The CSI Journal on
Computer Science and Engineering
Vol. 3, No. 2&4, 2005
Pages 27- 38
**Regular Paper**

# Performance Analysis of Fully Adaptive Routing Algorithms in Wormhole-Switched Interconnect Networks*

Farshad Safaei[1, 3] Ahmad Khonsari[2, 1] Mahmood Fathy[3]
Mohamed Ould-Khaoua[4]

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

## Abstract

As the number of elements in large-scale massively parallel computers, Multiprocessors System-on-Chip (MP-SoCs), and peer-to-peer communication networks increases, the probability of component failure becomes significant. Consequently, fault-tolerance turns out to be a key issue in the design of such systems. Adaptive routing algorithms have been frequently suggested as a means of improving communication performance in such systems. These algorithms, unlike deterministic routing, can utilize network state information to exploit the presence of multiple paths. Before such schemes can be successfully incorporated in networks, it is necessary to have a clear understanding of the factors which affect their performance potential. This paper investigates the performance of nine prominent adaptive fault-tolerant routing algorithms in wormhole-switched 2-D tori with a routing scheme suggested by Chalasani and Boppana [1], as an instance of a fault-tolerant method. The suggested scheme is widely used in the literature to achieve high adaptivity and support inter-processor communications in massively parallel computers due to its ability to preserve both communication performance and fault-tolerant demands in these networks. The performance measures studied in the paper are the throughput, average message latency and average usage of virtual channels per node. Results obtained through simulation suggest two classes of presented routing schemes as high performance candidates in most faulty networks. Furthermore, we propose an analytical model to assess the performance behavior of a fully adaptive routing which has been shown to be one of the most efficient routings in the torus networks. The validity of the model is demonstrated by comparing analytical results with those obtained through simulation experiments.

**Keywords:** Massively parallel computers, Interconnect networks, Torus, Adaptive routing, Virtual channels, Message latency, Queuing theory, and Performance evaluation.

## 1. Introduction

Large-scale massively parallel computers, Multiprocessors System-on-Chip (MP-SoCs), and peer-to-peer communication networks provide high performance computing that allows users to deal with large and heavy computational tasks. Parallel machines have a dedicated, high bandwidth interconnect network that deliver messages with a very low latency. This low latency is achieved by using fast routing components, single-chip switches with short wire delays, wide links, and putting the network interface close to the processor, typically on the processor-memory bus. In these systems, tasks are executed by a set of intercommunicating nodes or processors. The communication is usually carried out by means of passing messages from one node to another over the interconnect network. The performance of the interprocessor communication depends largely on the network topology, the switching method, and the path selection technique. The torus topology has become a popular interconnect architecture for constructing massively parallel computers. Many parallel systems adopt low dimensional torus networks due to their low communication latency and high bandwidth [2, 3].

Most of the contemporary massively parallel computers use wormhole switching (also widely known as wormhole

routing [3, 4]) mechanism to support their interprocess communication. In the wormhole switching, a message is divided into a sequence of fixed-size units of data, called *flits*. If a communication channel transmits the first flit of a message, it must transmit all the remaining flits of the same message before transmitting flits of another message. Wormhole switching only requires small buffers in the routers through which messages are routed. Also, it makes message latency largely insensitive to the distance in the network. The main drawback of wormhole switching is that blocked messages remain in the network, therefore they waste channel bandwidth and block other messages. In order to reduce the impact of message blocking, physical channels may be split into *virtual channels* by providing a separate buffer for each virtual channel and by multiplexing physical channel bandwidth between virtual channels. The use of virtual channels can increase throughput considerably by dynamically sharing the physical bandwidth among several messages [1-7].

Message passing in massively parallel computers is implemented based on a routing algorithm that determines the path which a message follows to reach its destination. Routing algorithms for these systems are generally classified as being either *deterministic* or *adaptive* [2-4]. In deterministic routing, all messages between a given source/destination pair will follow the same path. One of the main benefits is that in-order arrival of messages is preserved. However, deterministic routing usually makes an inefficient use of the network resources [4]. An alternative, but complementary approach, to improve network performance consists of using adaptive routing. This routing strategy provides alternative paths to route messages, thus avoiding congested regions in the network and increasing throughput.

While high throughput and low latency are key features of interconnect networks, the issue of fault-tolerance is now becoming increasingly important. The huge number of processors and associated devices (memories, switches, and links, etc.) significantly affects the probability of failure. Each individual component can fail and, thus, the probability of failure of the entire system increases dramatically. Therefore, in such systems, it is critical to keep the system running, even in the presence of failures. In addition, failures in the interconnect networks may isolate a large fraction of the machine, containing many healthy processors that could have been used. Although network components, like switches and links, are robust, they are working close to their technological limits and, therefore, they are prone to failures. Increasing clock frequencies leads to a higher power dissipation, which again could lead to premature failures. Fault-tolerance is the ability of a routing algorithm to bypass faulty nodes/links in the network. In a large sized network, it is essential to design a fault-tolerant routing algorithm that can route messages in the presence of faulty components. Fault-tolerant routing for large-scale parallel computers has been the subject of extensive research in recent years [1, 7-18]. Chalasani and Boppana [1] have presented an efficient framework to design fault-tolerant routings to handle a convex (or block) fault model for the torus networks. Also addressing convex faults, the key concept used in their method is that, for each fault region, a fault ring consisting of fault-free nodes and physical rings can be formed around it. These fault rings can be used to route messages around fault regions.

Most network performance evaluation studies have been conducted by means of software simulation [4-13, 15, 16, 19]. To study the relative performance merits of routing algorithms, however, using simulation techniques is limited by the excessive computation times required to run large simulations. In contrast analytical modeling offers a cost-effective and versatile tool to carry out such study typically by requiring a far lower computational load. In this paper, the performance of nine routing algorithms is investigated in 2-D wormhole-switched torus based on a routing scheme suggested by Chalasani and Boppana [1], as an instance of a fault-tolerant routing. This method is widely used in the literature for supporting high adaptivity and inter-processor communications in parallel computers due to its ability to preserve both communication performance and fault-tolerant demands in such networks. Two of these algorithms are the basis of the other four algorithms. The routing algorithms used are the Positive-Hop (PHop), Negative-Hop (NHop), Duato's routing, Minimal routing, and Fully-Adaptive routing. The other four algorithms are resulted from some modifications that are being applied to the basic routing algorithms. Furthermore, we will derive a mathematical model to assess the performance behavior of a fully adaptive routing which has been developed in the paper and shown to be one of the most effective routings in the torus networks.

The remainder of the paper is organized as follows. Section 2 reviews some definitions and background of the torus and describes the concept of fault models used in this paper. This section also introduces briefly Chalasani-Boppana's routing scheme. Section 3 describes the basic adaptive routing algorithms, which are fortified with Chalasani-Boppana's scheme. Four sets of the modified routing algorithms are introduced in Section 4. Section 5 gives simulation results on the performance study of the routing algorithms in the presence and absence of faults. This section also investigates the traffic load distribution around the fault-rings. Section 6 gives an overview of the model assumptions used in the paper, and outlines the analytical modeling approach. Section 7 compares the delays predicted by the analytical model with those obtained through simulation experiments. Finally, Section 8 summarizes the work reported in the paper and presents possible directions for future work.

