Communication Prediction of Scouting Switching in Adaptively-Routed Torus Networks

F. Safaei[1,3], A. Khonsari [2,1], M. Fathy [3], N. Talebanfard [4], M. Ould-Khaoua [5,6]

[1] IPM School of Computer Science, Tehran, Iran
[2] Dept. of ECE, Univ. of Tehran, Tehran, Iran
[3] Dept. of Computer Eng., Iran Univ. of Science and Technology, Tehran, Iran
[4] Faculty of Mathematical Sciences, Shahid Beheshti Univ., Tehran, Iran
[5] Dept. of Electrical and Computer Eng., Sultan Qaboos Univ., Al-Khodh, Oman
[6] Dept. of Computing Science, Univ. of Glasgow, UK
{safaei, ak}@ipm.ir,{f_safaei, mahfathy}@iust.ac.ir, mohamed@dcs.gla.ac.uk

Abstract. The switching technique determines how messages are propagated from source to destination, and has a great impact on network performance. Traditional flow control mechanisms such as Wormhole Switching (WS) realize very good performance, but prone to deadlock in the vicinity of faults. While techniques such as adaptive routing can alleviate the problem, it cannot by itself solve the problem. This has motivated the development of different switching techniques. The Scouting Switching (SS) has been suggested as an efficient switching method for reconciling the conflicting demands of communication performance and fault-tolerance in computer networks. In this paper, we present a novel mathematical model to predict communication delay of SS coupled with virtual channels and fully adaptive routing in 2-D torus networks. We have carried out extensive simulation experiments, the results of which are used to validate the proposed analytical model.

1 Introduction

In large-scale parallel computers, 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 inter-processor communication depends largely on the network *topology*, the *switching* technique, and the *path selection* technique. The topology of a network defines how the nodes are interconnected and is generally modeled as a graph in which the vertices represent the nodes and the edges denote the channels. The torus has become a widely accepted communication network due to its desirable and powerful topological properties [1].

The switching technique determines how messages are propagated from the source to the destination, including the hardware protocols for transmitting data across a physical channel and for buffering data at a router. Modern interconnect networks feature the use of message pipelining coupled with virtual channels to improve network performance and insure deadlock freedom [1, 2]. Messages are broken into small units called flits or flow control digits. In Wormhole Switching (WS), data flits

immediately follow the header flit into the network. Network buffers and channels are committed as soon as they become available. This nature of WS leads to high network performance and low average message latencies. However, in the vicinity of faults, this behavior can lead to situations where the header can become blocked, no longer making progress, and hence cause the network to become deadlocked. While techniques such as adaptive routing can alleviate the problem, it cannot by itself solve the problem. This has motivated the development of different switching techniques [1, 2]. Scouting Switching (SS) is a hybrid message flow control mechanism that can be dynamically configured to provide specific trade-offs between fault-tolerance and performance. In SS, the first data flit is constrained to remain K (is referred as the *scouting distance* or *probe lead*) links behind the routing header. When $K = 0$, the flow control is equivalent to WS, while large values can ensure path set-up prior to data transmission (if a path exists). Every time a channel is successfully reserved by the header, it returns a *positive acknowledgement*. If the header message encounters blocking/faulty situation at an intermediate node, it is forced to backtrack to the preceding node and must send a *negative acknowledgement*. For performance reason, when $K = 0$, no acknowledgements are sent across the channels (realizing close to WS performance). By statically fixing the value of K, we fix the trade-off between network performance (overhead of positive and negative acknowledgements) and fault-tolerance (the ability of the header to backtrack and be routed around faults). Moreover, by dynamically modifying K, we can gain improved run-time trade-offs between fault-tolerance and performance [2].

