Evaluating the Performance of Software-Based Routing Algorithms for Dynamic Fault-Tolerance in Tori

M. ValadBeigi  
Faculty of Electrical and Computer Engineering  
Shahid Beheshti University G.C.  
Evin 1983963113, Tehran, IRAN  
m.valadbeigi@mail.sbu.ac.ir

F. Safaei  
Faculty of Electrical and Computer Engineering  
Shahid Beheshti University G.C.  
Evin 1983963113, Tehran, IRAN  
f_safaei@sbu.ac.ir

A. Mortazavi  
Department of Electrical and Computer Engineering  
University of Tehran  
Tehran, IRAN  
a.mortazavi@ece.ut.ac.ir

Abstract—Fault-tolerance has been the focus of attention recently 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. In this paper, we present a comparative performance study of seven prominent fault-tolerant routings in 2-D wormhole-switched tori. The routing algorithms here are coupled with the Software-Based routing scheme which supports dynamic fault-model, enabling the network to remain fully operational at all times. The performance measures studied here are the throughput and average message latency. Results obtained through simulation suggest two classes of presented routing schemes as high performance candidates in most faulty networks.

Keywords—Fault-tolerance; Interconnection Networks; Software-Based Routing Methodology; Torus Topology; Dynamic Fault

I. INTRODUCTION

Fault-tolerance is defined as the ability of a system to continue an operation despite the presence of faults [1]. Three most applicable terms in fault-tolerance are reliability, availability and dependability [2]. However, due to the large application area, interconnection networks are found in systems with high requirements for reliability and continued operation. In this paper, we have studied reliability in interconnection networks within torus topology which is among the most commonly used interconnection networks [1].

The use of fault-tolerance mechanisms will assure, in case of a component failure, that the system keeps working, although in a degraded mode, until the failed component is repaired. Basically, there are three ways to cope with faults in the interconnection networks: component redundancy, fault-tolerant routing algorithms, and reconfiguration techniques. Using component redundancy has been the easiest though costly way to provide fault-tolerance. Components in the system are replicated and once a failed component is detected, it is simply replaced by its redundant copy. An example of using component redundancy can be found in the Tandem’s Himalaya Servers [3]. Fault-tolerant routing algorithms aim at avoiding messages traverse faulty components by providing some kind of routing path redundancy. To reach this end, messages must be able to be routed through alternative paths to circumvent or avoid faulty regions over the network. Fault-tolerant routing schemes should be designed to tolerate a certain number of faults, while still guaranteeing deadlock freedom in the network. However, to fulfill the requirements, fault-tolerant routing strategies often need to use additional network resources such as virtual channels or additional hardware at switches or routers. Applying reconfiguration [4] on any number of faults can be tolerated provided the network is physically connected. The reconfiguration consists, once a fault is detected, of discovering the new topology, computing a new routing scheme and updating the required components in the network. The main disadvantage of reconfiguration is the high delay packets that may occur during the reconfiguration process. Reconfiguration techniques often require additional network resources [4].

Each mechanism of fault-tolerant approach has different requirements on resources, and therefore different costs. Those fault-tolerant approaches with hardware requirements are usually costly, more expensive than software-based mechanisms. When the failure probability is high or the system has strong fault-tolerant requirements, like life support systems, it is normally preferable to use hardware-based solutions that are generally more expensive. On the contrary, for the less critical applications or low faults probability, it is preferable to use cheap mechanisms based on software.

The Software-Based [5] routing scheme is a popular fault-tolerant mechanism widely used in the literature for supporting high adaptivity and interprocessor communications in parallel computers due to its ability to preserve both communication performance and fault-tolerant demands in such networks.

Faults in an interconnection network can be dealt with *statically* or *dynamically* [6]. When a static fault-model is used, all the faults need to be known when the system is started. Thus, when a fault occurs, the system has to be restarted. When a dynamic fault-model is used, the system remains operational while measures are taken to circumvent the faulty component(s).
In this paper, the performance comparison of seven routing algorithms is investigated in 2-D wormhole-switched tori coupled with Software-Based routing scheme which is able to support dynamic fault-model. The employed routing algorithms are the Positive-Hop (PHop), Negative-Hop (NHop), Duato’s routing, Minimal-adaptive, and Fully-adaptive. Besides, Nbc and Pbc are resulted from some modifications on the first two algorithms [7].