# 2. Preliminaries

The topological structure of an interconnect network can be modeled by a graph. This fact has been universally accepted and used by computer scientists and engineers. Moreover, it has been practically demonstrated that the graph theory is a very powerful mathematical tool for designing and analyzing topological structure of interconnect networks. In this section, we will briefly recall some basic concepts and notation of the graph theory used in this paper as well as the corresponding backgrounds of fault-tolerant networks.

## 2.1 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 $k$ [2, 3]. Each node is connected via communication links to

two other nodes in each dimension. The neighbors of the node $x=(x_{n-1}, \ldots, x_0)$ in $DIM_i$ ($0 \le i \le n$) are $x=(x_{n-1}, \ldots, x_{i+1}, x_{i\pm1}, x_{i-1}, \ldots, x_0)$ 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 $<x, y>$ and virtual channels of class $i$ as $VC_i$. We assume that a crossbar is used in each router to connect its input channels to its output channels.

## 2.2 The Fault Model

In recent years, many studies have addressed several issues in the field of fault-tolerance and reliability analysis of large-scale parallel and distributed systems. These researches span a wide range of systems such as massively parallel processors [13], cluster-based systems [14], mobile systems [15], sensor networks [16], and more recently Mp-SoCs [17]. Fault-tolerant designs of these systems aim at providing continuous operations in the presence of faults by allowing the graceful degradation of system. The fault-tolerant computing literature is extensive and thorough in the definition of fault models for the treatment of faulty digital systems. One approach in fault characterization is by its *nature*, *duration* and *extent*. Classification by nature is either *random* or *systematic* faults. Random faults are usually hardware faults in components of a system, which occur with a certain probability, while systematic faults such as software failures are faults, which are not random, and either a component has it, or it does not have it [1-3, 8, 10, 12]. Faults can be classified by their duration as *transient* and *permanent* faults. Transient faults persist in the system for only a short duration, while permanent faults remain in the system until it is repaired and may be either *static* or *dynamic*. Static faults are presented in the network when the system is powered on. Dynamic faults appear randomly during the operation of the system. Finally, classification by extent is either *localized* faults that affect only a single hardware or software module or *global* faults, which have effects that permeate through the entire system [1-3, 10-12, 18].

In a network there exist two classes of faults: either the entire processing element (PE) along with 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* [3, 10]. On a node failure occasion, all physical links incident on the failed node are also marked faulty at adjacent routers [3, 10]. Adjacent faulty nodes are coalesced into *fault regions*, which may lead to different patterns of failed components. Faulty regions extended by faulty components, may form convex or concave shape [1-3, 10, 12].

In this paper, we focus on the block fault model, which is suitable for modeling faults at the chip, multi-chip modules, and board level in networks with grid structures, particularly in torus, and mesh topologies. Furthermore, our approach tolerates only node failures since the structure of nodes are more complex than links and thus possess higher failure rates [2, 3]. We assume fault patterns are static and do not *disconnect* the network. A network is disconnected if there exist two nodes without any fault-free path to route messages between them [1-3].

## 2.3 The Boppana-Chalasani's Scheme

Chalasani and Boppana [1] have presented an efficient method to enhance wormhole switching for deadlock-free fault-tolerant routings in the torus networks. They considered arbitrarily-located faulty blocks and assumed only local knowledge of faults. Messages are routed via shortest paths when there are no faults, and this constraint is only slightly relaxed to facilitate routing in the presence of faults. The key concept used in their method is that, for each connected fault region of the network, it is feasible to connect the fault-free components around the fault to form a ring. This is the fault ring, f-ring, for that fault and consists of the fault-free nodes and channels that are adjacent (row-wise, column-wise, or diagonally) to one or more components of the fault region. The f-rings provide alternate paths to messages blocked by faults and used to route messages around rectangular fault regions. At most four additional virtual channels are sufficient to make any adaptive algorithm tolerate multiple block faults in torus networks. They have shown that for a fully adaptive algorithm good performance can be obtained with as many as 10% failures.

# 3. Fault-tolerant Routing Algorithms in Torus

In this section, we utilize the Chalasani-Boppana's scheme, in which any adaptive routing algorithm can tolerate multiple block fault regions in a torus network. Our approach employs a number of prominent adaptive routings as many as possible to route messages. When a message is blocked by faults at node $x$ and there is no fault-free link $<x, y>$ such that the hop from $x$ to $y$ is along the shortest path, the additional routing logic is required to route the message around the fault regions. To achieve this goal, we incorporate the routing scheme suggested by Chalasani and Boppana [1], as a well-known instance of a fault-tolerant routing widely used in the literature to make any adaptive routing algorithm tolerate to multiple block faults in the torus networks.

Many researchers are investigating suitable adaptive wormhole switching algorithms for high performance and fault-tolerant routing in ($k$, $n$)-tori and other networks. Most of their results are on the design of adaptive wormhole switching algorithms using a 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 and fault-tolerant routing algorithms. To the best of our knowledge, this work is the first effort in characterizing a variety of adaptive routings with respect to their fault-tolerant performance and design trade-offs. To further illustrate our evaluation, we consider five well-known basic routing algorithms; two hop-based routings (i.e., PHop, NHop), Duato's routing, Minimal routing, Fully-Adaptive routing, and four other sets of improved algorithms based on the basic routings; and we will show how these algorithms may affect the performance of a network.

***The Positive-Hop (PHop) routing scheme***

In the well-known PHop algorithm (originally proposed for store-and-forward switching), the number of buffer classes in

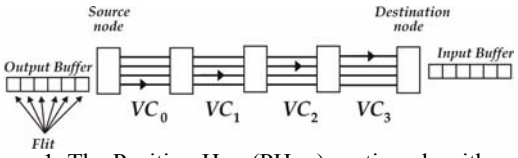


Figure 1. The Positive-Hop (PHop) routing algorithm

each node is equal to the diameter of the network plus one.

Thus, for ($k$, $n$)-torus networks the number of buffer classes is equal to $n\lfloor k/2 \rfloor + 1$. A message, as soon as generated, is placed in the buffer of class 0 in the source node. During the course of its journey towards its destination, the message occupies a buffer of class $i$ at an intermediate node if and only if the message has taken exactly $i$ hops to reach that intermediate node. A disadvantage of this algorithm is that, it requires a large number of classes of buffers. Figure 1 illustrates an example of the PHop routing algorithm [20].

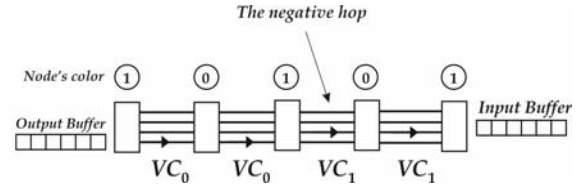


Figure 2. The Negative-Hop (NHop) routing algorithm

***The Negative-Hop (NHop) routing scheme***

The NHop wormhole-switched algorithm has been discussed in [18]. To employ the NHop algorithm, the network is colored, and each node is given a label corresponding to its color. A hop by a message is a negative hop if it moves from a node with higher label to a node with lower label. Any other hop is a non-negative hop. Messages when injected to the network have 0 negative hops and are routed minimally when there are no faults. If a message has taken $i \geq 0$ negative hops, then it uses virtual channels of class $i$ for its next hop (see Figure 2). The NHop uses $1+\lfloor n\lceil k/2 \rceil/2 \rfloor$ virtual channels and provides minimal fully adaptive routing in a fault-free torus. The advantage of the NHop scheme is that it requires fewer buffer classes than that of the PHop.