Path selection technique is concerned by selecting a path from the source node to the destination node. The term “routing” usually refers to the algorithm that is used to select the routing path. Routing in interconnect networks which belongs to the network layer of the OSI model is used to map communications to hardware resources [3]. The routing algorithm is generally classified as being either *deterministic* or *adaptive* [1]. If the path between every pair of source and destination is fixed, the routing is called deterministic. For better system performance, it is preferable that the routing algorithm adapts itself to the traffic congestion in network. Adaptive routing overcomes the performance limitations of deterministic routing by enabling messages to explore all available paths. Furthermore, this strategy can also be efficiently implemented with SS because unlike in other switching methods (such as WS), message deadlock cannot arise since all seized channels are released when blocking occurs.

There have been a few studies that have considered the performance of switching methods, e.g., [2-5]. However, they have been conducted mainly through software simulation. Analytical models are cost-effective and versatile tools for evaluating system performance under different design alternatives. The significant advantage of the analytical approach over simulation is that the analytical models can be used to obtain performance results for large systems, which are infeasible by simulation due to the excessive computation demands on conventional computers. Although SS has been around for a number of years, and can greatly benefit from adaptive routing as it reduces blocking in the network, there has been hardly any attempt to provide an analytical model for SS in interconnect networks when fully adaptive routing along with virtual channels is used. In an effort to fill this gap, this paper proposes a new analytical model to capture the message latency of SS in adaptively-routed 2-D torus

networks coupled with virtual channels. The model uses an M/G/1 queuing system [6] to calculate the average waiting time that a message experiences at the source node before entering the network. The validity of the model is demonstrated by comparing analytical results with those obtained through flit-level simulation experiments.

The rest of the paper is structured as follows. In Section 2, some preliminaries are given. Section 3 describes the analytical model of SS for 2-D torus networks. Section 4 validates the proposed analytical model through simulation experiments. Finally, Section 5 concludes this study.

2 The torus topology

The topology we have chosen to investigate is the 2-D torus (*k*-ary 2-cube). This is a topology which has become increasingly popular among researchers and has been implemented in a number of existing machines [1]. A k-ary 2-cube is a direct network with $N = k^2$ nodes; k is called the radix. Links (channels) in the torus can be either uni- or bi-directional. In this paper, we will focus on 2-D torus with bi-directional links as they have been more popular in parallel systems. Each node can be identified by a 2-digit radix k address (a_1, a_2). Nodes, with address (a_1, a_2), (b_1, b_2) are connected if and only if $a_1 = (a_2 + 1) \bmod k$ or $b_1 = (b_2 + 1) \bmod k$. In order to allow processors to concentrate on computational tasks and permit the overlapping of communication with computation, a *router*, is used for handling message communication among processors, and is usually associated with each processor. Consequently, each node consists of a Processing Element (PE) and router. A node is connected to its neighboring nodes via the input and output channels. The *injection/ejection* channel is used by the processor to inject/eject messages to/from the network.

3 Analytic modeling

In this section, we describe an analytical model to assess the performance of SS in an adaptively-routed 2-D torus. The most important performance metric in our model is the average message latency.

3.1 Assumptions

The model is based on the following assumptions, which are accepted in the literature [7-11], and are listed below.

- Nodes (processors) generate traffic independently of each other, following a Poisson process with an average rate of l_g messages/ cycle.
- Message destination nodes are uniformly distributed across the network.
- The message length is M flits, each of which requires one cycle to cross from one router to the next.
- 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 (³ 1) virtual channels per physical channel are used. When there are more than one virtual channels available that bring a message closer to its destination, one is chosen at random.
- Messages are assumed to always follow shortest paths in the absence of faults. Further, when the header message encounters a busy link because all the required virtual channels are occupied, it is forced to backtrack to the preceding node and send a negative acknowledgement.

In the subsequent section, we derive the mathematical model that approximate the behavior of 2-D torus communication system using SS flow control mechanism.

