Software-Based Fault-Tolerant Routing Algorithm in Multi-Dimensional Networks*

F. Safaei[1,3], M. Rezazad [1], A. Khonsari[2,1], M. Fathy[3], M. Ould-Khaoua[4], N. Alzeidi[4]

[1] IPM School of Computer Science, Tehran, Iran.
[2] Dept. of Electrical and Computer Engineering, University of Tehran, Tehran, Iran.
[3] Dept. of Computer Engineering, Iran University of Science and Technology, Tehran, Iran.
[4] Dept. of Computing Science, University of Glasgow, UK.

{safaei, rezazad, ak}@ipm.ir, {f_safaei, mahfathy}@iust.ac.ir, {mohamed, zeidi}@dcs.gla.ac.uk

Abstract

Massively parallel computing systems are being built with hundreds or thousands of components such as nodes, links, memories, and connectors. The failure of a component in such systems will not only reduce the computational power but also alter the network's topology. The Software-Based fault-tolerant routing algorithm is a popular routing to achieve fault-tolerance capability in networks. This algorithm is initially proposed only for two dimensional networks [1]. Since, higher dimensional networks have been widely employed in many contemporary massively parallel systems; this paper proposes an approach to extend this routing scheme to these indispensable higher dimensional networks. Deadlock and livelock freedom and the performance of presented algorithm, have been investigated for networks with different dimensionality and various fault regions. Furthermore, performance results have been presented through simulation experiments.

1. Introduction

Massively parallel systems, such as the Earth Simulator [2], the ASCI Red [3], and the BlueGene/L [4], are generally considered to be the most feasible way of achieving the enormous computational power. There exist many compute-intensive applications in science, engineering, and a number of other fields [5] that require continued research and technology development to deliver computers with steadily increasing computing power. Such systems are often composed of hundreds or thousands of components (i.e., routers, channels and connectors). Each individual component can fail, and thus, the probability of failure of the entire system increases dramatically. Therefore, in these systems, it is critical to keep the system running even in the presence of failures.

Fault-tolerance is one of the dominant issues facing the design of networks for massively parallel systems architectures. One of the most important aspects about a network is whether it can perform all functions in the presence of component failures or not. Failures in the interconnection network may isolate a large fraction of the machine containing many healthy processors that could otherwise be used. Although network components, like nodes and links, are robust, they are working close to their technological limits, and therefore, they are prone to failures. Hence, fault-tolerant routing algorithms for interconnect networks are becoming a critical design issue for large massively parallel systems. We say that a routing algorithm is fault-tolerant if it is able to persist in delivering messages between all non-faulty nodes in the presence of failures. Many fault-tolerant routing algorithms have been proposed for networks [1, 6-11]. The Software-Based fault-tolerant routing algorithm [1] is a one of popular routings that widely used in literature for achieving fault-tolerance capability in the networks. In this paper, we extend the algorithm proposed in [1] to higher dimensional networks.

The rest of the paper is organized as follows. Section 2 and Section 3 review some preliminaries in fault-tolerant interconnect networks and describe the fault modes and fault patterns, respectively. Section 4 presents the Software-Based approach for wormhole-switched networks. Performance evaluation based on the results of simulations is presented in Section 5. Finally, conclusions and directions for future research are given in Section 6.

* This research was in part supported by a grant from I.P.M. (No. CS1384-3-01).

1-4244-0054-6/06/$20.00 ©2006 IEEE

2. Preliminaries

This section provides brief descriptions of *n*-dimensional, radix-*k* torus (or *k*-ary *n*-cube) topology and its node structure. A *k*-ary *n*-cube is a class of regular graphs, consist of $N=k^n$ nodes arranged in an *n*-dimensional cube with *k* nodes along each dimension. Being a direct network, each of these *N* nodes serves simultaneously as an input terminal, output terminal, and the switching node of the network. Each node is assigned an *n*-digit radix-*k* address $\{a_{n-1}, \ldots, a_0\}$ and is connected by a pair of channels (one in each direction) to all nodes with addresses that differ by $\pm 1 (\text{mod } k)$ in exactly one address digit. Torus is regular (all nodes have the same degree) and is also edge-symmetric, which helps to improve load balance across the channel. A node is connected to its neighbouring nodes via 2-input and 2-output channels. The Processing Element (PE) to inject/eject messages to/from network uses the remaining channels, respectively. Messages generated by the PE are transferred to the router through the injection channel. Messages at the destination are transferred to the local PE through the ejection channel. Each physical channel is associated with some, say *V, virtual channels*. A virtual channel has its own flit queue, but shares the bandwidth of the physical channel with other virtual channels in a time-multiplexed fashion [12]. The router contains flit buffers for any incoming virtual channel. A *3V-way* crossbar switch, direct message flits from any input virtual channel to any output virtual channel. Such a switch can simultaneously connect multiple input to multiple output virtual channels while there is no conflicts.

