To appear in *Advanced Research in VLSI and Parallel Systems 1992.* Thomas F. Knight, Jr. and John Savage, editors. MIT press, March 1992.

Fault Tolerance and Performance of Multipath Multistage Interconnection Networks

Frederic Chong, Eran Egozy, and André DeHon
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
545 Technology Square
Cambridge, MA 02139
ftchong@ai.mit.edu, eran@ai.mit.edu, andre@ai.mit.edu

Abstract

In building a multiprocessor system, we can minimize the system's mean time to failure by providing an architecture resilient to component faults. We compare the fault tolerance and performance characteristics of various fault-tolerant multistage interconnection networks. We primarily focus on networks composed of *dilated* routing components. A dilated router features redundant outputs in each logical direction, and can thus be used to construct *multipath* networks. Multipath networks have multiple paths from any input to any output. An *interwired* multipath network disperses its routers' redundant outputs to input ports of physically distinct components. We introduce a deterministic wiring scheme for routing interwired networks that maximizes the number of routing paths available for each endpoint. We compare the deterministically-wired network to both randomly-interwired networks and non-interwired multipath networks. We characterize fault tolerance by measuring the probability of a single endpoint disconnection as faults accumulate in the network. Our performance simulations are based on traffic from shared memory applications and include barrier synchronization to expose the effects of localized performance degradation. We find that at a minor performance cost, deterministically-interwired networks are more fault tolerant than randomly-interwired networks and non-interwired networks.

1 Introduction

Sufficient fault tolerance is an important part of any practical, large-scale computer system. In this paper, we focus on selecting wiring schemes for Multistage Interconnection Networks (MINs) which offer the most fault tolerance and performance for a fixed amount of hardware.

We examine *interwired* and *non-interwired* multipath networks. An interwired network connects redundant outputs of each routing component to physically distinct components in the next stage of the network. Such interwiring was first used by Bassalygo and Pinsker [1]. Upfal [18] later used interwiring to form what he called *multibutterflies.* We introduce a deterministic algorithm for interwiring networks and compare it to the random interwiring methods presented by Leighton and Maggs [15]. We also examine replicated networks where multiple single-path networks are attached together at the endpoints to form a multipath network.

To measure fault tolerance, we use the simple fault metric of *completeness* coupled with random fault generation to probabilistically quantify the number of faults a given network can sustain. A network is said to be complete as long as there is some non-faulty path between every pair of network endpoints. This metric is conservative if the system can actually tolerate endpoint isolation or network partitioning. Nonetheless, it provides a simple decision test for comparing the effects of faults in various networks.

For our performance comparisons, we simulate message traffic over faulty and non-faulty networks to quantify the effective aggregate bandwidth of such networks in the presence of faults. We model several shared memory applications by simulating message distributions and processor synchronization.

Section 2 describes the basic networks which we study, as well as the routing models assumed. Section 3 details the various network wirings considered in this paper. Section 4 describes our fault tolerance experiments and Section 5 presents the data resulting from our experiments. Section 6 explains our performance modeling and Section 7 summarizes our performance results. We discuss related issues to this paper in Section 8 and draw final conclusions in Section 9.

2 Network and Routing Models

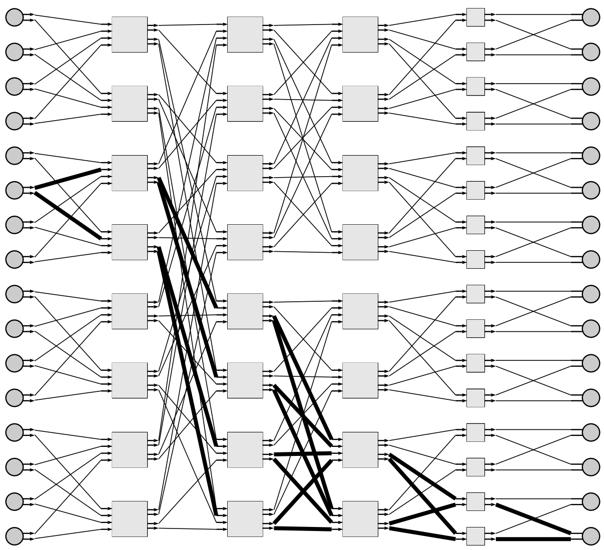
2.1 Networks

This paper focuses on MINs built from several stages of crossbar style routing elements. For simplicity, we will assume crossbar routing switches with an equal number of inputs and outputs. Each crossbar switch can connect any of its inputs to any of its outputs with the restriction that only one input can be connected to each output at any given moment. A particular crossbar router is characterized by both its *radix* and its *dilation*. The *radix* of a router is the number of distinct logical directions distinguished by the routing component. The *dilation* is the number of logically equivalent outputs in each distinct direction. Thus, with a dilation greater than one, redundant routing is provided in each direction. Figure 1 shows an example of such a MIN.

We further limit ourselves to multipath networks with redundant connections to each endpoint. The redundant endpoint connections make it possible to guarantee that no single fault will isolate the endpoint from the network. Figures 1 through 5 show MINs each of which has two connections from and to each endpoint.

Much previous work has been done on this class of networks. Kruskal and Snir [10] suggest using multiple copies of a MIN to provide fault tolerance. They also describe a network with dilated routing components and Grondalski [6] implements such a network. The Kappa network [9] also uses dilated routers to achieve fault-tolerant interconnect. Leighton and Maggs [15] suggest effective ways of wiring and using randomly-wired multibutterfly networks.