3.2 Communication analysis

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 [7]

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

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

3.2.1 Calculation of the mean network latency

Under the 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 [8]

$$\bar{k} = k/4, \quad \bar{D} = 2\bar{k} \quad (2)$$

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, l_c, on a given channel is determined as follows. When the header that has made i hops is blocked, it sends a negative acknowledgement, backtracks to the preceding node, and makes a new attempt to set-up a connection. Given that the probability of blocking is Pb_i (determined later by Eq. (9)), the header visits, on average, $\bar{D} + \sum_{i=0}^{\bar{D}-1} iPb_i$ channels before it successfully establishes a connection. Since a node generates traffic at a rate of l_g messages/cycle and a router has 4 output network channels, we can write the rate of traffic on a channel, l_c, as

$$l_c = l_g\left(\bar{D} + \sum_{i=0}^{\bar{D}-1} iPb_i\right)\Big/4 \tag{3}$$

The average network latency, $\bar{S}$, consists of two parts: the average time to set-up a path, $\bar{C}$, and the actual message transmission time. Given that a message makes, on average, $\bar{D}$ hops to reach its destination, $\bar{S}$ can be written as

$$\bar{S} = \bar{C} + \bar{D} + M \tag{4}$$

where M and K indicate the message length and the scouting distance, respectively.

The average time to set-up a path, $\bar{C}$, is derived as follows. Consider the header message that is currently at a node being i hops away from the source. Let Pb_i be the probability that the header experiences blocking situation because all the required virtual channels are occupied at an intermediate node. Let $\bar{C}^t_{i,j}$ denote the expected duration time for the header to reach the destination from the current node that is i hops away from the source and has backtracked j times before reaching the current state. If the header succeeds in reserving the required virtual channel and advances to the next node, the residual expected duration time becomes $\bar{C}^t_{i+1,j}$. This case occurs with probability $1 - P^t_j Pb_i$; where P^t_j is the probability denoting that the header has backtracked j times. On the other hand, if the header message encounters blocking and backtracks to the preceding node, the residual expected duration time is $\bar{C}^t_{i-1,j+1}$. Given that the header requires one cycle to move from one node to the next, the above argument reveals that the expected duration $\bar{C}^t_{i,j}$ fulfils the following difference equations

$$\bar{C}^t_{i,j} = \begin{cases} (1 - Pb_0)(\bar{C}^t_{1,j} + 1) + Pb_0\bar{C}^t_{0,j} & i = 0 \\ (1 - P^t_j Pb_i)\bar{C}^t_{i+1,j} + P^t_j Pb_i\bar{C}^t_{i-1,j+1} + 1 & 0 < i < \bar{D} \\ 0 & i = \bar{D} \end{cases} \tag{5}$$

Let us proceed to compute the quantity of P^t_j which is denoted as the probability that the header message has backtracked j times over in a single source-destination path. We use two important facts that correspond to the header actions to cross the network. First, the gap between the header and the first data flits can grow up to $2K - 1$ hops while the header can make forward progress. Second, when there is no available output channel at a router to select, due to it being busy, the header can backtrack over the previous links up to K hops. To simplify our model, we assume that the number of consecutive channels that the header message will be forced to backtrack is limited to the scouting distance (i.e., $t \le K$). If $j = t$ no more backtracks could happen and thus $P^t_j = 0$. Assume that $j < t$. When the header at a node being i hops away from the source, there has occurred j backtracks and hence the header has to backtrack $t - j$ times along on its $\bar{D}$-hop path. Each backtrack could be made, on

average, along $\bar{D}$ places. We can conclude that the number of ways that $t - j$ backtracks can be distributed among $\bar{D}$ places is equal to $\binom{t-j+\bar{D}-1}{\bar{D}-1}$. We can also examine that, the number of ways that $t - j$ backtracks can be distributed among $\bar{D}$ places such that a specific place contains at least one backtrack is $\binom{t-j+\bar{D}-2}{\bar{D}-1}$. Thus, the probability P_j^t can be obtained as

$$P_j^t = \begin{cases} \dfrac{\binom{t-j+\bar{D}-2}{\bar{D}-1}}{\binom{t-j+\bar{D}-1}{\bar{D}-1}} & j < t \\ 0 & j = t \end{cases} \qquad (6)$$