Large messages are partitioned into fixed size message packets. The switching method determines the way messages visit intermediate routers. The network is routed using *wormhole switching* (also called as wormhole routing). Wormhole switching realizes very superior performance; but is prone to deadlock situations in the presence of faults [13]. In wormhole switching, messages are broken up into small units referred to as *flow control digits* or *flits* [14]. Data flits immediately follow the *header* flit as a pipelined fashion. A simple idea to route messages is to use a *deterministic* approach [14]. This routing algorithm routes packets by crossing dimensions in increasing order, nullifying the offset in one dimension before routing in the next one. Alternatively, *adaptive* routing algorithms which consider network state while making a decision can be utilized. One of the famous adaptive routing algorithm was proposed by [15] allows efficient router implementation due to its low requirement for virtual channels. Adaptive routing overcomes the performance limitations of deterministic routing by enabling messages to explore all potential paths between a pair of nodes. This routing flexibility, however, often requires dedicated hardware resources to ensure deadlock-freedom. Although the wormhole switching can achieve short message latency, its fault-tolerant capability is very low. In order to implement fault-tolerant communication in parallel computers, fault-tolerant routing algorithm is necessary. Software-Based routing that has been introduced as an efficient routing algorithm for reliable interprocessor communication can route a message from source to destination, even in the presence of faulty components. It is noteworthy mention that, in a fault-free network, the behaviour of deterministic and adaptive Software-Based routing is identical to dimension-order (e-cube) routing [13] and Duato's Protocol (DP) [15] fully adaptive routing, respectively. In Section 4, we will describe the Software-Based approach to fault-tolerant routing in wormhole-switched *n*-dimensional networks.

3. Fault models and fault patterns

In order to have simple idea of different fault patterns, it is firstly necessary to understand the nature of component failure modes. A failure mode is defined as occurrence of any error in the system that arises from a physical phenomenon [16]. Some failures such as Gaussian noise or alpha-particle strike result in *transient* faults, i.e., do not permanently defect machine operations. Transient faults are usually described by BER (Bit-Error Rate) or SER (Soft-Error Rate) modes [16] and usually can be handled by communication protocols, using CRCs. The other failures (such as connector failure or electro-migration of a conductor on a chip) cause a *permanent* fault. Permanent faults may be either *dynamic* or *static* [14, 16]. Dynamic faults occur randomly and consequently intermittently in time while the system is operating. In static fault, a faulty component stops functioning permanently and requires to be repaired. Sever static faults may cause system failures and consequently shutting down the system.

There are two different types of component failures: either the entire PE and its associated router can fail or just a physical link may fail. The former is referred to as a *node failure*, and the latter as a *link failure* [14]. On a node failure occasion, all physical links and virtual channels incident on the failed node are also marked faulty at adjacent routers. Adjacent faulty nodes may be coalesced into *fault regions*. Fault regions extended by faulty components, may form *convex* (also known as block faults) or *concave* shape [1, 14]. I-shaped, II-shaped or □-shaped and U-shaped,

+-shaped, T-shaped, H-shaped are examples of convex and concave regions, respectively. Some examples of fault regions are illustrated in Fig. 1.

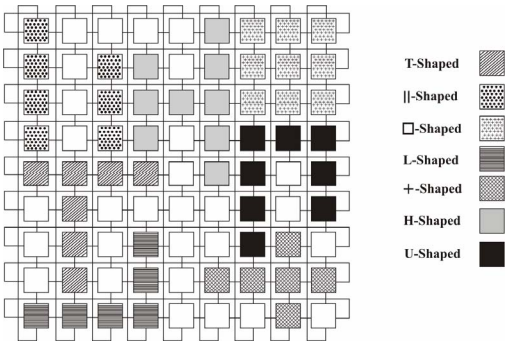

Fig. 1: Examples of coalesced fault regions in a 2-D torus network.