An alternative approach to achieving fault-tolerant MINs is to construct *extra-stage* networks with more switching stages than are actually required to uniquely specify a destination ([13], [3] *et. al.*). The set of routing specifications that reach the same physical destination defines a class of equivalent paths. So long as one path of each such class remains intact in a faulty extra-stage network, any endpoint will be able to successfully route to its destination. However, such schemes require the endpoint to specify a path through the network. The extra stages in these schemes also result in larger latencies than the corresponding multipath network, even in



A multipath MIN constructed from 4×2 (inputs$\times$radix) dilation-2 crossbars and 2×2 dilation-1 crossbars. Each of the 16 endpoints has two inputs and outputs for fault tolerance. Similarly, the routers each have two outputs in each of their two logical output directions. As a result, there are many paths between each pair of network endpoints. Paths between endpoint 6 and endpoint 16 are shown in bold. This network was wired using the deterministic interwiring algorithm described in Section 3.4.

Figure 1: 16×16 Multipath Network

the absence of faults. We do not include this class of networks in our analysis of multipath networks.

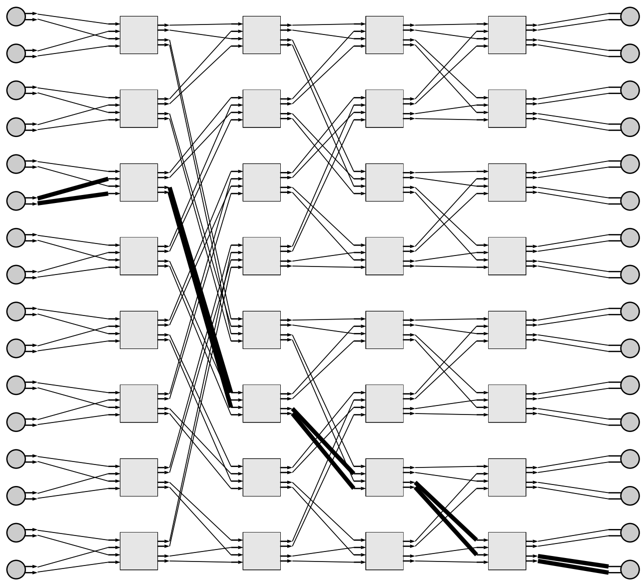
2.2 Routing

Our fault tolerance results described below are independent of the details of routing and fault-identification. That is, routing can be circuit-switched or packet-switched using any number of routing strategies. The fault tolerance results only depend on network topology and the routers' ability to use their redundant connection in each logical direction to avoid faults.

Our performance modeling assumes additional implementation details. The routers perform source-responsible, pipelined, circuit-switched routing. Data are pipelined between routers in adjacent stages of the network to improve throughput and available bandwidth. The circuit-switched, source-responsible protocol is simple and can be implemented very efficiently in VLSI hardware. The routing protocol assumed is described in detail in [5]. We have used this routing protocol in the VLSI routing component RN1 [17]. An RN1 routing component can be configured to act as a single 8-input, radix-4, dilation-2 routing component or as a pair of independent 4-input, radix-4, dilation-1 routing components.

3 Network Wiring

We consider four basic wiring strategies for building multipath multistage routing networks. In this section, we describe these strategies and show an example of each



A dilated, non-interwired four-stage network connecting 16 endpoints. Each component is a 4×2, dilation-2 crossbar.

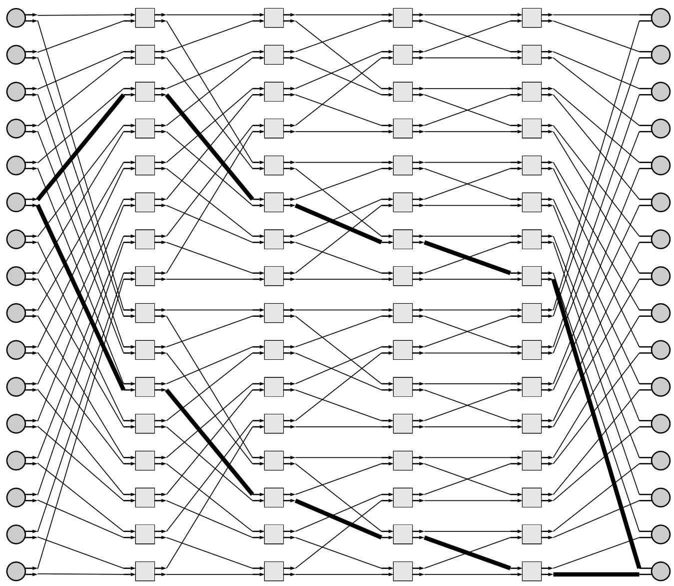
Figure 2: Dilated Non-Interwired Network

network.

As with all bidelta [11] networks, each stage in these multipath networks routes by successively subdividing the set of possible destinations into a number of *equivalence classes* equal to the radix of the routing components. For example, consider a radix-4 network. When connections enter the network, any input can reach any destination. The first stage of routing components divides this class into 4 different equivalence classes based on desired destination. Each succeeding network stage further subdivides a previous stage's equivalence classes into 4 more equivalence classes. When there is a single destination in each equivalence class, the network has uniquely determined the desired destination and can connect to the destination endpoints. This successive subdivision can be easily seen in the networks shown in Figures 1 through 5.

