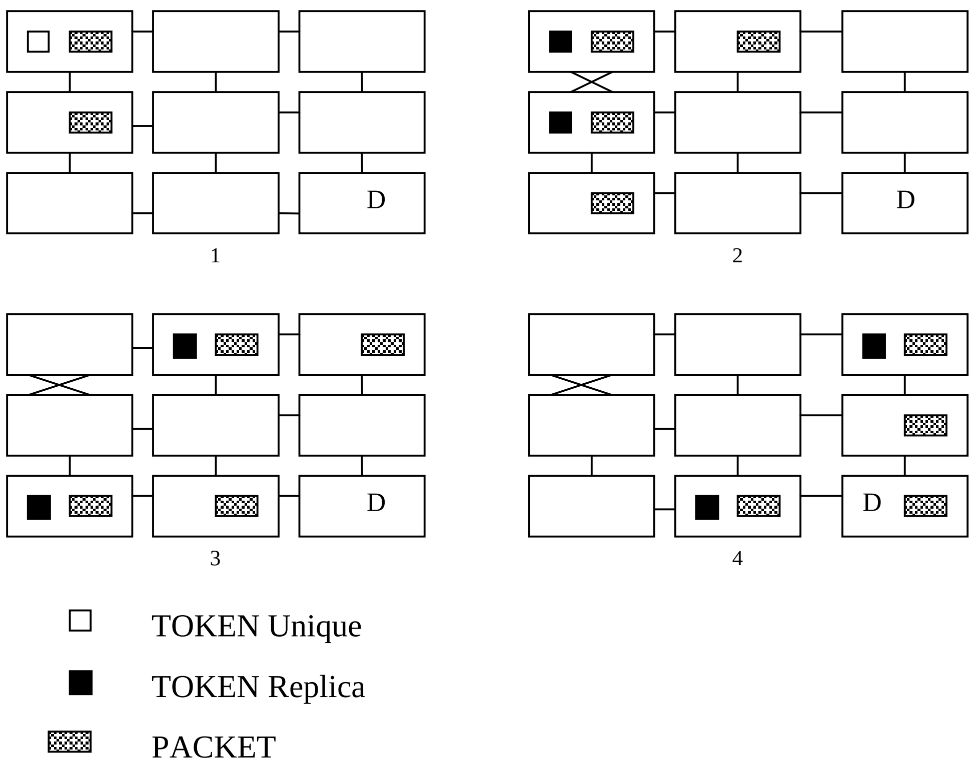

Figure 10: Fault Handling under the UTP

4.4 Review

The Reliable Router provides reliable and high performance communication between nodes of parallel computers. It uses a simple and efficient adaptive routing algorithm with minimal resource requirements. It is not very clear which turn model is used by the fault-handling network. Also the fault tolerance of network of RRs is not discussed and it can be seen that the network is not provably fault tolerant. The *turn model* (as noted in section 3) is not adaptive for some cases. If the packet happens to hit a fault for one of those routes, it will fail to deliver. The design could be changed to use the modified *negative first* algorithm proposed in section 3 to make it provably fault tolerant. It is not clear whether this can be accomplished while maintaining the tight clock constraints. There are various other techniques proposed in the paper for issues like clock skew, pin constraints, which we have not gone into. It would be interesting to do throughput and latency characterization on a network of RRs as the channel assignment is quite complex and uses a combination of various algorithms. The paper is instructive in the level of detail it provides on the architecture and the low level protocol. The unique token protocol provides a storage and bandwidth efficient way to recover from link and node errors.

5. Conclusion

We have reviewed three papers related to adaptive fault tolerant routing on meshes. Two of them [1, 4] propose high level algorithms to do adaptive routing, while the third [2] provides the low level architecture and protocol details of one fault tolerant implementation. The first paper [1] proposes an adaptive algorithm using dimension reversals and virtual channels. The second paper [4] does not use virtual channels and modifies the *negative first* algorithm (based on the *turn model*) to make it fault tolerant. Both the papers provide simulation results to characterize the performance of their algorithms. It is difficult to compare the two algorithms because the traffic patterns and the mesh topologies used in simulations are different but it is clear that the algorithm presented in section 3 cannot be used in isolation as it saturates at very low traffic volume. It could potentially be used on one set of virtual channels to provide fault tolerance while the other sets do more balanced adaptive routing. The third paper gives a detailed description of the architecture and protocol of a router that implements adaptive fault tolerant routing. It implements three different routing algorithms on disjoint sets of virtual channels, one for adaptive routing (minimal), the second for deadlock avoidance (dimension ordered) and third for fault tolerance (turn model).

References

[1] W. J. Dally and H. Aoki, “Deadlock –free adaptive routing in multicomputer networks using virtual channels”, *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 4, pp. 466-475, April 1993.
[2] W. J. Dally, L. R. Dennison, D. Harris, K. Kan, and T. Xanthopoulos, “Architecture and implementation of the reliable router”, *Proceedings of Hot Interconnects Symposium II*, August 1994.
[3] J. Duato, S. Yalamanchili, and L. Ni, “Interconnection networks, an engineering approach”, IEEE Computer Society, 1997**.**
[4] C. J. Glass and L. M. Ni, “Fault tolerant routing in meshes”, *Proceedings of the Twenty-Third International Symposium on Fault-Tolerant Computing*, pp 240 – 249, Aug. 1993.
[5] C. J. Glass and L. M. Ni, “The turn model for adaptive routing”, Proceedings of the 19th International Symposium on Computer Architecture, pp. 278-287, May 1992.