# 4. Modifications of the basic routing algorithms

The hop-based fully adaptive routings (PHop and NHop) described in the previous section do not utilize virtual channels because they start their journey originating from virtual channel 0. However, very few packets take the maximum number of hops (network diameter) and use all the virtual channels. Virtual channels with lower number are utilized more than virtual channels with higher numbers. Figure 3 depicts the average usage of different virtual channels in an 8×8 torus under uniform traffic. In our simulation methods, the channel usage is identical for all traffic generation rates and any virtual channel set size.

The PHop and NHop routing schemes can be improved by giving each header flit a number of *bonus cards*. In the PHop routing algorithm with bonus cards (Pbc), the number of bonus cards is equal to the diameter of the network minus the number of hops it is going to take. For the NHop scheme with bonus cards (Nbc) it is equal to the maximum possible negative hops minus the number of required negative hops to reach the destination node. A message with no bonus cards, is routed exactly the same as in the cases of PHop and NHop algorithms. In routing a message with $b$ bonus cards, ($b \geq 1$), any of virtual channels numbered 0, 1, …, $b$ can be used for the first hop of the message. Thus, a message with bonus cards has a wider choice of virtual channels and is likely to choose the least congested one for the first hop. The average usage percent of virtual channels for the basic routing algorithms using bonus cards has been illustrated in Figure 3.

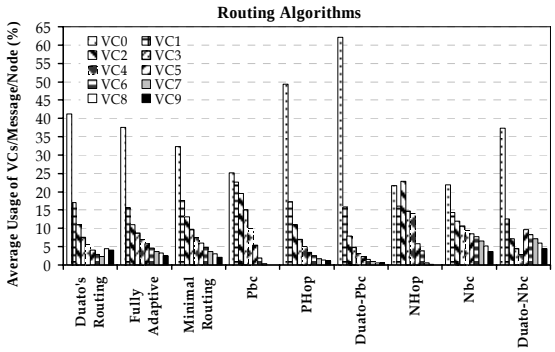


Figure 3. Virtual channel utilization under uniform traffic in an 8×8 torus for adaptive routing algorithms with 64-flit message length and 10 virtual channels per physical channel

## 4.1 Increasing Adaptivity of the Routing Algorithms

We have used Duato's protocol [2, 3, 19] for designing fully adaptive routing algorithms to further improve the adaptivity of the basic algorithms (PHop, NHop) and those improved with bonus cards (Pbc, Nbc). According to the Duato's approach, virtual channels are divided into adaptive (class $a$) and deadlock-free (class $b$) virtual sub-networks. At each step, a message visits adaptively any available virtual channel from class $a$. If all the virtual channels belonging to class $a$ are busy it visits a virtual channel from class $b$ using a deadlock-free routing algorithm described in Section 3 (basic and improved ones with bonus cards). The virtual channels of class $b$ define a complete deadlock-free virtual sub-network, which acts like as a *drain* for the sub-network built from virtual channels belonging to class $a$. If we employ PHop and NHop routing schemes (with or without bonus cards), at least $n\lfloor k/2 \rfloor + 1$ and $1+\lfloor n\lceil k/2 \rceil/2 \rfloor$ virtual channels are required for class $b$, respectively. For example, to implement the PHop and NHop algorithms on an 8×8 torus, these values are equal to 9 and 5 virtual channels per physical channel, respectively. Network performance is maximized when the extra virtual channels are added to adaptive virtual channels in class $a$ [2, 3, 19]. Therefore, the best performance is achieved when class $b$ contains minimum required virtual channels and extra virtual channels are allocated to class $a$.

In Figure 3, we use the Pbc and Nbc algorithms (PHop and NHop with bonus cards) for routing using class $b$ virtual

channels. This figure illustrates the average number of virtual channels used for a network with 64 nodes when Pbc and Nbc routing schemes are integrated with Duato's routing as discussed above. In this figure, virtual channels 0 ($VC_0$) and 1 ($VC_1$) belong to class *a*, and other virtual channels belong to class *b*.

## 5. Simulation Results

To compare the performance issues of the routing algorithms, we have developed an event-driven simulator at flit-level. The simulator is written in C# language and can be used for wormhole switching in (*k*, *n*)-torus with and without faults. The crossbar switch in the router allows multiple messages to traverse a node simultaneously. Virtual channels that have messages to transmit use the physical channel in a round robin manner. Each virtual channel uses one input buffer and one output buffer. Each buffer can store one flit. Also the small capacity is selected to minimize the router complexity and to reduce its delay. Also it takes one network cycle to transmit a flit between neighbors.

To limit the search space, we have fixed some important parameters in the following performance comparisons. We have conducted our simulations for $N=8^2$ tori. In the literature, fixed-length messages with 32, 64, or 100 flits are commonly considered [8-12, 18]. We have considered 64-flit messages in this study. The destination of each message is distributed randomly, and the message inter-arrival times are distributed exponentially. We have also considered uniform traffic pattern. This type of traffic could be representative of the traffic generated in massively parallel computations in which array data are distributed among the nodes using hashing techniques [1-3, 7-12, 18]. We consider only node failure and assume that a faulty node is randomly generated subject to the overlapping fault-ring model. We also assume that faults are non-malicious, fault patterns are static [1, 3, 8-10] and do not disconnect the network. Therefore only non-faulty nodes generate messages. Further, messages are destined only to fault-free nodes. These assumptions are commonly made in fault analysis in the literature [1, 3, 8-18]. We ran each simulation for 300,000 flit times and sufficient warm-up times (by discarding the information obtained during the first 10,000 flit times) provided to allow the network to reach the steady state. After the warm-up time, the network traffic is sampled at periodic intervals and the performance measures have been gathered through the simulation.

The most important performance measures are message latency and network throughput. We use normalized throughput and average message latency as the performance metrics. The normalized throughput is defined as the number of messages received over the number of messages that can be transmitted at the maximum load [2]. The normalized throughput and average message latency achieved are plotted against traffic generation rate in Figures 4 and 5 for uniform traffic using 64-flit messages and 10 virtual channels per physical channel. For low traffic load, all nine algorithms have the same latency. However, the six hop-based schemes and the other three algorithms behave differently during and after saturation. The two algorithms achieved from store-and-forward switching have similar throughputs with NHop being slightly better: NHop has better latency and throughput in saturation. In particular, NHop being saturate after 0.066 and PHop shows signs of saturation at about 0.045.