3.1 Dilated Non-Interwired Network

The dilated networks described in [10] [6] connect all outputs in a given logical direction to the same physical routing component in the subsequent stage of the network. The topology is thus identical to the corresponding non-dilated bidelta network [11]. Such networks gain performance by having multiple paths. These networks can also tolerate some faults in the wiring between components. Unfortunately, these networks are just as susceptible to component failures as non-multipath networks. An example of a dilated non-interwired network composed of 4×2, dilation-2 routers is given in Figure 2.



Two non-dilated four-stage networks connecting 16 endpoints are attached together at the endpoints. Each component is a 2×2, dilation-1 crossbar.

Figure 3: Replicated Network

3.2 Replicated Network

In a replicated network, such as the ones proposed by Kruskal and Snir [10], multiple single-path networks are used to achieve fault tolerance. Each single-path network is a bidelta network using dilation-1 crossbar switches. Each endpoint gets one connection to and from each of the single-path networks. An endpoint chooses which network to use when it needs to make a connection. As long as each pair of endpoints is connected by a path in at least one of the networks, the replicated network will remain complete. Replicated networks are used in the telecommunications field to minimize contention and increase available bandwidth over single-path networks [7]. A replicated network, where each subnetwork is constructed from 2×2 routing components, is shown in Figure 3.

3.3 Randomly-Interwired Network

An interwired network takes advantage of the redundant outputs of dilated routing components and connects them to physically distinct components in the next stage of the network. Bassalygo and Pinsker [1] used such interwiring to construct the first non-blocking networks of size $O(N \log N)$ and depth $O(\log N)$. Upfal [18] later used interwiring to form networks he called *multibutterflies*. Leighton and Maggs [15] further analyzed the high *group expansion* (or *α-β expansion*) of these randomly-interwired networks. Group expansion is a description of the degree to which any subset of components in one stage will fan out into the next stage. More formally, any subset of α components from one stage must connect to at least $\alpha \times \beta$ components in the next stage, where β is the expansion factor and $\alpha \times \beta$ is less than the number of components in each stage. Good fault tolerance is an inherent property of high group expansion [15].

Deterministic construction of graphs with high group expansion is a difficult problem. Recent work [8] describes a deterministic algorithm based upon Ramanujan graphs which, unfortunately, does not provide as much expansion as random multibutterfly generation. A randomly generated multibutterfly has a high probability of possessing high group expansion. This high probability results from the high fraction of networks in the class of multibutterflies that have high group expansion. Consequently, it is common practice to randomly generate multibutterfly networks until one is found with the desired group expansion.

We use a heuristic algorithm based on the random interwiring ideas presented in [15]. Specifically, we constrain every router to send its redundant output paths to physically distinct components in the next stage, thus improving group expansion over a purely random interwiring scheme. Figure 4 shows the algorithm used to generate a randomly interwired network.

We use dilation-1 routers in the final stage of the network to achieve a high degree of fault tolerance. The two ends of the network are the weakest links to achieving a high degree of fault tolerance since fewer alternative components are available in these stages to make any given connection. To maximize fault tolerance, we must maximize the number of distinct components connected to each endpoint. The heuristic described above guarantees that the input connections from any endpoint connect to distinct components. Similarly, using dilation-1 routers in the final stage maximizes the number of distinct components that provide outputs from the network to each endpoint. Unfortunately, the cost of using dilation-1 routers in the last stage is a small performance decrease (see Section 7). However, if dilation-1 routers were not used, a single component failure in the network could render the network incomplete.

An example of a randomly-interwired four-stage network using radix-2 routing components is shown in Figure 5.

3.4 Deterministically-Interwired Network

Although random interwiring provides for a reasonable degree of group expansion, it does not guarantee each endpoint the maximum number of redundant paths through the network. We have developed an algorithm that provides the maximum amount of *path expansion* through the network at the expense of some group expansion. Path expansion refers to the number of distinct components in the network that any one endpoint can reach.

To achieve maximum path expansion, we connect the network with the algorithm listed in Figure 6. The paths from any input to any output may fanout by no more than a factor of d, the dilation of the routers, at each stage. This fanout may also become no larger than the size of the routing equivalence classes at that stage. The routine *groupsz* returns the maximum fanout size allowed by both of these factors. Each stage is partitioned into *fanout classes* of this size, which are then used to calculate network wiring.

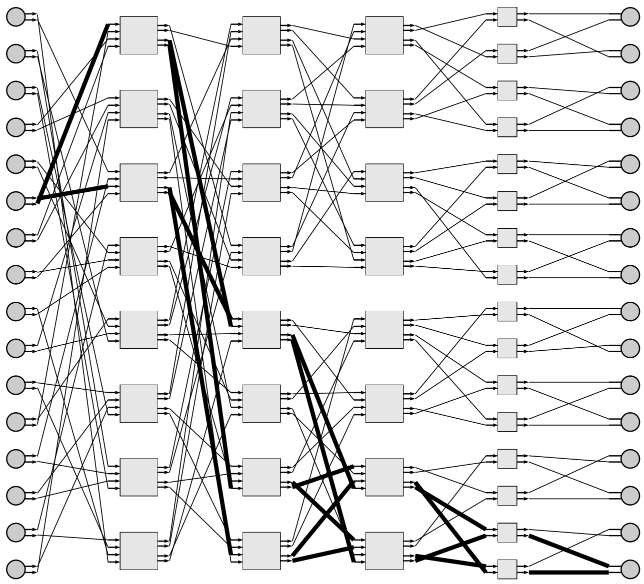
As with the randomly-interwired network, the last stage is composed of dilation-1 routers to increase fault tolerance. Figure 1 (Section 2) shows a deterministically-interwired network composed of radix-2 routers.

```
▷ in_set contains all the input ports of a single equivalence class in the next stage.
▷ connections is an array matching in_ports and out_ports, initially empty.
▷ out_ports_list lists the output ports of a single equivalence class in the currect stage.
wire_eq_class(in_set, connections, out_ports_list)
foreach out_port
    in_port ← choose and remove a random input port from in_set
    while(connected(router#(in_port), router#(out_port), connections))
        put in_port back in in_set
        in_port ← choose and remove a random input port from in_set
    connect(in_port, out_port, connections)
return(connections)

connected(in_router, out_router, connections_array)
if in_router is in already connected to out_router
    return(true)
else return(false)
```