The coming sections are as follows: Section II presents related work on fault-tolerant routings. Section III introduces terms and backgrounds to understand the paper. Section IV explains briefly the basic adaptive routing algorithms coupled with the Software-Based scheme. The extension to the dynamic fault-model and dynamic transition phase are described in Section V. Section VI gives simulation results on the performance of the seven routing algorithms in the presence and absence of faults. Finally, Section VII summarizes the work reported in this study.

II. RELATED WORK

A large number of fault-tolerant routing algorithms for multiprocessor systems have previously been proposed, especially for systems with mesh and torus topologies. Most of the routing techniques have been proposed for wormhole switching. In this section we provide a brief survey of fault-tolerant routing algorithms that have been proposed according to torus topology. First, we focus on techniques which require hardware support. Later, we describe techniques based on software support.

In [8], Chalasani and Boppana proposed an efficient method to design fault-tolerant routing algorithms to handle a convex (or block) fault-model suited for torus networks. Besides addressing convex faults, Chalasani and Boppana introduced the concept of fault-rings to assist fault-tolerant routing. This method requires a total of six virtual channels to tolerate non-overlapping block faults. It was further improved by the same author, proposing a method tolerating overlapping block faults in torus using three virtual channels [9] and another method [10] tolerating overlapping solid faults using four virtual channels. In [11] a fault-tolerant routing technique based on routing adaptively for some source-destination pairs to an intermediate node, and also routing adaptively from an intermediate node to the destination is proposed. This technique requires three virtual channels for meshes and torus while tolerates up to five faults. An additional virtual channel (total of four) is required for obtaining deadlock free minimal routing (in meshes and torus). In [11] a low cost fault-tolerant routing algorithm for virtual cut through switching [1] valid for any topology was proposed. Gaughan and Yalamanchili [12] proposed pipelined circuit switching (PCS) in which the header sets up a path of virtual channels before the data is sent. In the simulations PCS performed poorly when it used with only one virtual channel. Scouting is a variation of this scheme, proposed by Duato et al. [13]. Moreover a further improvement was proposed by Dao et al. [14]. In the latter proposal, a configurable flow control is used to enable a wormhole like flow control to be used in the fault-free case while scouting is used when faults are present. Kim et al. [15] proposed compressionless routing for wormhole routed torus, providing adaptive routing without the use of virtual channels.

The addition of virtual channels and the enforcement of routing restrictions between hardware and software based fault-tolerant routing can impact the design and implementation of the switches. However, in the environments where the fault rates are relatively low, the use of expensive, custom, fault-tolerant switches often cannot be justified. Moreover, contemporary routers are compact, oblivious and fast. To overcome the drawbacks of the hardware-based solutions, a Software-Based fault-tolerant routing scheme [5] can be used. In [7] the performance vicissitudes of nine routing algorithms is investigated in 2-D wormhole-switched tori coupled with Software-Based routing scheme. In this paper, we have tried to extend the above mentioned idea to a dynamic fault-model.

III. PRELIMINARIES

In this section, we address the network model and the necessary background information that is used in the paper.

A. The Torus and Its Node Structure

A \((k, n)\)-torus (also known as \(k\)-ary \(n\)-cube) has \(n\) dimensions, denoted \(DIM_0, DIM_1, ..., DIM_{n-1}\), and \(N = k^n\) nodes. Each node is uniquely indexed by an \(n\)-tuple in radix \(k\). Each node is connected via communication links to two other nodes in each dimension. The neighbors of the node \(x = (x_0, x_1, ..., x_n)\) in \(DIM_i\) (\(0 \leq i \leq n\)) are \(x = (x_0, x_1, x_2 \pm 1, x_3, ..., x_n)\) where addition and subtraction are modulo \(k\). A link is said to be wraparound link if it connects two neighbors whose addresses differ by \(k - 1\) in \(DIM_i\). In this paper, to simplify presentation, we consider \((k, 2)\)-torus networks with bi-directional links implemented using two unidirectional physical communication channels. We denote the link between nodes \(x\) and \(y\) by \(\prec x, y \succ\). We assume that a crossbar is used in each router to connect its input channels to its output channels.