4. Software-Based fault-tolerant routing in *n*-D torus networks

In this section we briefly describe the Software-Based approach to fault-tolerant routing in wormhole-switched networks. This proposed technique is applicable to both deterministic and adaptive routing and specially targeted toward commercial multiprocessors where the mean time to repair (MTTR) is much smaller than the mean time between failures (MTBF). The main idea of 2-D Software-Based routing is as follows. When a message encounters faulty component, it is absorbed (ejected from the network) and delivered to the message passing layer of the local node. In order to keep track of a manner in which a message is re-routed, the header of message should be modified to reflect a different, non-minimal path around the fault region. It is evident that care must be taken to prevent a message from livelock and deadlock due to the occurrence of some combination of fault regions. When re-routed messages are absorbed at nodes due to faults, the algorithm uses three tables in making the re-routing decision for the message. Absorbed messages have priority over new messages to prevent *starvation*. In summary, these tables attempt to capture the following ideas. When a message encounters a fault, it is first re-routed in the same dimension in the opposite direction. If another fault is encountered, the message is routed in an orthogonal dimension in an attempt to route around the faulty regions. To the best of our knowledge, the Software-Based algorithm is only designed for two dimensional networks. Recent systems, however, mainly employed higher order dimensional interconnection topologies. Hence, it is essential to develop a suitable routing algorithm for higher dimensional networks.

Fig. 2 shows deadlock free fault-tolerant Software-Based routing in an *n*-dimensional torus network (which is depicted by SW-Based-nD in this figure). The algorithm uses two subroutines, namely detRouting2D and adapRouting2D. The detRouting2D subroutine deterministically routes messages in the network. In the absence of faults, detRouting2D subroutine is equivalent to e-cube [13] (also known as dimension order routing) algorithm. Similarly, adapRouting2D subroutine is implemented to adaptively route messages toward their destinations using modified DP fully adaptive routing [15].

```
Subroutine detRouting2D (Xcur, Ycur, Xdest, Ydest)
Begin
   while (Xcur,Ycur) != (Xdest, Ydest) do
      if no fault is encountered then /* for normal cases*/
         update Xcur & Ycur using Dimension_Order_Routing
      otherwise   /* for faulty cases */
         update Xdest, Ydest by applying SW-Based-2D.
         /*Software-Based routing algorithm for 2-D torus
         networks*/
      fi
   od
End
```

```
Subroutine adapRouting2D (Xcur, Ycur, Xdest, Ydest)
Begin
   while (Xcur,Ycur) != (Xdest, Ydest) do
      if no fault is encountered then /* for normal cases*/
         update Xcur & Ycur using Fully_Adaptive_Routing
      otherwise   /* for faulty cases */
         routing_type:= Deterministic;
         detRouting2D (Xcur, Ycur, Xdest, Ydest);
      fi
   od
End
```

```
Algorithm:
Software-Based Fault-Tolerant Routing in n-D Torus Networks
Var
   routing_type: {Deterministic, Adaptive};
Procedure SW-Based-nD
/* called by a node to route the messages initiated at the source
node towards the destination node*/
Input
   (cur1, cur2, …, curn): source node;
   (dest1, dest2, …, destn): destination node;
   n: network dimension;
Begin
   for i=1,2,3,…,n-1 do
      if routing_type = Deterministic then
         detRouting2D (curi, curi+1, desti, desti+1);
      otherwise
         adapRouting2D (curi, curi+1, desti, desti+1);
      fi
   od
End
```

Fig. 2: Extension of Software-Based routing scheme in *n*-D torus networks.

Note that in this case once a message finds the outgoing channel at a node leads to a fault, the message is absorbed at a local node and then detRouting2D subroutine will be called. From this point, faulted messages are always routed using detRouting2D subroutine and effectively follow the deterministic paths.