This algorithm randomly interwires an equivalence class. To interwire a whole stage, the algorithm is repeated for each class (boundary cases omitted for clarity).

Figure 4: Pseudo-code for Random Interwiring



A randomly-interwired four-stage network connecting 16 endpoints. Each component in the first three stages is a 4×2, dilation-2 crossbar. To prevent any single component from being in an endpoint's critical path, the last stage is composed of 2×2, dilation-1 crossbars.

Figure 5: Randomly-Interwired Network

▷ Returns the next-stage router to which to wire for maximum path expansion
wire_to_port(n,d,s) ▷ n=router number, d=dilated port number, s=router stage
$outgrpsz \leftarrow$ **groupsz**(s)
$ingrpsz \leftarrow$ **groupsz**$(s+1)$
$eq\_start \leftarrow ingrpsz \times \lfloor n/(radix \times dilation \times outgrpsz) \rfloor$ ▷ offset to beginning of fanout class
$eq\_router \leftarrow ((n \times dilation + d) \bmod ingrpsz)$ ▷ offset to specific chip within fanout class
return$(eq_start + eq_router)$

▷ Calculates size of fan-out class
groupsz(s)
$expansion \leftarrow dilation^{s}$ ▷ maximum fanout due to dilation
$eq\_class \leftarrow radix^{stages-s}$ ▷ equivalence class size
return(min$(expansion,\ eq_class)$)

An algorithm designed to maximize path expansion. Each endpoint will have the maximum number of redundant paths possible through this type of network.

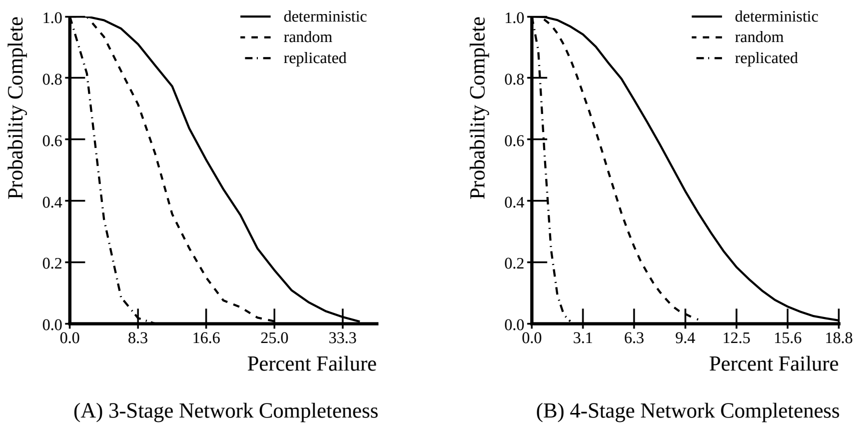
Figure 6: Pseudo-code for Deterministic Interwiring

4 Fault Modeling

We concern ourselves only with complete component failures rather than to individual wire failures. We characterize network completeness through Monte Carlo simulation. For each trial, we randomly add component faults until the network is incomplete. Results are tabulated to approximate the probability of network completeness for each fault level. We also derive the expected number of faults each network can tolerate.

Because the routing components in the final stage of our interwired networks are half the size of routers in the previous stages, we assign two such routers to one physical component package and label both routers faulty if the physical component is chosen to be faulty. Furthermore, the two routers are assigned so that removing any such pair will not cut off an endpoint. We make this assignment so that fault increments will be of constant hardware size. This assignment also simulates how the pair of 4×4, dilation-1 routers in an RN1 routing component may be assigned.

We generated three-stage and four-stage networks for each of the four types of networks described above, each connecting 64 and 256 endpoint nodes respectively. Each endpoint has two connections to and from the network to provide for the minimal amount of redundancy necessary to achieve fault tolerance. Every network uses radix-4 routers of either dilation-1 or dilation-2 and hence could be implemented using the RN1 component. Thus, all the networks with a given number of stages contain the same amount of hardware. Network organization and component configuration are solely accountable for the fault tolerance and performance differences of these networks.



The probability that a network with a given number of faults is complete for the replicated, the randomly-interwired, and the deterministically-interwired 3-stage and 4-stage networks. (A) Each 3-stage network uses 48 radix-4 components that interconnect 64 endpoints. (B) Each 4-stage network uses 256 radix-4 components that interconnect 256 endpoints.

Figure 7: Completeness of (A) 3-stage and (B) 4-stage Multipath Networks

5 Fault Tolerance of Networks

For each network, we plot the probability of the network remaining complete for a given number of uniformly distributed random faults. Since the non-dilated routers of the replicated network are half the size of dilated routers in the rest of the networks, we normalize the component count on the graph so that the same amount of hardware is faulty at each fault level. Results for the three-stage and four-stage networks are shown in Figure 7. The expected number of faults that each network can tolerate is summarized in Table 1.