Noting that the average path set-up time is at most $\bar{C}_{0,0}^t + 2K - 1$, solving the above equations yields the expected time, $\bar{C}_{0,0}^t$ needed for the header to reach the destination starting from the source. As a result, the average time to set-up a path, $\bar{C}$ with the SS flow control mechanism can be expressed as

$$\bar{C} = \frac{\sum_{t=0}^{K} (\bar{C}_{0,0}^t + 2K - 1)}{K + 1} \qquad (7)$$

Examining the Eqs. (3) and (5) reveals that the probability of blocking, Pb_i, is required to calculate $\bar{C}$. The probability that the header message is blocked at a given channel depends on its current network position. This is because the number of alternative paths that the header can take to progress is determined by the number of hops made by the header, and the way that these hops are distributed among the dimensions [11]. In order to calculate the probability Pb_i, that the header is blocked after making i hops, we need to enumerate the number of ways to distribute i hops among two dimensions. To do so, let Q_i^m be the probability that the header message has entirely crossed m $(0 \# m \quad 1)$ dimensions after making i hops. The details of calculation of Q_i^m have been developed elsewhere [7]. We recollect briefly here the main equations for the calculation of Q_i^m. The number of channels, and thus the number of virtual channels, that the header can select at a given hop depends on the number of dimensions still to be visited. When the header has made i $(0 \# i \quad \bar{D} - 1)$ hops, these hops can be a combination of (x,y) hops, with x and y being the number of hops achieved in the X and Y dimensions respectively, where $(x + y = i), (0 \# x,y \quad \bar{k})$. To determine the probability that the header message has crossed all the channels of one dimension, two cases need to be considered:

(1) When $(0 \le i \le \bar{k})$, the header has not yet crossed any dimension since it has to make $\bar{k}$ hops along each dimension. Therefore, the header can choose among virtual channels of both dimensions.

(2) When $(\bar{k} \le i \le \bar{D} - 1)$, the number of ways to distribute these hops along the two dimensions is $(\bar{D} - i + 1)$. In only two cases, $(x = \bar{k}, y = i - \bar{k})$ and $(x = i - \bar{k}, y = \bar{k})$, the header has crossed all channels of one dimension and thus, all the remaining hops have to be made in other dimension.

So, when the header has made i hops, the probability that there remains only one dimension to be crossed, P_{j_i}, can be written as

$$P_{j_i} = \begin{cases} 0 & 0 \le i \le \bar{k} \\ \dfrac{2}{\bar{D} - i + 1} & \bar{k} \le i \le \bar{D} - 1 \end{cases} \tag{8}$$

When the header arrives at the i-th hop channel, it has already made $(i - 1)$ hops and has entirely crossed, say, m $(0 \le m \le 1)$ dimensions. At its next hop, the header message can select any available $(2 - m)V$ virtual channels from the remaining $(2 - m)$ dimensions. Blocking occurs when all possible virtual channels at the remaining dimensions to be visited are occupied. If P_V (is given by Eq. (12)) denotes the probability that V virtual channels at a given physical channel are busy, the probability Pb_i, that the header is blocked is given by

$$Pb_i = \sum_{m=0}^{1} Q_i^m (P_V)^{2-m} \qquad 0 \le i \le \bar{D} - 1 \tag{9}$$

where Q_i^m is the probability that the header has entirely crossed m dimensions along on its i-hop path and is given by

$$\Theta_i^m = \begin{cases} 1 - P_{j_i} & m = 0 \\ P_{j_i} & m = 1 \end{cases}$$

(10)

3.2.2 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 [6]. 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 λ_g / V.