Deadlock freedom

A fault-tolerant routing algorithm should guarantee the delivery of messages in the presence of faulty nodes/links. When a message is blocked by a fault, message follows an alternative path to reach its destination. Deadlock freedom is an essential attribute of any routing algorithm and a challenging part is therefore devoted to prove that a proposed algorithm is deadlock free. The proof is based on demonstrating that the channel dependency graph of the routing algorithm is acyclic [14-16]. It has been proved in [1] the two dimensional Software-Based routing (i.e., SW-Based-2D in Fig. 2) is deadlock free. In our proposed n-dimensional routing algorithm, a message traverse two consecutive dimensions at each passing step (i.e., dimension i and i+1). Since, messages do not use more two dimensions at each step; following the same reasoning based on [1], the dependency graph is acyclic in intermediate steps. Moreover, the acyclicity of the channel dependency graph for the path through all dimensions is demonstrated by noting the fact that message traverses dimension in a monotonically increasing or decreasing order. Consequently, the routing algorithm SW-Based-nD is deadlock free.

Livelock freedom

Unlike deadlock, livelock messages continue to move through the network, but never reach their destination. This is primarily a concern for non-minimal routing algorithms that can misroute messages. If there is no guarantee on the maximum number of times a message may be misrouted, the message may remain in the network indefinitely [16]. In [1] it has been proved that depending on the location and shape of the fault patterns, livelock freedom property can be guaranteed for large number of faults. Livelock freedom property of SW-Based-nD routing is very similar to the one for SW-Based-2D routing because messages are routed in two dimensions at each passing step.

5. Performance evaluation

This section starts with introducing the assumptions made in this study and then describes the simulation experiments and their results.

5.1 Assumptions

The following assumptions have been made for the networks examined.

(a) Nodes generate traffic independently of each other, and which follows a Poisson process with a mean rate of λ messages/node/cycle.
(b) The arrival process at a channel is approximated by an independent Poisson process.
(c) Message length is fixed.
(d) Messages are transferred to the local PE as soon as they arrive at their destinations through the ejection channel.
(e) V virtual channels ($V \geq 1$) are used per physical channel. At a given routing step, a message chooses randomly one of the available virtual channels at one of the physical channels, if available, that brings it closer to its destination.
(f) When a header flit reaches a router, a routing decision selects the next output channel. The router's decision time is assumed to T_d.
(g) A flit is transmitted from one node in one cycle unit time if the corresponding buffer in the next node is empty.
(h) Faults distributed randomly or may be coalesced in fault regions. Furthermore, faults do not disconnect the network.
(i) When a message encounters a faulty component upon reaching an intermediate node, it is removed from the network by the local router and delivered to the messaging layer of the local node's operating system. The message passing software either i) modifies the header so the message may follow an alternative path or ii) computes an intermediate node address [1]. It is assumed that the message encounters delay overhead of Δ cycles due to it's re-injected at the intermediate node.

5.2 Simulation experiments

Simulation experiments have been realized using a discrete-event simulator that mimics the behaviour of Software-Based routing algorithm with faulty components at the flit level. It accept several parameters including network size, message length, number of virtual channels, buffer length, message generation rate, number of faulty components, router decision time, delay overhead for re-routing and many other parameters. Extensive simulations for several networks, message length and buffer sizes have been carried out for two different routings: namely deterministic and adaptive. The results were very interesting and we mention here some results as a