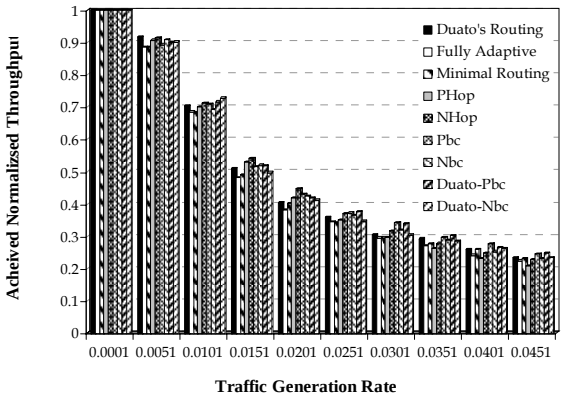



Figure 4. Comparison between the throughput of routing algorithms against the traffic load in an 8×8 torus with 64-flit message length and 10 virtual channels per physical channel

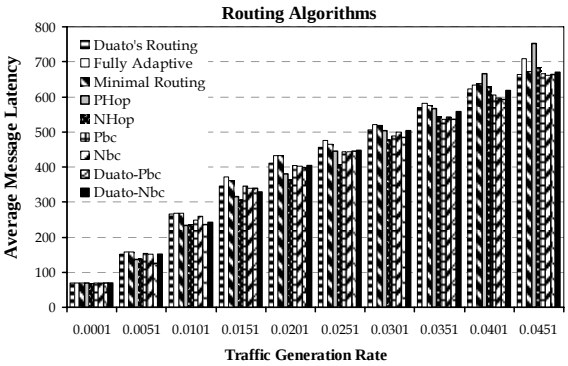



Figure 5. The average message latency of adaptive routing algorithms against the traffic load in an 8×8 torus using 64-flit message length and 10 virtual channels per physical channel

The latencies of the other algorithms rise abruptly at the point of saturation. Furthermore, the achieved throughputs of the three algorithms increase steadily. The NHop and Duato-Nbc algorithms achieve their peak throughputs of 0.35 and 0.36, respectively, at 100% traffic load and the Nbc algorithm does not exhibit better performance than the Duato's routing. The Fully-Adaptive algorithm has lower peak throughput than Minimal routing and saturates more quickly. The Minimal routing has a peak throughput of 0.34, which occurs at traffic rate of 0.025. Another observation from Figure 4 is that Duato's routing performs better than Minimal and Fully-Adaptive routings. It is evident from the curves that the fully adaptive hop-based schemes augmented with Duato's methodology (i.e., Duato-Pbc, and Duato-Nbc) yield better throughputs compared to the other routing algorithms for the conditions considered in this study. This could be due to the use of more virtual channels per physical channel in class b, balancing the traffic load on virtual channels. For instance, Duato-Pbc gives better throughput and also uses more virtual channels than any other algorithm.

The average message latency for an 8×8 torus is plotted against offered traffic in Figure 5. In this configuration we need 9 and 5 virtual channel classes for PHop and NHop

routings, respectively. We have also used 1 and 2 virtual channels in each class, respectively. Therefore, in this network each physical channel has 10 virtual channels that can ensure a fair comparison under almost equal hardware cost. Network configuration for Pbc and Nbc algorithms is the same as that for the PHop and NHop routing algorithms.

Figure 5 reveals that for low and medium traffic loads all six routing algorithms have the same latency. But they begin to behave differently around the saturation region. The basic algorithms with bonus cards (Pbc and Nbc) have better performance than the two basic algorithms. This figure also depicts that under an equal number of virtual channels per physical channel, Duato-based improved routing algorithms have higher throughputs and lower latencies compared to the basic algorithms with bonus cards. It can be seen that Duato's methodology with Nbc (Duato-Nbc) has a better performance, due to the fact that in the Nbc routing scheme, there are more virtual channels in the adaptive class (i.e., class *a*).

## 5.1 Performance Study of the Routing Algorithms with Faults

We have simulated an 8×8 torus with 5% and 10% of total network nodes faulty. In each case, we have randomly generated the required number of faulty nodes. To see the performance degradation with faults, we have also simulated the routing algorithms on a fault-free torus. The simulation results reported in this section are for the mentioned above algorithms fortified with the Chalasani-Boppana's scheme. Comparative performance across different fault cases is specific to the fault sets used. Therefore, we have further simulated the routing algorithms for mentioned above fault cases. For each case, we have simulated 10 different fault sets for 100% traffic load. The values obtained from the 10 different fault sets are averaged and shown in Figures 6 and 7.

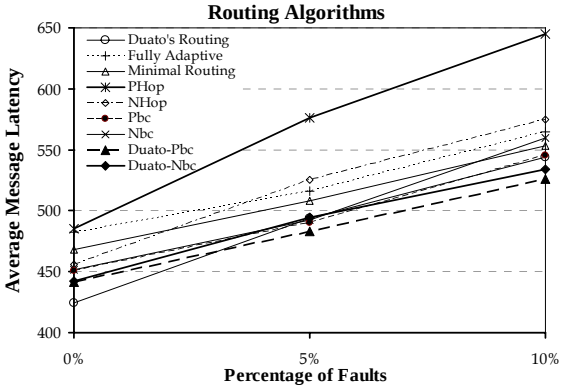



Figure 6. The average message latency of adaptive routing algorithms against the traffic load in an 8×8 torus with 64-flit message length, 10 virtual channels per physical channel, and various fault cases 0%, 5%, and 10%

As the number of faults is increased, the latency increases and the throughput also drop steadily. By comparing the fault-free and 10% fault cases, we find that Duato-Nbc has 21% increase in latency and 24% decrease in throughput. For the fault-free network, the Duato-Nbc has a peak utilization of 0.36 at a latency of 535 cycles. The Duato-Nbc shows the

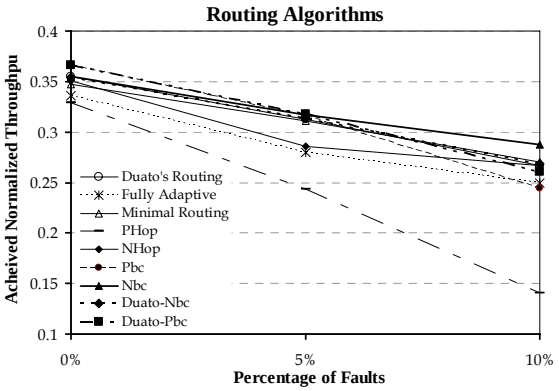



Figure 7. Comparison between the throughput of adaptive routing algorithms against the traffic load for an 8×8 torus using 64-flit message length, 10 virtual channels per physical channel, and various fault cases 0%, 5%, and 10%

graceful degradation of performance in the presence of faults. Moreover, In the case of failure, Nbc gives better throughput than the NHop and PHop due to utilizing the fewer virtual channels. A possible explanation is that the load on virtual channels is balanced in Nbc but not in NHop and PHop algorithms.

## 5.2 Traffic Analysis of Routing Algorithms Around Fault-rings

Interconnect networks are based on the most regular topologies. Routing algorithms, switching techniques, and even deadlock freedom proofs benefit from the regularity of such networks, which would be affected by existence of faulty components. In regular networks with uniform traffic, network resources are used evenly among all the nodes. However, in the presence of faults, traffic is jammed in some regions of network and creates bottlenecks. These local bottlenecks may degrade the overall performance of network especially in case of wormhole-switched networks. These bottlenecks may cause uneven power dissipation in Networks-on-Chip (NoCs) and make hotspots in some regions in the chip. It is therefore essential for a fault-tolerant routing algorithm to be equipped with a comprehensive traffic analysis. Unfortunately, it is a time consuming process and requires advanced tools to examine the traffic around fault regions. Therefore, many of researchers did not perform a detailed analysis of traffic around fault regions [1, 8-12, 16]. In this section, in an attempt to gain a deep understanding of the issue of traffic analysis on faulty networks, we investigate the traffic load distribution around fault-rings.