Dilated Non-Interwired Network Because every component in this network is in some critical path between two endpoints, it cannot sustain a single component fault and is tolerant only to wire failures. It is easy to see that this network will tolerate exactly as many wire faults as an equivalently-sized replicated network. Both networks have exactly as many redundant paths between each pair of successive stages as there are redundant connections to each endpoint (see Figures 2 and 3). The dilated non-interwired network always has fewer connections between internal routing stages than the interwired networks and is thus more susceptible to wire faults (see Figures 1, 2, and 5).

Replicated Network For each pair of endpoints, the replicated network provides exactly as many paths through the network as there are connections to each endpoint. This provides a degree of fault tolerance as shown in Figure 7 and Table 1.

Randomly-Interwired Network Since these networks are non-unique, ten random networks were generated, and the one with the highest fault-tolerant characteristics was selected for comparison. The differences in expected number of tolerable

Network Stages	Wiring	Total # Comp.	Test Trials	Expected Failure Tolerated # Faults	% Network	Error Bound # Faults
3	Non-Interwired	48	—	0	0.0%	0
3	Replicated	96	2500	3.1	3.1%	0.025
3	Random	48	1000	5.0	10%	0.063
3	Deterministic	48	1000	8.1	16%	0.079
4	Non-interwired	256	—	0	0.0%	0
4	Replicated	512	5000	4.1	0.8%	0.024
4	Random	256	5000	11.8	4.6%	0.075
4	Deterministic	256	5000	22.6	8.8%	0.130

The above table shows the expected number of faults each network can tolerate while remaining complete. Each network was fault tested by the method described in section 4 for the indicated number of trials.

Table 1: Fault Tolerance of Various Multipath Networks

faults between the best and worst of the ten three-stage and four-stage networks is 0.3 faults and 0.2 faults, respectively. Because the difference in expected number of faults between the random networks is small, the choice of one particular network over another makes no difference when comparing randomly-interwired networks against other wiring schemes. When actually implementing a network, one is certainly free to generate a large number of random networks and select the best one.

Randomly-interwired networks provide significantly more tolerance to component faults than replicated networks, as shown in Figure 7 and Table 1. The expansive properties allow the number of paths between each pair of endpoints to increase towards the middle of the network. In the first and final routing stages, the number of components available to make a connection between any pair of endpoints is still limited by the number of redundant connections to each endpoint. However, within the network, more components are available in each routing stage and more wires are available between every pair of routing stages.

Deterministically-Interwired Network This network achieves the highest fault tolerance of all the networks presented here for two reasons. First, the grouping of endpoint connections makes it less likely that losing a small number of routers from the first stage will result in an incomplete network. Second, the deterministic wiring guarantees maximal path expansion. The deterministic wiring thus guarantees that a maximum number of components in each stage are available for routing between each pair of endpoints. Similarly, the maximum number of wires between each pair of stages is available for connecting any pair of endpoints.

6 Performance Modeling

Given the completeness characteristics of the networks we have examined, we are interested in how their performance degrades in the presence of faults. For randomly-wired multibutterflies, Leighton, Lisinski and Maggs [14] found promising results from simulating 1024 node networks under some simple models of network architecture and utilization.

We investigate the empirical performance of several multipath networks under a specific architectural model. We attempt to derive efficient, yet realistic, simulations of network utilization by modeling several 64-processor, shared memory applications. As our performance metrics, we measure the I/O bandwidth utilization and average message latencies of these applications. We also simulate processor synchronization to expose the effects of localized network performance degradation.

6.1 Performance Metrics

We calculate average I/O bandwidth utilization by observing the percentage of router cycles for which processor nodes are successfully receiving or replying to messages. The performance of a network is measured by the amount of useful communication it provides. For our analysis, useful communication occurs when a processor node is receiving or replying to a message. An I/O bandwidth utilization of 100 percent indicates that the processor nodes are receiving maximum usable performance from the network. Any more traffic the network can handle will not result in any increase in useful work. Since we model an architecture in which memory is contained within each processing node, our I/O bandwidth metric is analogous to memory bandwidth metrics used by [14] and [12].

We also calculate average message latencies. Latencies are measured from time of message injection into the network until time of message completion, which includes acknowledgment from the receiver. Note that our latencies vary with the message lengths of each application. Also realize that a processor node does not necessarily suffer the full latencies given. For certain types of messages, some useful work could begin before the entire message is received.

6.2 Modeling Shared Memory Applications

Our goal is to provide a realistic model of network utilization that can be used to compare many different networks and parameters. To keep simulation time tractable, we use a variant of uniform traffic. Our simulation sends messages to random destinations in a uniform distribution. However, message lengths are randomly generated according to distributions derived from specific parallel applications. These application were taken from caching studies done in [2]. These studies simulate a shared memory architecture with coherent caches at each processing node.

Cache-coherent shared memory systems utilize a small set of message types, each of a fixed length. The messages sent through the network read cache lines, write cache lines, and maintain cache coherency. A cache-line read, for example, requires an 8-byte read request followed by a reply containing the data. The reply consists of an 8-byte header followed by 16 bytes representing the desired cache line.