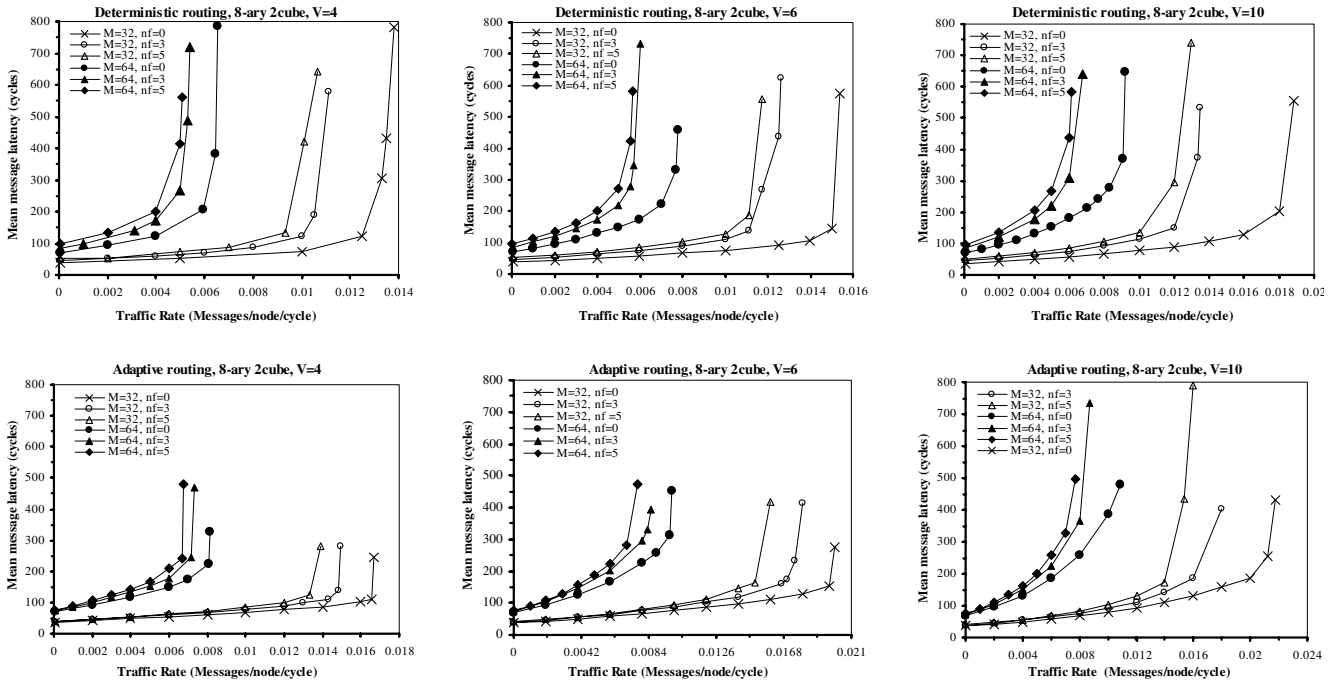

Fig. 3: The mean message latency provided by the simulation against the traffic rate using deterministic and adaptive routing in an 8-ary 2-cube with message length *M*=32 and 64 flits, different number of virtual channels per physical channel *V*=4, 6, 10 and different number of failed nodes *nf*=0, 3, 5.