Figure 8 depicts the simulation results on the distribution of traffic load around fault-rings. Three fault regions with a row-wise overlapping are considered: a region of size 3×2 and two regions of size 1×1. For the fault-free network, NHop, Nbc, and Duato-Nbc are very attractive, with peak traffic load 39%, 40.9%, and 39.9%, respectively. However, faults may affect the performance of Nbc and Duato-Nbc significantly. As depicted in this figure, performance degradation in the routing algorithms is mainly related to some bottlenecks in small areas of network especially at the corners of fault rings. This can be explained as follows. The

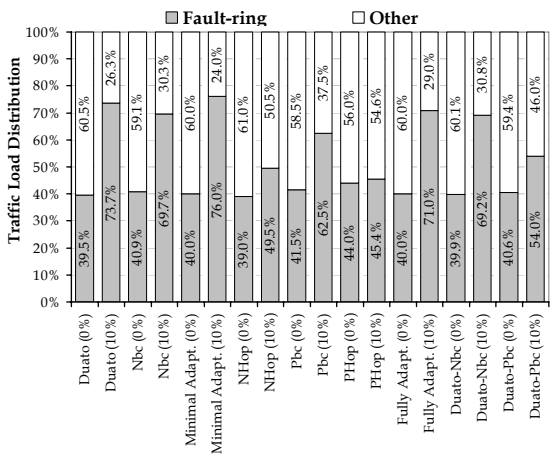



Figure 8. Traffic load distribution for adaptive routing algorithms around fault-rings in an 8×8 torus using 64-flit message length, 10 virtual channels per physical channel, and various fault cases 0%, and 10%

bottleneck in a corner of fault region could propagate traffic to neighbor nodes and increase the total network latency. The reason behind this behavior has a root in the nature of wormhole switching, which has a great potential to propagate effects of regional bottlenecks into the entire network. Moreover, there is heavy a competition for channels around each fault-ring, which makes each fault-ring act like a hotspot. For example, the performance of Minimal routing with a peak traffic of 75.5% is worse than other adaptive routings. The interesting point is that, messages that are routed via this routing are unable to avoid fault-rings along their shortest paths and this event reduces throughput and increases latency even for normal messages, which may be waiting for the channels reserved by misrouted messages prior to misrouting. Therefore, the effects of hotspots observed for Minimal routing are more severe compared with the other routing algorithms. To alleviate the hotspot problem, we can use more virtual channels. However, the hardware cost will also be increased.

# 6. Analytic Performance Modeling

In this section, we present an analytical model for fully adaptive wormhole routing in a torus network. Due to superior performance of Duato-Nbc algorithm, our analysis focuses on this routing scheme but the proposed modeling approach can be applied for the other routing algorithms after few changes in the model. The most important performance measure in our model is the average message latency.

## 6.1 Model Assumptions

The model makes assumptions, which are commonly used in the literature [21-26], and are listed below.

(i) Messages destinations are uniformly distributed across the network nodes.

(ii) Nodes generate traffic independently of each other, following a Poisson process with an average rate of $\lambda_g$ messages per cycle.

(iii) The message length is fixed at $M$ flits, each of which requires one cycle transmission time between two adjacent routers.

(iv) The local queue at the injection channel in the source node has infinite capacity. Messages at the destination node are transferred to the local PE one at a time through the ejection channel.

(v) $V$ virtual channels per physical channel are used. These virtual channels are used according to the Duato-Nbc routing scheme, which are divided into $V_1$ and $V_2$ classes ($V = V_1 + V_2$). This number of virtual channels yields the optimal performance compared to the other algorithms proposed in [3, 18, 19].

In the following section, we derive the mathematical model that approximate the behavior of 2-D torus communication system using Duato-Nbc routing algorithm.

## 6.2 The Proposed Analytical Model

The average message latency is composed of the average network latency, $\bar{S}$, which is the time to cross the network and the average waiting time seen by the message in the source node, $\bar{W}_s$, before entering the network. However, to capture the effects of virtual channels multiplexing, the mean message latency has to be scaled by a factor, say $\bar{V}$, representing the average degree of virtual channels multiplexing, that takes place at a given physical channel. Therefore, the average message latency can be approximated as [23]

$$Average\ Message\ Latency = (\bar{S} + \bar{W}_s)\bar{V} \tag{1}$$

In what follows, we will describe the calculation of $\bar{S}$, $\bar{W}_s$, and $\bar{V}$.

### 6.2.1 Calculation of the Mean Network Latency

Under uniform traffic pattern, the average number of channels that a message visits along a given dimension and across the network, $\bar{k}$, $\bar{d}$ respectively, are given by Agarwal [24]

$$\begin{cases} \bar{k} \approx k/4 \\ \bar{d} = 2\bar{k} \end{cases} \tag{2}$$

Fully adaptive routing allows a message to use any available channel that brings it closer to its destination resulting in an evenly distributed traffic rate on all network channels. The mean arrival rate, $\lambda_c$, on a given channel is determined as follows. A PE generates, on average, $\lambda_g$ messages in a cycle, which are evenly distributed among the 4 output channels. Since each message travels, on average, $\bar{d}$ hops to cross the network, we can write the rate on a channel, $\lambda_c$, as

$$\lambda_c = \lambda_g \bar{d} / 4 \tag{3}$$

Since the torus topology is symmetric, averaging the network latencies seen by the messages generated by only one node for all other nodes gives the average message latency in thenetwork. Let $s = (s_x, s_y)$ be the source node and $d = (d_x, d_y)$ denotes a destination node such that $d \in G - \{s\}$ where $G$ is

the set of all nodes in the network. We define the set $H = \{h_x, h_y\}$, where $h_x$ and $h_y$ denote the number of hops that the message makes along $DIM_x$ and $DIM_y$, respectively, i.e. $(s_x + h_x) \bmod k = d_x$, and $(s_y + h_y) \bmod k = d_y$.

$$\begin{cases} h_x = \|s_x - d_x\| \\ h_y = \|s_y - d_y\| \end{cases} \tag{4}$$

where $\|x - y\|$ denotes the distance between a source node $x$ and a destination node $y$.

The network latency, $S_H$, seen by a message crossing the network from node $s$ to node $d$ consists of two parts: the first part is the actual message transmission time, and the second part is the blocking time in the network. Therefore, $S_H$ can be written as

$$S_H = M + \|H\| + \sum_{h=1}^{\|H\|} B_h \tag{5}$$

where $M$ is the message length, $\|H\|$ is the distance (in terms of hops made by the message) between the source and the destination nodes, and $B_h$ is the blocking time seen by a message in its $h^{\text{th}}$ hop. The terms $\|H\|$ and $B_h$ are given by

$$\|H\| = h_x + h_y \tag{6}$$

$$B_h = P_{block_h} \bar{W}_c \tag{7}$$

where $P_{block_h}$ is the probability that a message is blocked at its $h^{\text{th}}$ hop channel, and $\bar{W}_c$ is the average waiting time to acquire a virtual channel in the event of blocking. Let us now calculate the probability $P_{block_h}$. To do so, let $\varphi_h$ be the number of dimensions, or output channels, that a message still has to visit when crossing the $h^{\text{th}}$ hop channel. The calculation of $\varphi_h$ has been derived in [25]. We recollect briefly here the main equations for the calculation of $\varphi_h$. The number of channels, and thus the number of virtual channels, that a message can select at a given hop depends on the number of dimensions still to be visited. When a message arrives at the $h^{\text{th}}$ hop channel, it has already made $(h-1)$ hops and has crossed, say, $t$ dimensions, $(0 \le t \le 1)$. At its next hop, the message can use $(2-t)$ channels at the remaining $(2-t)$ dimensions still to be visited. Averaging over all of the possible cases yields the number of channels, $\varphi_h$, that the message can select when crossing the $h^{\text{th}}$ hop channel, $(1 \le h \le \bar{d})$, as