6.3 Synchronization

Our performance simulation includes barrier synchronization to eliminate *skew*. Our simulation models applications which assume some degree of synchronization between the multiple processor nodes of the system. When a simulation violates this assumption, results are skewed. We prevent this skew by performing periodic barrier synchronization according to grain sizes estimated for each application.

It is important to eliminate simulation skew because it can mask the effects of localized network degradation. Analytic models suggest that faults and congestion may severely affect the performance observed by specific destination nodes while leaving others largely unaffected [12]. Without synchronization, such localized degradation would be lost in the *average* I/O bandwidth utilization. Modeling synchronization, however, forces all processors to wait for those falling behind, resulting in a more realistic decrease in I/O bandwidth utilization.

6.4 Application Descriptions

Message distributions were taken from trace simulations of four applications: SIMPLE, SPEECH, FFT, and WEATHER. The trace simulations involved the following system configuration: 64 processors, full-map directories, and 16-byte cache line. This configuration corresponds to 3-stage, radix-4 networks. SIMPLE models the hydrodynamic behavior of fluids using finite difference methods to solve the equations in two dimensions. SPEECH is the lexical decoding stage of a phonetically-based spoken language understanding system. It uses a variant of the Viterbi search algorithm. FFT is a radix-2 Fast Fourier Transform. WEATHER uses finite-difference methods to solve partial differential equations which model the atmosphere around the globe.

We also simulate a uniform message distribution, FLAT24. FLAT24 uses 24-byte messages and other simulation parameters which are similar to the messages and parameters of SIMPLE and SPEECH. FLAT24 shall serve as a basis for network comparison.

7 Performance Results

In this section we summarize our performance results for each network in the presence of faults. Performance was measured for complete networks only. For each fault level shown, multiple trials were run in which faults were randomly chosen. After fault insertion, applications were simulated on those networks which remained complete. Data shown represent the average I/O bandwidth utilization and latencies over those trials involving complete networks. These results do not represent the actual performance of these applications on hardware. Rather, our data provides a basis for network comparison by illustrating performance trends in the presence of faults.

First, we compare the performance of interwired and non-interwired dilated networks. Since the non-interwired network can tolerate no faults, we only compare the fault free case. To isolate the effects of interwiring, the deterministic and random networks presented use dilation-2 components in the last stage. We studied 3-stage non-interwired, randomly-interwired, and deterministically-interwired networks. I/O bandwidth utilization varied by less than 2 percent, a variation not significant for our simulation accuracy.

However, to avoid single-point disconnections in the last stage, the interwired networks we shall analyze for fault performance are constructed from dilation-1 components in the last stage. Although dilated components provide better performance, the use of these dilation-1 components substantially increases fault tolerance (see Section 3.3). For applications studied on 3-stage networks, the decrease in I/O bandwidth utilization was less than 6 percent.

Figure 8 details the fault performance of our applications on 3-stage, radix-4, networks which can withstand faults. The application FLAT24, shown as a solid line, is representative of graph trends and will be used in network comparisons.

Figure 9 compares the performance of those networks which can tolerate faults. The number of faults on the replicated network has been normalized to reflect an equivalent hardware loss. Each component in the replicated network is half the size of the components used in the other two networks. Our results show that deterministically and randomly-interwired networks deliver superior performance to the replicated network of equivalent hardware. The performance of the random network is slightly better than that of the deterministic network. However, recall that our figures are for complete networks only. For each fault level shown, the random networks have a lower probability of remaining complete than the deterministic networks.

8 Discussion

Although we have concentrated upon the fault tolerance of multipath networks, these networks also perform well under unbalanced loading. Intuitively, the effects hot-spots are very similar to component failures. Patterns such as matrix transpose have been shown perform well on randomly-wired multibutterflies by [14]. Further investigation of non-uniform loading is merited.

Additionally, our completeness metric assumes that a network is unusable when any input is disconnected from any output. This may not be true in the case of reconfigurable systems. In such systems, other metrics such as the number of surviving processors after network disconnection are important as well.