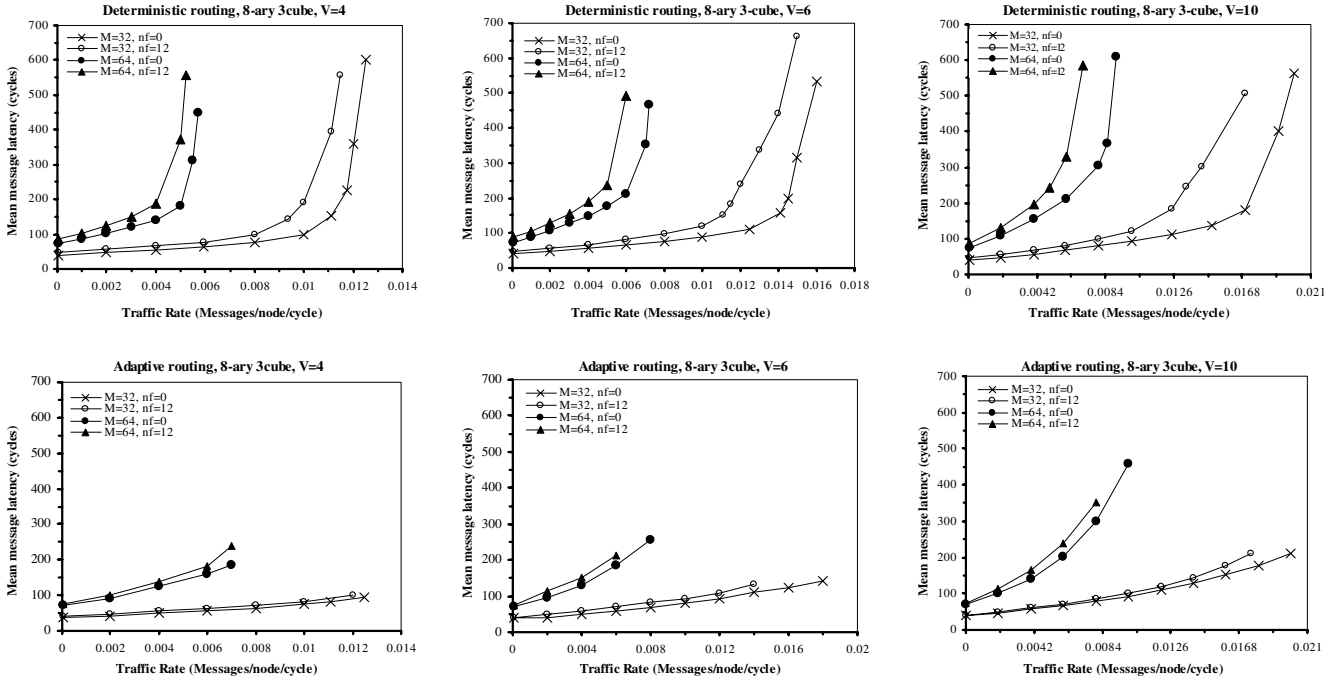

Fig. 4: The mean message latency provided by the simulation against the traffic rate in an 8-ary 3-cube using deterministic and adaptive routing with message length *M*=32 and 64 flits, different number of virtual channels per physical channel *V*=4, 6, 10 and different number of failed nodes *nf*=0, 12.

summary of results for concluding. In all experiments mentioned here we assumed the router decision time, T_d, and the overhead delay of local queue for re-injection are equal to zero. It is due to the fact that, the decision time and overhead delay compared to the channel cycle time are usually negligible. For each generation rate a total of 100,000 messages are derived. Statistics gathering was inhibited for the first 10,000 messages to avoid distortions due to the startup transient. The simulator uses the assumptions were mentioned in Section 5.1, and some of these assumptions are detailed here with a view of making the network operation clearer. The network cycle time is defined as the transmission time of a single flit from one router to the next. Messages are generated at each node according to a Poisson process with a mean inter-arrival rate of λ messages/node/cycle. Message length is fixed in term of flits. Random faulty nodes are determined using a uniform random number generator. The mean message latency is defined as the mean amount of time from the generation of a message until the last data flit reaches the local PE at the destination node. Our approach provides fault-tolerance in n-dimensional torus networks. Hence, numerous experiments have been performed for different sizes of the network and message length. However, for the sake of the clarity, the experimental results will mainly be based on a two and three dimensional torus networks. Our method can tolerates both node and link failures. A link failure can be modelled by the failure of two nodes connected to it. Therefore, we focus only on node failures. In what follows, we investigate the mean message latency in networks in the presence of faults.

Results for deterministic and adaptive routing with random failed nodes

In this section, we investigate the mean message latency for the number of random failed nodes with deterministic and adaptive routing. Figs. 3 and 4 illustrate the mean latency curves provided by the simulator for 8×8 and 8×8×8 torus networks when deterministic and adaptive routing are used, respectively; with different message lengths, M=32, 64 flits, V=4, 6, 10 virtual channels per physical channel and different number of random failed nodes nf=0, 3, 5 and 12. The horizontal axis in the figures shows the traffic generation rate at each node (λ) while the vertical axis shows the mean message latency. The figures reveal that as the number of faulty nodes increases, the mean message latency also increases and the network saturates at lower traffic rates. This is because the percentage of messages that encounter the fault and re-injected to the network is increases. Moreover, it is evident from the figures that, the latency for longer messages (i.e., 64-flit) is higher than shorter messages (i.e., 32-flit). The reason behind this observation is that the message latency is proportional to message length.

Results for deterministic and adaptive routing with different fault regions

In this section, we capture the mean message latency for various fault regions using deterministic and adaptive routing algorithm. Fig. 5 depicts the mean message latencies of deterministic and adaptive routing for some of convex and concave fault regions. As is evident from this figure, due to the greater difficulty in entering and exiting a concave fault region, the mean message latency is greater in the presence of concave than for convex fault regions. Also, the message latency for adaptive routing is substantially lower than the message latency for deterministic routing.