$$\varphi_h = \sum_{t=0}^{1} (2-t)\, \psi_h^t \tag{8}$$

where $\psi_h^t$ is the probability that the message has entirely crossed $t$ dimensions along on its $h$-hop path. This probability is a function of the number of dimensions that the message has still to cross. Hence, to calculate $\psi_h^t$ we need first to find how many dimensions remain for the message to reach its destination. The number of alternative routes that a message can select, at its next hop, to advance towards its destination depends on the number of hops already made in both dimensions. When a message has made $h\,(1 \le h \le \bar{d})$ hops, these hops can be a combination of $(h_x, h_y)$ hops, with $h_x$ and $h_y$ being the number of hops achieved in the first and second dimensions respectively, so that $(h_x + h_y = h)$ and $(0 \le h_x, h_y \le \bar{k})$.

To compute the probability that a message has crossed all the channels of one dimension, two cases need to be considered:

(a) When $(0 \le h < \bar{k})$, the number of $(h_x, h_y)$ combinations is $(h+1)$. In this case, the message still has to cross channels in both dimensions and therefore, can choose among adaptive virtual channels of both dimensions.

(b) When $(\bar{k} \le h < \bar{d} - 1)$, the number of $(h_x, h_y)$ combinations is $(\bar{d} - h + 1)$. In only two cases, $(\bar{k}, h - \bar{k})$ out of these combinations, the message has crossed all channels of one dimension and thus, for the remaining hops, the message only crosses channels of the other dimensions.

The probability that there remains only one dimension to cross a message which is $h$ hops away from its destination, $P_{\varphi_h}$, can be expressed as

$$P_{\varphi_h} = \begin{cases} \dfrac{2}{\bar{d} - h + 1} & \bar{k} \le h < \bar{d} - 1 \\ 0 & 0 \le h < \bar{k} \end{cases} \tag{9}$$

Therefore, the probability that the message has entirely crossed $t$ dimensions along on its $h$-hop path is given by

$$\psi_h^t = \begin{cases} 1 - P_{\varphi_h} & t = 0 \\ P_{\varphi_h} & t = 1 \end{cases} \tag{10}$$

A message is blocked at a given channel when all the adaptive virtual channels of Class $a$ and also $V_2 - \lceil \Delta/2 \rceil + 1$ virtual channels of Class $b$ (which are used for Nbc algorithm) are busy, where $\Delta$ is the number of remaining hops to reach the destination. When blocking occurs a message has to wait for all $V_1$ fully adaptive virtual channels of Class $a$, and $V_2 - \lceil \Delta/2 \rceil + 1$ virtual channels of Class $b$. In order to calculate $P_{block_h}$, we need to categorize messages into three classes based on their previous and next hop.

- **Class 1:** This class contains messages which have used a virtual channel of class $a$ in their previous hop (with probability $P_{block_1}$).
- **Class 2:** This class contains messages which have used a virtual channel of Class $b$ in their previous hop and their next hop is negative (with probability $P_{block_2}$).
- **Class 3:** This class contains messages which have used a virtual channel of Class $b$ in their previous hop and their next hop is positive (with probability $P_{block_3}$).

Since the number of messages in Class 2 is identical to the number of messages in Class 3, we can write the probability of message blocking as

$$P_{block_h} = \left( P_{block\,1} + \frac{1}{2}\left(P_{block\,2} + P_{block\,3}\right) \right)^{\varphi_h} \tag{11}$$

In what follows, we compute $P_{block_1}$, $P_{block_2}$, and $P_{block_3}$.

When a message used a virtual channel of Class *a* in its previous hop, it can use any of $V_1$ virtual channels of Class *a*, and also $V_2 - \lceil \Delta/2 \rceil + 1$ virtual channels of Class *b*. The blocking occurs, when all $V_1 + V_2 - \lceil \Delta/2 \rceil + 1$ virtual channels at a given physical channel are busy. We can therefore write

$$P_{block_1} = \sum_{v=V_1+V_2-\lceil \Delta/2 \rceil+1}^{V} P_{v_0} P_v \frac{\binom{\lceil \Delta/2 \rceil - 1}{v - V_1 - V_2 + \lceil \Delta/2 \rceil - 1}}{\binom{V}{v}} \tag{12}$$

where $P_v$, ($0 \le v \le V$), and $P_{v_0}$ denote the probability that $v$ virtual channels at a physical channel are busy (which is calculated later in Section 6.2.3), and the probability that the message uses a virtual channel of Class *a* in its previous hop, respectively. $P_{v_0}$ is given by

$$P_{v_0} = \frac{V_1}{V_1 + V_2 - \lceil \Delta/2 \rceil + 1} \tag{13}$$

Moreover, a message belonging to the Classes 2 and 3 which has already used virtual channel *l* at its previous hop, can employ any of $V_1$ virtual channels of Class *a*, and also $V_2 - \lceil \Delta/2 \rceil - l + 1$ and $V_2 - \lceil \Delta/2 \rceil - l + 2$ virtual channels of Class *b*, respectively. Therefore, the $P_{block_2}$ and $P_{block_3}$ are obtained by

$$P_{block_2} = \sum_{l=1}^{V_2 - \lceil \Delta/2 \rceil} \sum_{v=V_1+\lceil \Delta/2 \rceil+l}^{V} P_{v_l} P_v \frac{\binom{V_2 - \lceil \Delta/2 \rceil - l}{v - V_1 - \lceil \Delta/2 \rceil - l}}{\binom{V}{v}} \tag{14}$$

$$P_{block_3} = \sum_{l=1}^{V_2 - \lceil \Delta/2 \rceil + 1} \sum_{v=V_1+\lceil \Delta/2 \rceil+l+1}^{V} P_{v_l} P_v \frac{\binom{V_2 - \lceil \Delta/2 \rceil - l - 1}{v - V_1 - \lceil \Delta/2 \rceil - l - 1}}{\binom{V}{v}} \tag{15}$$

where $P_{v_l}$ indicates the probability that a message has used virtual channel *l* at its previous hop and is given by

$$P_{v_l} = \frac{1}{V_1 + V_2 - \lceil \Delta/2 \rceil + 1} \qquad 1 \le l \le V_2 - \lceil \Delta/2 \rceil + 1 \tag{16}$$

The average network latency, $\bar{S}$, is obtained by averaging $S_H$, which is the average network latency of $H$ hops messages, over the $(N-1)$ possible destination nodes in the network. Therefore, $\bar{S}$ can be determined as

$$\bar{S} = \frac{1}{N-1} \sum_{d \in G - \{s\}} S_H \tag{17}$$

To determine the average waiting time to acquire a virtual channel, a physical channel may be treated as an M/G/1 queue [22]. Since the minimum service time at a channel is equal to the message length, *M*, following a suggestion proposed in [26], the variance of the service time distribution can be approximated by $(\bar{S} - M)^2$; where $\bar{S}$ is the average service time at a given channel and can be calculated as the mean of $S_H$ of all source and destination nodes that have at least one path between each other that traverse the channel. Hence, the average waiting time becomes

$$\bar{W}_c = \frac{\lambda_c \bar{S}^2 \left(1 + \frac{(\bar{S} - M)^2}{\bar{S}^2}\right)}{2(1 - \lambda_c \bar{S})} \tag{18}$$