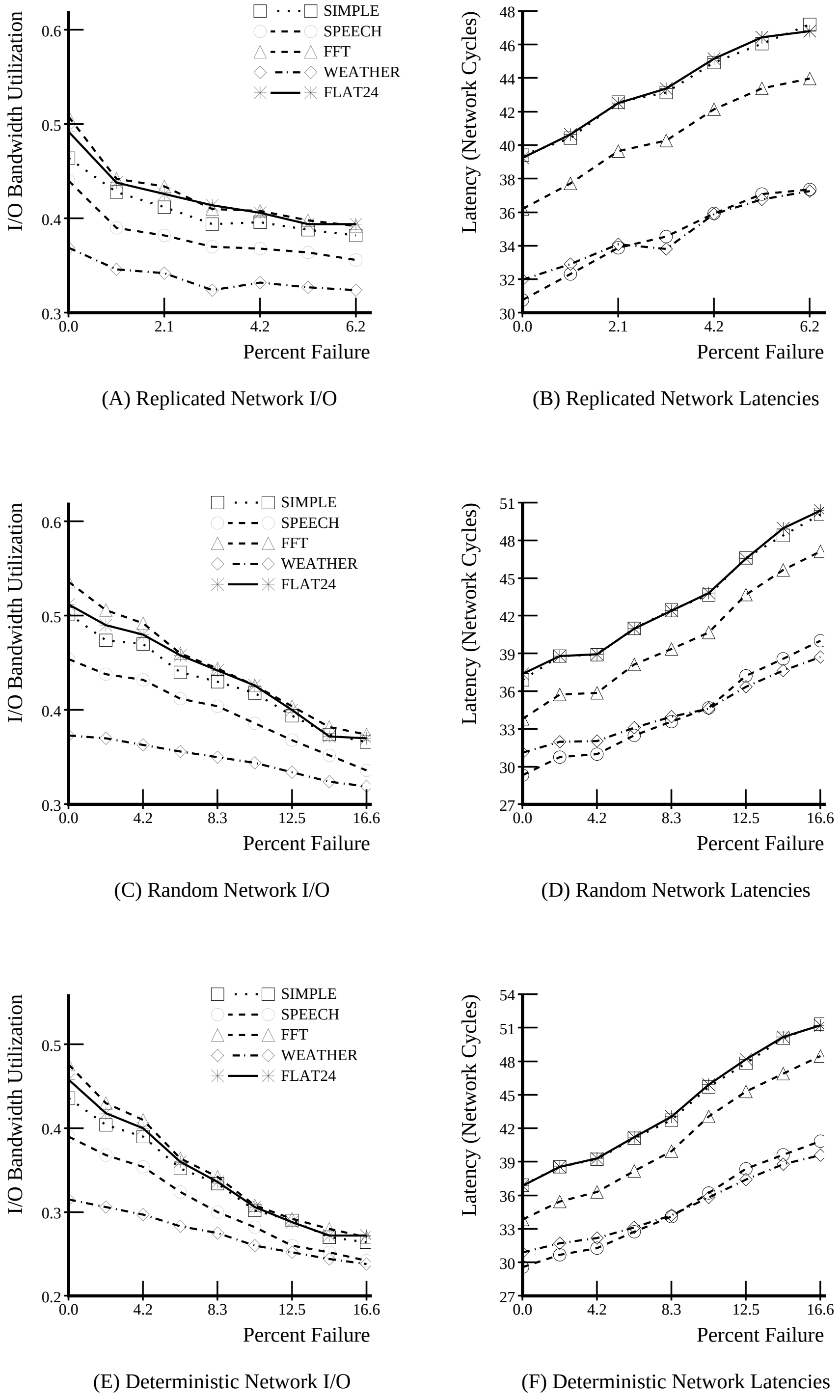
Fault tolerance becomes increasingly important as networks become large. The wiring techniques we have presented are applicable to a wider range of networks than traditional MINs. Fat-trees [16] [4] are an example of a highly scalable network which would benefit from the interwiring described here. As system size increases, a number of additional issues arise. We can use hybrid interwiring schemes which combine the high fault tolerance of deterministic wiring with the robust performance of random wiring. For instance, we can wire the the first portion of a network deterministically and the remainder randomly. The disparity between non-interwired networks and interwired networks also increases with system size. While non-interwired networks have a constant number of paths between endpoints, interwiring exploits dilation to create an exponentially increasing number of paths with each stage within the first half of the network.

9 Conclusion

We have compared the fault tolerance and performance of several multipath networks and have found a number of tradeoffs. The use of non-dilated components in the final stage of the network increases fault tolerance with some loss in performance. With a smaller decrease in performance, deterministic interwiring can tolerate more faults than random interwiring under our completeness metric. In general, interwired networks provide significantly more fault-tolerance than comparably-sized, non-interwired networks.

10 Acknowledgments

Much of the work from this paper grew from the ideas of our advisor, Tom Knight. Additionally, we would like to thank David Chaiken for providing the data describ-



(A) Replicated Network I/O

(B) Replicated Network Latencies

(C) Random Network I/O

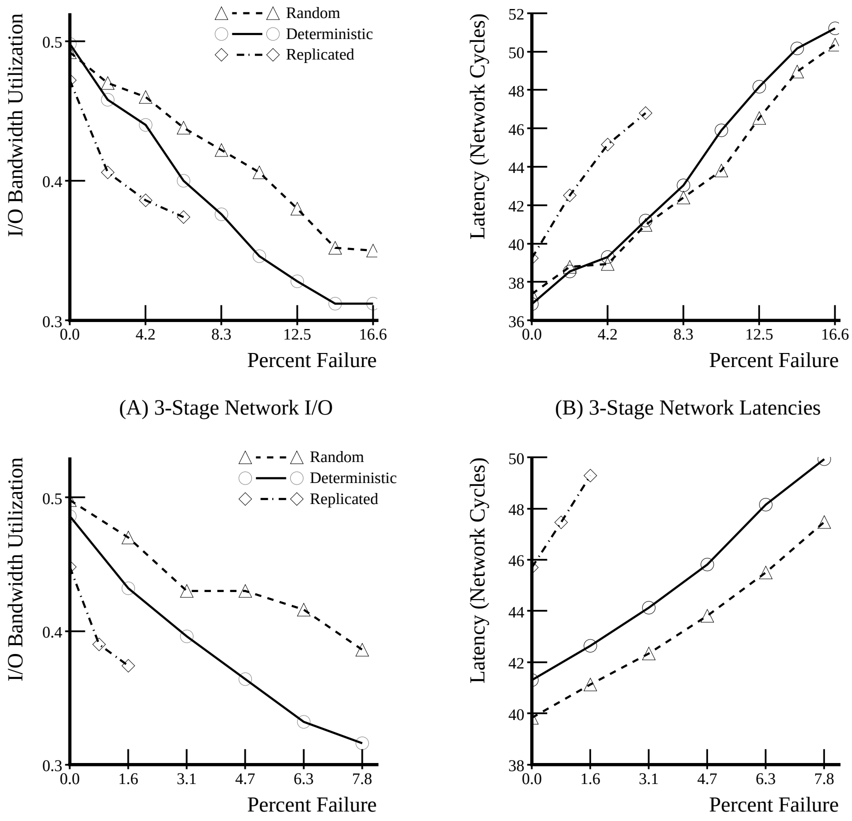
(D) Random Network Latencies

(E) Deterministic Network I/O

(F) Deterministic Network Latencies

I/O bandwidth utilization and latencies for applications on 3-stage replicated, random, and deterministic networks. The application FLAT24, shown as a solid line, is representative of graph trends and will be used for network comparison.