B. Software-Based Fault-Tolerant Routing Scheme

Software-Based fault-tolerant routing scheme [5], referred to as \(e\)-sft, provides a solution for tolerating failures with concave fault regions, which are not tolerated by some
functions may be prone to deadlocks if they coexist in the two different and individually deadlock-free routing algorithms as possible to route messages. When a message is blocked by faults at node and there is no fault-free link such that the link from to is along the shortest path, the additional routing scheme is needed to route the message around the fault regions. To do so, we couple the Software-Based routing scheme [5], as a well-known instance of a fault-tolerant technique widely used in the literature to improve adaptive routing algorithms for fault-tolerant routings.

Many researchers [6-15] are investigating suitable adaptive wormhole-switched algorithms for high performance and fault-tolerance in torus networks. Most of their results are on the design of adaptive wormhole-switched algorithms using as few virtual channels as possible. Incorporating adaptivity may improve the throughput and average message latency. However, there are no general results, which show the applicability of these algorithms to derive corresponding wormhole-switched fault-tolerant routing algorithms. To fill this gap, in this work, we characterize a variety of adaptive routings considering their fault-tolerant performance and design trade-offs. To further illustration of our evaluation, we consider six well-known basic routing algorithms: two hop-based routings (i.e., PHop, NHop), Duato’s methodology, Minimal-adaptive, Fully-adaptive, and two other sets of modified algorithms based on the basic routing algorithms [7]. These fully adaptive routing algorithms have been characterized in [7] of which the one using negative hop-based deadlock-free routing, augmented with a new idea called bonus cards, has shown to have the best performance. For a comprehensive survey of the important issues of these routings, the reader is referred to the articles in [17].

V. EXTENSION TO DYNAMIC FAULT MODEL

Using the method with a dynamic fault model leads to more complexities. Under dynamic fault model assumption, sending fault info and calculating routing function must be performed fully distributed and locally at each node. We assume that every link fault causes neighbors to be informed about the location of the fault. According to this scheme, when a node becomes faulty, all neighbors could be informed about the faulty node. Thus, related nodes can begin the distribution of the status information.
In Software-Based scheme, when a message encounters a faulty node, the message that is removed from the network is again re-injected [5]. According to the re-injection and message-passing software [5], each intermediate node is informed about their neighboring nodes whether they are faulty or not. So we don’t need to distribute the information of the faulty nodes.

As shown in the rest of this section, after any failure in the network; distribution of the status information will be done. Update messages cause changes in nodes and let them calculate new routing function. However, in dynamic transition phase between the old and the new routing function, ghost channel dependency [16] is a concern. The old routing function may become disconnected due to new faulty components in the network. Suppose packet $P$ arriving in class $A$ by the old routing function. If a node failure makes class $A$ disconnected when the distribution procedure is setting-up new classes for the faulty region, there is two ways to route packet $P$: the first way is to change packet class from $A$ to a neighbor class; this choice may offers channel dependency between two adjacent classes. The second way is to drop packet $P$ from the network. Dropping from network could be transient or for a period of time. Also there are some methods for deadlock-free dynamic reconfiguration. For example, the double schemes [18, 19] and Lysne et al. [20] provide two different approaches to achieve deadlock- free dynamic reconfiguration for arbitrary topologies and routing algorithms. We do not consider these methods, and just drop packets, because of the network costs. Another reason is that some packets will be dropped due to links disconnection. Considering that the packets being buffered at a failing node or being transmitted on a failing link are generally lost. Thus, lossless transition is almost impossible and costly.

In the Software-Based approach when a message encounters a faulty link, it is removed from the network or absorbed by the local router and delivered to the messaging layer of the local node’s operating system. According to the above mentioned issue, our old and new functions are the same in this case. Therefore, there is no ghost dependency in dynamic software based approach.

VI. PERFORMANCE EVALUATION

To study the performance issues, a flit-level, listener-based object oriented simulator [21] has been used to illustrate the performance of routing algorithms. This simulator can be used for wormhole switching in $(k, 2)$-torus networks with and without faults. The crossbar switch in the router allows multiple messages to traverse a node simultaneously. The virtual channels on a physical channel are demand time-multiplexed. Only the virtual channels with messages to transmit use the physical channel in a round-robin manner. Each virtual channel, upon receiving permission to use the physical channel, transmits the next flit of its message and yields the physical channel to the next virtual channel in the queue. Idle virtual channels do not consume any bandwidth. It takes one network cycle to transmit a flit between neighbors.