### 6.2.2 Calculation of the Average Waiting Time at the Source Node

In this section, we compute the average waiting time at the source node ( $\bar{W}_s$ ). We assume that, messages have an arbitrary (but known) length or service distribution. However, the arrival process will be taken to be Poisson, a single server is assumed, and the queue buffer size is taken to be infinite. Such a queue is called M/G/1 queue, using Kendal notation [22]. Since a message in the source node can enter the network through any of the $V$ virtual channels, the average arrival rate to the queue is $\lambda_g/V$. Applying the Pollaczek-Khinchine (P-K) mean value formula [22] yields the average waiting time experienced by a message at the source node as

$$\bar{W}_s = \frac{\frac{\lambda_g}{V} \bar{S}^2 \left(1 + \frac{(\bar{S} - M)^2}{\bar{S}^2}\right)}{2\left(1 - \frac{\lambda_g}{V} \bar{S}\right)} \tag{19}$$

### 6.2.3 Calculation of the Average Degree of Virtual Channels Multiplexing

The probability, $P_v$ ($0 \le v \le V$), that $v$ virtual channels at a given physical channel are busy can be determined using a Markovian model (details of the model can be found in [21, 23, 25]). In the steady state, the model yields the following probabilities [21].

$$P_v = \begin{cases} Q_0 = 1 \\ Q_v = Q_{v-1} \lambda_c \bar{S} & (0 < v < V) \\ Q_v = \dfrac{Q_{v-1} \lambda_c}{1/\bar{S} - \lambda_c} & (v = V) \\ P_0 = \left( \sum_{l=0}^{V} Q_l \right)^{-1} \\ P_v = P_{v-1} \lambda_c \bar{S} & (0 < v < V) \\ P_v = \dfrac{P_{v-1} \lambda_c}{1/\bar{S} - \lambda_c} & (v = V) \end{cases} \tag{20}$$

When multiple virtual channels are used per physical channel they share the bandwidth in a time multiplexed manner. The average degree of virtual channel multiplexing, that takes place at a given physical channel, can be estimated by [21]

$$\overline{V} = \frac{\sum_{v=1}^{V} v^2 P_v}{\sum_{v=1}^{V} v\, P_v} \tag{21}$$

# 7. Validation of the Model

To further understand and evaluate the performance issues of the Duato-Nbc routing algorithm, we have used the simulator described in Section 5. The simulator uses the same assumptions as the analysis, and these assumptions were detailed in Section 6.1. Extensive validation experiments have been performed for several combinations of network sizes, message lengths, and virtual channels. However, for the sake of specific illustration, we have conducted our simulations for the following cases only:

- Network size $N = 8^2$ and $16^2$ nodes.
- Number of virtual channels $V$ = 10 per physical channel.
- Message length $M$ =32 and 64 flits.

The average message latencies obtained from the simulation and analytical results are plotted in Figure 9 for different network sizes, and message lengths with 10 virtual channels per physical channel.

The *x*-axis in this figure represents the traffic rate injected by a given node in a cycle ($\lambda_g$) while the *y*-axis shows the average message latency (in terms of flit cycles). Each graph illustrates the simulation results and the analytical model prediction for two message length $M$ = 32 and 64 flits. The figure indicates that the simulation results and the values predicted by the analytical model are in good agreement (lower than 5%) under steady state regions, i.e. under light and moderate traffic and near the saturation point. However, due to the approximations that have been made in the analysis to ease the model development, some discrepancies (of at most 15% error) are apparent around the saturation point. One of the most significant term in the model under heavily loaded network, is the average waiting time in the source queue. The approximation which is made to compute the variance of service time received by a message at a given channel is a factor of the model inaccuracy. Since the independent assumptions are essential in ensuring a tractable model, and given that most evaluation studies concentrate on network performance in the steady state regions, it can be concluded that the present analytical model constitutes a cost-effective evaluation tool for assessing the performance behavior of fully adaptive routing algorithms in torus networks.

# 8. Conclusions

In this paper, we have investigated the performance of several wormhole-switched adaptive routing algorithms fortified with the fault-tolerant Chalasani-Boppana's routing scheme. We have considered nine adaptive algorithms which are divided into two classes, namely basic and modified. PHop, NHop, Minimal routing, Fully-Adaptive routing, and Duato's routing are five routings as the basic algorithms and Nbc, Pbc, Duato-Pbc and Duto-Nbc are modified algorithms. Minimal and Fully-Adaptive routings use the fewest virtual channels in tori. The basic algorithms PHop and NHop were

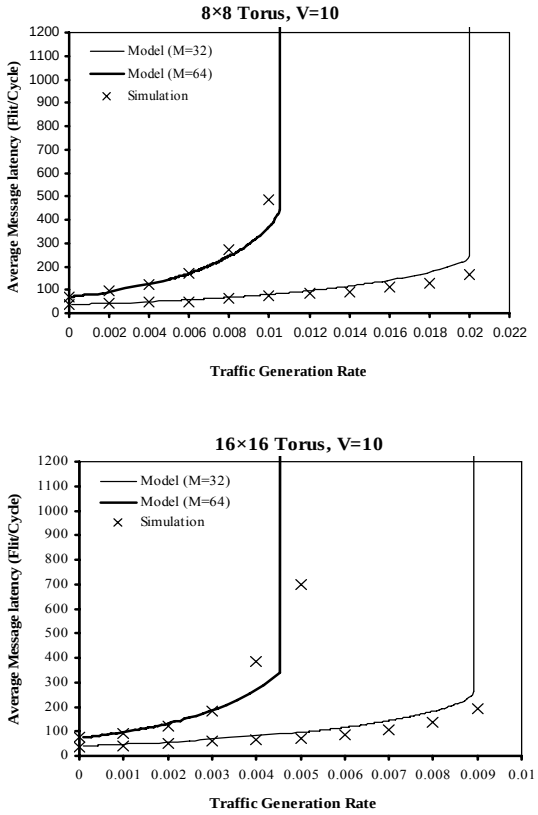



Figure 9. Average message latency calculated by the analytical model against those obtained from simulation for 8×8 and 16×16 torus networks using Duato-Nbc routing with message length $M$=32, 64 flits, and $V$=10 virtual channels per physical channel

improved with bonus cards (Pbc and Nbc). Moreover, these four algorithms are designed based on Duato's routing algorithm. Among these, NHop and Nbc use the lowest number of virtual channels, which is nearly half of the network diameter. PHop and Pbc use the highest number of virtual channels that equals to the diameter of the network. Furthermore, improved algorithms with bonus cards try to balance the traffic load on virtual channels, which is a feature not given much attention previously. For uniform traffic, the improved algorithms with bonus cards have better performance than the other basic algorithms. Accordingly, Duato's methodologies that utilize the basic algorithms with bonus cards have the best performance. A comparison between the basic algorithms and those obtained from the bonus cards reveals that balancing the traffic load over virtual channels yields higher throughput and lower latency.