Using adaptive routing under the uniform traffic pattern results in the average service time seen by messages at all source nodes being the same, and equal to the average

network latency, i.e., $\bar{S}$ [7]. When the header message does not encounter any blocking during the path set-up stage, the minimum network latency seen by the message is $M + (2K - 1) + \bar{D}$. Applying the Pollaczek-Khinchine (P-K) mean value formula [6] with an approximated variance $(\bar{S} - M - (2K - 1) - \bar{D})^2$ [9] yields the average waiting time seen by a message at the source node as

$$\bar{W}_s = \frac{\lambda_g \left(\bar{S}^2 + (\bar{S} - M - (2K - 1) - \bar{D})^2 \right)}{2(V - \lambda_g \bar{S})} \tag{11}$$

3.2.3 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 [7, 10, 11]). In the steady state, the model yields the following probabilities [10].

$$P_v = \begin{cases} Q_0 = 1 & \\ Q_v = Q_{v-1} \lambda_c \bar{S} & (0 < v < V) \\ Q_v = \dfrac{Q_{v-1} \lambda_c}{1/\bar{S} - \lambda_c} & (v = V) \\ P_0 = \left(\sum_{l=0}^{V} Q_l \right)^{-1} & \\ P_v = P_{v-1} \lambda_c \bar{S} & (0 < v < V) \\ P_v = \dfrac{P_{v-1} \lambda_c}{1/\bar{S} - \lambda_c} & (v = V) \end{cases} \tag{12}$$

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 [10]

$$\bar{V} = \frac{\sum_{l=1}^{V} l^2 P_l}{\sum_{l=1}^{V} l P_l} \tag{13}$$

4 Simulation experiments

To further understand and evaluate the performance issues of the SS flow control mechanism, we have used a discrete-event simulator that operates at the flit level. The simulator uses the same assumptions as the analysis, and these assumptions were detailed in Section 3.1. Extensive validation experiments have been performed for several combinations of network sizes, message lengths, scouting distance, and virtual channels. However, for the sake of specific illustration, we have conducted our simulations for the following cases only:

- Network size $N = 64$, and 256 nodes.
- Number of virtual channels $V = 1$, and 6 per physical channel.
- Message length $M = 32$, and 64 flits.
- Scouting distance $K = 0, 1, 3$, and 6.

The average message latencies obtained from the simulation and analytical results plotted in Fig. 1. The *x*-axis in this figure represents the traffic rate injected by a given node in a cycle (i.e., l_g) while the *y*-axis shows the average message latency (in terms of flit cycles). The figure reveals that the simulation results closely match those predicted by the analytical model in the steady state regions. 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. The approximations which are made to compute the variance of service time distribution and those made for determining the traffic rate on network channels are the factors of the model inaccuracy. Since the independence 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.

5 Conclusions

This paper has presented a novel mathematical model to assess the relative performance merits of Scouting Switching (SS) in 2-D tori when adaptive routing and virtual channel flow control are used. The proposed analytical model is general and can easily be extended to other topologies. Results from flit-level simulation experiments have revealed that the latencies captured by the analytical model are in good agreement with those obtained through simulation. Our next step in this work is to extend our suggested modeling approach to consider the performance behavior of SS in the presence of failures.

References

1. Dally, W.J., Towles, B.: Principles and practices of interconnection networks, Morgan Kaufman Publishers (2004).
2. Dao, B. V., Duato, J., Yalamanchili, S.: Dynamically configurable message flow control for fault-tolerant routing, IEEE Transactions on Parallel and Distributed Systems, 10 (1) (1999) 7-22.
3. Theiss, I.: Modularity, Routing and Fault Tolerance in Interconnection Networks, PhD thesis, Faculty of Mathematics and Natural Sciences, University of Oslo, (2004).
4. Colajanni, M., Dell'Arte, A., Ciciani, B.: Performance evaluation of message passing strategies and routing policies in multicomputers, Simulation Practice and Theory, 6 (4) (1998) 369-385.
5. Gaughan, P. T., Yalamanchili, S.: A family of fault-tolerant routing protocols for direct multiprocessor networks, IEEE Transactions on Parallel and Distributed Systems 6(5) (1995) 482-497.
6. Kleinrock, L.: Queuing Systems, Vol. 1, John Wiley, New York (1975).