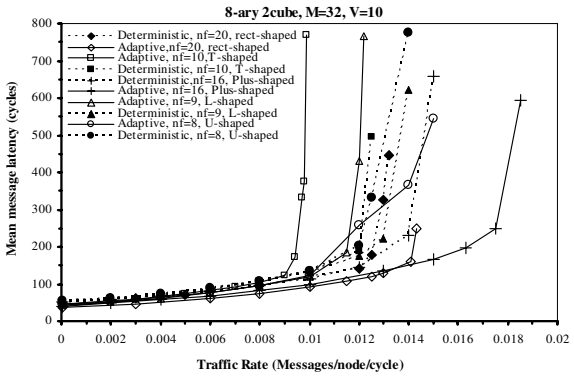

Fig. 5: The mean message latency provided by the simulation as a function of traffic generation rate in an 8-ary 2-cube using deterministic and adaptive routing with message length M=32 flits, V=10 virtual channels per physical channel and various fault regions.

In order to make the results independent of relative positions of failures, we have run simulations (for each number of failures); each of them corresponding to a different randomly selected failures. Fig. 6 shows the mean overall network throughput achieved for different number of node failures in a 16×16 torus, with message length M=32 flits and V=6 virtual channels per physical channel using deterministic and adaptive routing. Throughput is the rate at which messages are delivered by the network for a particular traffic pattern [16]. It is measured by counting the messages that arrive at destination over a time interval for each flow in the traffic pattern and computing from these flow rates the fraction of the traffic pattern delivered. As is evident from the Fig. 6, the network performance is not seriously affected by the presence of failures in both

cases. Moreover, this figure reveals that the throughput for deterministic routing is lower than adaptive routing. This is due to the fact that in deterministic routing when a message visits a faulty node or link on its way to destination is then delivered to the current node, which in turn encounters the software overhead upon re-injection to the network from its location. Adaptive routing, however, is not restricted to choose always one path to reach to destination and has the flexibility of selecting any profitable paths to destination. Thus, adaptive routing provides alternative paths to destination and a message is delivered to current node when all available paths are faulty. Therefore, this type of routing most of the time avoids the delivering of messages to the intermediate nodes and in turn is not suffering the big software overhead.

The plots in Fig. 7 show the number of messages queued in adaptive and deterministic cases by message absorbing nodes for message injection rates of 70 and 100 in the presence of random failed nodes ranging from 1 to 12 nodes. The number of messages queued is the number of messages absorbed due to faults. It is clear that, a given message contributes more than once to this count if it is absorbed multiple times. As the number of node faults increases, the number of messages is queued increases.

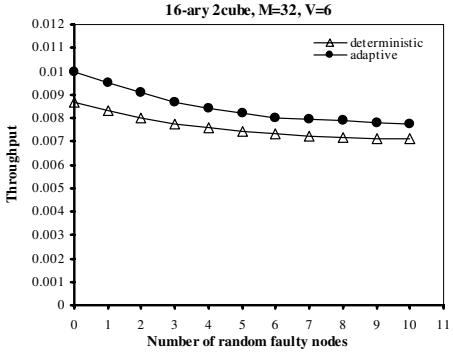

Fig. 6: Comparison between the throughput of adaptive and deterministic Software-Based routing against the number of failed nodes in a 16-ary 2-cube with message length *M*=32 flits and *V*=6 virtual channels per physical channel.

This figure illustrates that the number of messages queued is much higher for deterministic routing than for adaptive case. Indeed, at the lower and higher injection rates, the number of messages absorbed remaining relatively constant for adaptive routing while approximately doubling for deterministic routing. This is due to the fact that, adaptive routing can provides different paths to deliver the messages to their destinations and a message only is delivered to the local queue of current node when all existing paths still to be visited are faulty. As a result, adaptive routing can avoids the delivering of messages to the local queue of the intermediate nodes.