More specifically, using Duato's protocol for routing deadlock-free virtual channels with Nbc algorithm gives the best performance. We showed that the Duato-Nbc approach is a good candidate for incorporating into the design of fault-tolerant routing algorithms in parallel computer networks. Thus, we derived a mathematical model to predict the message latency in torus when Duato-Nbc routing algorithm is used. Simulation experiments confirmed that the message latency results captured by the analytical model are in good agreement with those obtained through simulation experiments. We showed that the proposed analytical model manages to achieve a good degree of accuracy while

maintaining simplicity, making it a practical evaluation tool that can be used by the researchers in the field to gain insight into the performance behavior of fully adaptive routings in wormhole-switched tori. Our next objective is to investigate the performance behavior of different proposed routing algorithms for faulty networks through analytical models.

## References


[1] S. Chalasani, R.V. Boppana, "Adaptive wormhole routing in tori with faults," *IEE Proc.- Comput. Digit. Tech.*, Vol. 42, No. 6, pp. 386-394, Nov. 1995.

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

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

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

[5] W.J. Dally, H. Aoki, "Deadlock-free adaptive routing in multicomputer networks using virtual channels," *IEEE Transactions on Parallel and Distributed Systems,* Vol. 4, No. 4, pp. 66-74, 1993.

[6] J. Duato, "Improving the efficiency of virtual channels with time-dependent selection functions," *Proceedings Parallel Architectures and Languages*, pp. 635-650, 1992.

[7] R.V. Boppana and S. Chalasani, "A Comparison of Adaptive Wormhole Routing Algorithms," *20th Annual Int'l Symp. on Computer Architecture*, pp. 351-360, 1993.

[8] M. E. Gómez, N. A. Nordbotten, J. Flich, P. López, A. Robles, J. Duato, Tor Skeie, Olav Lysne, "A Routing Methodology for Achieving Fault Tolerance in Direct Networks," *IEEE Trans. On Comput.*, Vol. 55, No. 4, pp. 400-415, 2006.

[9] A. Mejia, J. Flich, J. Duato, Sven-Arne Reinemo, Tor Skeie, "Segment-Based Routing: An Efficient Fault-Tolerant Routing Algorithm for Meshes and Tori," *IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, 2006.

[10] Y. J. Suh, B.V. Dao, J. Duato, S.Yalamanchili, "Software-based rerouting for fault-tolerant pipelined communication," *IEEE Trans. On Parallel and Distributed Systems*, Vol. 11, No. 3, pp. 193-211, 2000.

[11] J. M. Montañana, J. Flich, A. Robles, P. López, J. Duato, "A Transition-Based Fault-Tolerant Routing Methodology for InfiniBand Networks," *IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, 2004.

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

[13] S. Chakravorty, L. V. Kalé, "A Fault Tolerant Protocol for Massively Parallel Systems," *IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, 2004.

[14] J. N. Al-Karaki, "Performance Analysis of Repairable Cluster of Workstations," *IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, 2004.

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

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

[17] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, "Performance Evaluation and Design Trade-Offs for Network-on-Chip Interconnect Architectures," *IEEE Trans. Computers*, Vol. 54, No. 8, pp. 1025-1040, 2005.

[18] R.V. Boppana and S. Chalasani, "A Framework for Designing Deadlock-Free Wormhole Routing Algorithms," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 7, No. 2, pp. 169-183, 1996.

[19] 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.

[20] A. E. Kiasari, H. Sarbazi-Azad, M. S. Rezazad, "Performance comparison of adaptive routing algorithms in the star interconnection network," *IEEE HPC-Asia conference*, pp. 257-264, 2005.

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

[22] L. Kleinrock, *Queueing Systems*, Vol. 1, John Wiley, New York, 1975.

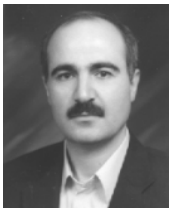
[23] M. Ould-Khaoua, "A Performance model for Duato's adaptive routing algorithm in *k*-ary *n*-cubes," *IEEE Trans. Computers*, Vol. 48, No. 12, pp. 1-8, 1999.

[24] A. Agarwal, "Limits on interconnection network performance," *IEEE Trans. Parallel & Distributed Systems*, Vol. 2, No. 4, pp. 398-412, 1991.

[25] H. Sarbazi-Azad, *Performance analysis of wormhole routing in multicomputer interconnection networks*, PhD Thesis, Computing Science Department, Glasgow University, 2001.

[26] J. Draper, J. Ghosh, "A comprehensive analytical model for wormhole routing in multicomputers systems," *Journal of Parallel and Distributed Computing (JPDC)*, Vol. 32, pp. 202–214, 1994.

**Farshad Safaei** received the B.Sc. and M.Sc. degrees in Computer Engineering both from Iran University of Science and Technology (IUST), Iran, in 1994 and 1997, respectively. He is currently pursuing his Ph.D. in Computer Systems Architecture at IUST. In addition, he is a Research Assistant in School of Computer Science, Institute for Studies in Theoretical Physics and Mathematics (I.P.M.), Iran. His research interests include high performance computer architectures, performance modeling/evaluation, VLSI design and low power synthesis, computer networks and distributed systems.
**E-mail:** safaei@ipm.ir



**Ahmad Khonsari** received the B.Sc. degree in electrical and computer engineering from Shahid-Beheshti University, Iran, in 1991, and M.Sc. degree in computer engineering from the Iran University of Science and Technology, Iran, in 1996 and Ph.D. degree in computer science from the University of Glasgow, UK, in 2003. He is currently an assistant professor in the Department of Electrical and Computer Engineering, University of Tehran, Iran and a researcher in School of Computer Science, Institute for Studies in Theoretical Physics and Mathematics (I.P.M.), Iran. His research interests are performance modeling/evaluation, mobile and ubiquitous computing, communication networks and distributed systems, and high performance computer architecture.
**E-mail**: ak@ipm.ir

**Mahmood Fathy** received B.Sc. degree in electronics from Iran University of Science and Technology in 1985, and M.S. in 1987 from Bradford University, UK, in computer architecture, and the Ph.D. degree in image processing computer architecture in 1991 from UMIST, UK. Since 1991, he has worked in the Computer Engineering Department as an Associate Professor. His research interests include mobile computing, sensor networks, real-time image processing in particular in traffic engineering and QoS in computer networks including video and image transmission over Internet.
**E-mail**: mahfathy@iust.ac.ir

**Mohamed Ould-Khaoua** received his B.Sc. degree from the University of Algiers, Algeria, in 1986, and the M.App.Sci. and Ph.D. degrees in Computer Science from the University of Glasgow, U.K., in 1990 and 1994 respectively. He is currently a lecturer in the Department of Computing Science at the University of Glasgow, U.K. His research focuses on applying theoretical results from stochastic processes and queuing theory to the quantitative study of hardware and software architectures. His current research interests are performance modeling/evaluation of parallel and distributed systems, parallel algorithms, ATM and multimedia networks. Dr. Ould-Khaoua is a member of the IEEE-CS and BCS.
**E-mail**: Mohamed@dcs.gla.ac.uk

* This research was in part supported by a grant from I.P.M. (No. CS1385-2-02).