Figure 8: Applications on 3-stage Replicated, Random, and Deterministic Networks



(A) 3-Stage Network I/O

(B) 3-Stage Network Latencies

(C) 4-Stage Network I/O

(D) 4-Stage Network Latencies

Comparative I/O bandwidth utilization and latencies for 3-stage and 4-stage random, deterministic, and replicated networks on **FLAT24**. Recall from Table 1 that expected percentages of failure tolerated by replicated, random, and deterministic networks are, respectively: 3.1%, 10%, and 16% for 3-stages; and 0.8%, 4.6%, and 8.8% for 4-stages. Note that the performance degradation appears to level off because only complete networks are measured. Although the surviving networks suffer less degradation as percentage of failure increases, the number of surviving networks is becoming substantially smaller.

Figure 9: Comparative Performance of 3-Stage and 4-Stage Networks

ing the shared memory applications. We would also like to thank Mike Bolotski, David Chaiken, Steve Keckler, Bruce Maggs, Patrick Sobalvarro, Thomas Simon, and Deborah Wallach for their helpful comments.

This research is supported in part by the Defense Advanced Research Projects Agency under contract N00014-91-J-1698. This material is based upon work also supported under an Office of Naval Research Graduate Fellowship, a National Science Foundation Fellowship, and the MIT Undergraduate Research Opportunities Program.

References

[1] L. A. Bassalygo and M. S. Pinsker. Complexity of optimum nonblocking switching networks without reconnections. *Problems of Information Transmission*, 9:64–66, 1974.

[2] David Chaiken, Craig Fields, Kiyoshi Kurihara, and Anant Agarwal. Directory-Based Cache-Coherence in Large-Scale Multiprocessors. *IEEE Computer*, 23(6):41–58, June 1990.

[3] Chin Chi-Yuan and Kai Hwang. Connection principles for multipath packet switching networks. In *10th Annual Symposium on Computer Architecture*, pages 99–108, 1984.

[4] André DeHon, Thomas F. Knight Jr., and Henry Minsky. Fault-tolerant design for multistage routing networks. In *International Symposium on Shared Memory Multiprocessing.* Information Processing Society of Japan, April 1991.

[5] André DeHon, Thomas F. Knight Jr., and Henry Minsky. RNP: Fault tolerant routing protocol. Transit Note 41, MIT Artificial Intelligence Laboratory, March 1991.

[6] Robert Grondalski. A VLSI chip set for massively parallel architecture. In *IEEE International Solid-State Circuits Conference*, pages 198–199, 1987.

[7] Joseph Y. Hui. *Switching and Traffic Theory for Integrated Broadband Networks.* Kluwer Academic Publishers, Boston, 1990.

[8] Nabil Kahale. Better expansion for Ramanujan graphs. In *32nd Annual Symposium on Foundations of Computer Science*, pages 398–404. IEEE, October 1991.

[9] S.C. Kothari, G.M. Prabhu, and R. Roberts. The kappa network with fault-tolerant destination tag algorithm. *IEEE Transactions on Computers*, 37(5):612–617, May 1988.

[10] Clyde P. Kruskal and Marc Snir. The performance of multistage interconnection networks for multiprocessors. *IEEE transactions on Computers*, C-32(12):1091–1098, December 1983.

[11] Clyde P. Kruskal and Marc Snir. A unified theory of interconnection network structure. In *Theoretical Computer Science*, pages 75–94, 1986.

[12] Vijay P. Kumar and Andrew L. Reibman. Failure dependent performance analysis of a fault-tolerant multistage interconnection network. *IEEE Transactions on Computers*, 38(12):1703–1713, December 1989.

[13] Duncan Lawrie and Krishnan Padmanabhan. A class of redundant path multistage interconnection networks. *IEEE Tr. on Computers*, C-32(12):1099–1108, December 1983.

[14] Tom Leighton, Derek Lisinski, and Bruce Maggs. Empirical evaluation of randomly-wired multistage networks. In *International Conference on Computer Design: VLSI in Computers and Processors.* IEEE, 1990.

[15] Tom Leighton and Bruce Maggs. Expanders might be practical: fast algorithms for routing around faults on multibutterflies. In *IEEE 30th Annual Symposium on Foundations of Computer Science*, 1989.

[16] Charles E. Leiserson. Fat-trees: Universal networks for hardware efficient supercomputing. *IEEE transactions on Computers*, C-34(10):892–901, October 1985.

[17] Henry Minsky, André DeHon, and Thomas F. Knight Jr. RN1: Low-latency, dilated, crossbar router. In *Hot Chips Symposium III*, 1991.

[18] E. Upfal. An $O(\log N)$ deterministic packet routing scheme. In *21st Annual ACM Symposium on Theory of Computing*, pages 241–250. ACM, May 1989.