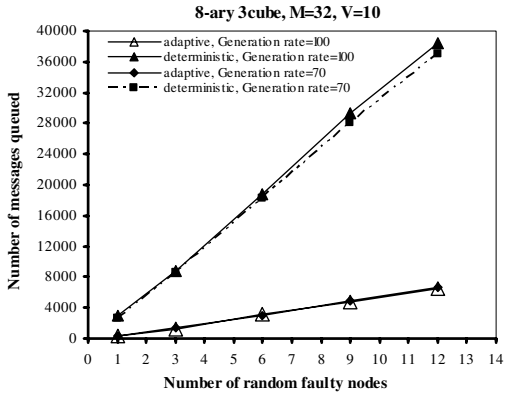

Fig. 7: The number of messages in the local queue (messages queued) as a function of random node failures using adaptive and deterministic Software-Based routing for different traffic generation rates 70 and 100 in an 8-ary 3-cube with message length *M*=32 flits, and *V*=10 virtual channels per physical channel.

6. Conclusions

Fault tolerance and network routing have been among the most studied topics in the research of parallel processing and computer networking. A fault-tolerant routing algorithm should guarantee the delivery of messages in the presence of faulty components (i.e., nodes or links). In this paper, we have presented an approach to extend the 2-dimensional Software-Based fault-tolerant routing algorithm in multi-dimensional networks that enables us to study the behavior of these networks in terms of individual component failure. There are several advantages of such this approach. First, Software-Based routing can tolerate more traffic load than other routing algorithms in *n*-dimensional networks. This is because more messages in other routing algorithms transit the network at a given time; while faulty messages in Software-Based routing are always removed from the network releasing resources to be used by non-faulty messages. Second, routers designs will minimally impacted, and thus remain compact and fast. Only messages that encounter faulty components are affected, while the machine is ensured of continued operation until the faulty components can be replaced. Our next object is to develop an analytical modeling approach to investigate the performance behavior of Software-Based fault-tolerant routing algorithm.

References

[1] Y.J. Suh, et al., Software-based rerouting for fault-tolerant pipelined communication, *IEEE Trans. On Parallel and Distributed Systems*, Vol. 11, No. 3, March 2000.

[2] Earth Simulator Center: http://www.es.jamstec.go.jp/esc/eng/index.html.

[3] ASCI Red Web Site: http://www.sandia.gov/ASCI/Red/.

[4] IBM BG/L Team, An overview of BlueGene/L Supercomputer, *ACM Supercomputing Conf.*, 2002.

[5] P. W. Dowd, M. Carrato, High speed routing in a parallel processing environment: a simulation study, *Proc. 24th Annual Simulation Symposium. New Orleans, Louisiana, IEEE Computer Society Press*, pp. 60-72, 1991.

[6] M. E. Gómez, et al., A New Adaptive Fault-Tolerant Routing Methodology for Direct Networks, pp. 462-473, *HiPC* 2004.

[7] J. M. Montañana, et al., A Transition-Based Fault-Tolerant Routing Methodology for InfiniBand Networks, *IPDPS* 2004.

[8] M. Gómez, et al., An Effective Fault-Tolerant Routing Methodology for Direct Networks, pp.222-231, *ICPP* 2004.

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

[10] D. Karimou, J. Myoupo, A Fault-Tolerant Permutation Routing Algorithm in Mobile Ad-Hoc Networks, *Lecture Notes on Computer Science*, Vol. 3421, pp.107-115, 2005.

[11] G. Gupta, M. Younis, Fault-tolerant clustering of wireless sensor networks, *IEEE Conf. on Wireless Communications and Networking*, pp. 1579-1584, March 2003.

[12] W.J. Dally, Virtual channel flow control, *IEEE Trans. Parall. Distrib. Syst.* Vol. 3, No. 2, pp. 194–205, 1992.

[13] W. J. Dally and C. L. Seitz, Deadlock-free message routing in multiprocessor interconnection networks. IEEE Transactions on Computers, Vol. 36, No. 5, pp. 547-553, May 1987.

[14] J. Duato, S. Yalamanchili, L.M. Ni, *Interconnection networks: An engineering approach*, Morgan Kaufmann Publishers, 2003.

[15] J. Duato, A new theory of deadlock-free adaptive routing in wormhole networks, *IEEE Transactions Parallel Distributed Systems*, Vol. 4, No. 12, pp. 1320-1331, 1993.

[16] W.J. Dally and B. Towles, *Principles and practices of interconnection networks*, Morgan Kaufman Publishers, 2004.