![Figure 1. Simulation results for dynamic fault mode which shows average message latency. (a) One faulty node is injected after 40,000 cycles; (b) Two faulty nodes are injected after 40,000 cycles; (c) Three faulty nodes are injected after 40,000 cycles.](image)

We have used 10 virtual channels per physical channel in this study. We have considered square torus in the simulation. The torus size considered for simulation is 8×8. One, two and three faults have been simulated in the experiments. In the literature, fixed-length messages with 32, 64, or 100 flits are commonly considered [5-7, 9-11, 15]. We have used 100-flit messages in this study. In the simulation, we have used a uniform traffic pattern; that is, a
processor sends a message to any other active nodes with equal probability. Messages are generated at time intervals chosen from an exponential distribution. The processors route messages in an asynchronous manner. Conflicts of requests for an output channel by multiple messages are resolved in a random manner.

Three most important performance measures are network throughput, message latency and the network power. We use throughput and average message latency as the performance metrics. The throughput is equal to the number of consumed messages over the number of messages that can be transmitted at the maximum load [1, 13, 14].

In order to evaluate the Software-Based scheme with dynamic fault-model, a processing delay of 70 cycles is added after any dynamic changes in the network. Each simulation has been run for 80,000 cycles. Figures 1(a), 1(b), and 1(c) show the average message latency for 90% of the maximum accepted load over time. In Figure 1(a), all nodes and links are healthy and after 40,000 cycles, one faulty node is injected into the network, while in Figure 1(b), two faulty nodes and in Figure 1(c), three faulty nodes are injected into the network.

Moreover, Figures 2(a), 2(b), and 2(c) show the throughput message latency for 90% of the maximum accepted load over time. In Figure 2(a), all nodes and links are healthy and after 40,000 cycles, one faulty node is injected into the network while in Figure 2(b), two faulty nodes and Figure 2(c), three faulty nodes are injected into the network.

As the number of faults increases, the latency increases steadily while the throughput drops steadily. After cycle 40,000, there is a significantly sharper increase in latency and decrease in throughput than what we seeing previously [7].

Comparing Figure 1(a) with Figure 1(b) and also Figure 1(b) with Figure 1(c), we note that Nbc has 35% and 22% increase in latency respectively. However, Nbc has a less decrease in throughput. The Nbc shows the graceful degradation of performance in the presence of faults. Moreover, In the presence of failure, PHop gives better throughput than NHop, Nbc and despite the use of less virtual channels. A possible explanation is that the load on virtual channels is balanced in Nbc but not in NHop and PHop algorithms.

Finally, Figure 3 illustrates the comparison of average message latency between seven routing algorithms by adding one, two, and three faulty nodes. It reveals that with absence of faults the latency is considerably low. With the injection of one fault the latency increases noticeably. The process is the same even when two faults are injected. But this increase is not much different when three faults injection happens. Eventually, Figure 3 points out that Nbc has a better performance with less message latency.
VII. CONCLUSIONS

This paper investigates the performance of several wormhole-switched adaptive routing algorithms incorporated with the fault-tolerant Software-Based routing scheme which is able to support dynamic fault-model. The methodology for supporting dynamic fault-model allows the network to remain fully operational in the case of failures without stopping network traffic at any time. We have considered seven adaptive algorithms divided into two classes: basic and modified. Basic algorithms include PHop, NHop, Minimal routing, Fully-adaptive routing, and Duato’s methodology, while Nbc and Pbc are the modified. The basic algorithms PHop and NHop were improved with bonus cards (i.e., Pbc and Nbc). The modified algorithms are a combination of the basic and modified cases (i.e., Pbc and Nbc) that show a quite better performance. In the congested cases of a network, the virtual channels are busy. The Software-Based routing scheme follows a specific rule in choosing virtual channels. When it is coupled with an algorithm, it uses virtual channels in a random way without any regularity causing the virtual channels of higher classes to be quickly occupied. As a result, the Software-Based scheme does not proceed in acquiring virtual channels results in a network rapidly congested and the message latency increases. Moreover, supporting dynamic fault model leads to considerable increase and decrease of average message latency and throughput. When the faults increase in the network, the level of increase in message latency and the decrease in throughput reduces.

REFERENCES