7. Ould-Khaoua, M.: A Performance model for Duato's adaptive routing algorithm in k-ary n-cubes, IEEE Trans. Computers, 48 (12) (1999) 1-8.
8. Agarwal, A.: Limits on interconnection network performance, IEEE Transactions on Parallel and Distributed Systems, 2 (4) (1991) 398-412.
9. Draper, J., Ghosh, J.: A comprehensive analytical model for wormhole routing in multicomputers systems, Journal of Parallel and Distributed Computing, 32 (1994) 202–214.
10. Dally, W. J.: Virtual channel flow control, IEEE Transactions on Parallel and Distributed Systems, 3 (2) (1992) 194-205.
11. Min, G., Ould-Khaoua, M.: A Comparative Study of Switching Methods in Multicomputer Networks, Journal of Supercomputing, 21 (2002) 227-238.

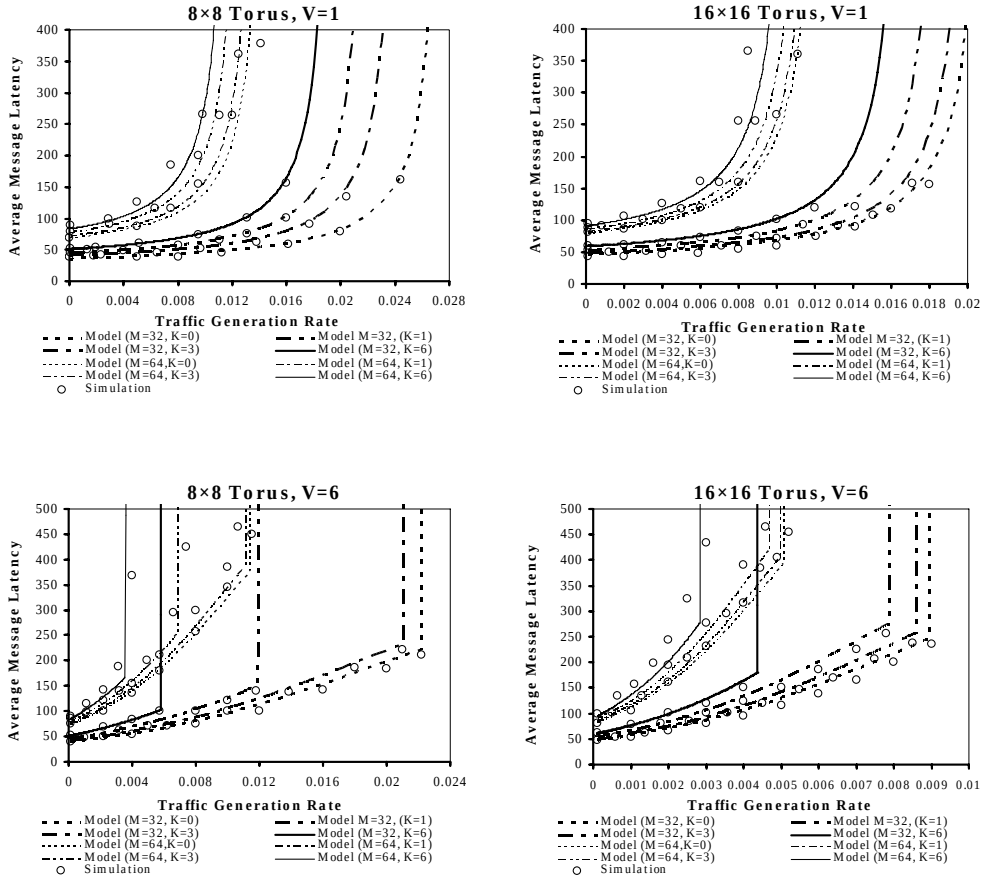

Fig. 1. Comparing the analytical model and flit-level simulation experiments in the 8×8 and 16×16 torus networks using Scouting Switching with message length M=32, 64 flits, Scouting distance K= 0,1,3,6, and V=1, 6 virtual channels per physical channel.