An Analytical Model of Pipelined Circuit Switching in Hypercubes in the Presence of Hot Spot Traffic

F. Safaei
School of Computer Science, Institute for Fundamental Studies
Email: safaei@ipm.ir

A. Khonsari
School of Computer Science, Institute for Fundamental Studies
Email: ak@ipm.ir

M. Fathy
Dept. of Computer Eng. Iran University of Science and Technology
Email: mahfathy@iust.ac.ir

M. Ould-Khaoua
Dept. of Computing Science, University of Glasgow
Email: mohamed@dcs.gla.ac.uk

Abstract

Several recent studies have revealed that PCS can provide superior performance characteristics over Wormhole Switching under uniform traffic. Analytical model of PCS for common networks (e.g., hypercube) under uniform traffic pattern have recently been reported in the literature. In this paper we propose an analytical model of PCS in the hypercube network augmented with virtual channel in the presence of hot spot traffic. The model has a good agreement with simulation experiments.

1. Introduction

In the modern world, there is an apparently insatiable demand for ever-greater processing power, particularly in science and engineering. The hypercube has been one of the most popular networks for multicomputers due to its desirable and powerful topological properties. The iPSC/2 [1], N-CUBE 2 [2] are examples of practical systems that are based on the hypercube.

The switching method determines the way messages visit intermediate routers. In Wormhole Switching (also widely known as wormhole routing [3]), a message is divided into *flits* (each of a few bytes) for transmission and flow control. The *header flit* governs the path and the remaining data flits follow it in a pipelined fashion. If the header is blocked in Wormhole Switching, the data flits stop advancing and remain spread across the channels that they have already acquired.

Gaughan and Yalamanchili [4] have proposed PCS that combines aspects of Circuit Switching (CS) and Wormhole Switching. PCS sets up a path before starting data transmission as in CS. Basically, PCS differs from CS in that virtual channel instead of physical channels form paths. In PCS, data flits do not immediately follow the header flits into the network as in Wormhole Switching. Consequently, increased flexibility is available in routing the header flit. For example, rather than blocking on a faulty output channel at an intermediate router, the header may backtrack to preceding router and release the previously reserved channel. A new output channel may now be attempted at the preceding router in finding an alternative path to the destination. When the header finally reaches the destination node, an *acknowledgement flit* is transmitted back to the source node. Now data flits can be pipelined over the path just as in Wormhole Switching. This approach is flexible in that headers can perform a backtracking search of the network, reserving and releasing virtual channels in an attempt to establish a fault-free path to the destination.

PCS has been proposed with the original aim of reconciling the conflicting demands of communication performance and fault tolerance in multicomputer networks [5], and has been used in recent systems including the Ariadne [6], METRO [7], and MMR Router [8]. A common advantage of PCS and CS is their ability to provide messages with a guaranteed latency once a connection has been established. Such a feature makes them attractive for transmitting multimedia data that is very sensitive to the jitter latency (i.e., variation in latency) [8]. Indeed, several researchers have recently proposed router architectures that incorporate PCS to efficiently support multimedia applications.

Analytical model of PCS under uniform traffic has been reported in the literature [9]. A number of studies [3, 10] have revealed that the performance advantages of interconnection networks are more noticeable when traffic is non-uniform. A message traffic model that has attracted much attention is the *hot spot* model studied in [10] where all (or a large number of) nodes attempt to direct a fraction of their generated messages to a single destination node with relatively high probability. This may lead to extreme network congestion resulting in serious performance degradation. Hotspot traffic behaviour may be exhibited in many applications such

Proceedings of the 2005 International Conference on Parallel Processing Workshops (ICPPW'05)
1530-2016/05 $20.00 © 2005 IEEE

as cache coherency protocols, synchronization and many operating system functions [11]. It can be produced directly by certain collective communication operations including *gathering* and *barrier synchronization* [12]. This paper proposes an analytical model of PCS in hypercube network under hot spot traffic pattern. The validation of the model is illustrated by comparing analytical results to simulation experiments.

The rest of the paper is organized as follows. Section 2 describes the node structure of the network. Section 3 presents our analytical model. Section 4 validates the analytical model through simulation results. Section 5 concludes the paper.

2. Preliminaries

An n-dimensional hypercube (also called n-cube) consists of $N = 2^n$ nodes connected by $n \times 2^n$ links (or channels), offers many attractive features [13]. In an n-cube, each node can communicate directly with its n neighbours. For non-neighbouring nodes, they have to communicate through some intermediate nodes. In a hypercube, each node can be addressed by a unique n-bit binary code from 0 to $2^n - 1$. For example, $b_n b_{n-1} \cdots b_2 b_1$ is an address for a node q_x $(0 \le x \le 2^n - 1)$. The *i-th* rightmost bit b_i is referred to as *dimension i*. Two nodes q_x and q_y are connected by a direct link if and only if their addresses differ in exactly one bit.

Each node consists of a processing element (PE) and router. The PE contains a processor and some local memory. Also, the router has *(n+1)* input and *(n+1)* output channels. A node is connected to its neighbouring nodes via *n* inputs and *n* output channels. The remaining channels are used by the PE to inject/eject messages to/from the network, respectively. The router contains buffers for inputting virtual channels. The input and output channels are connected through a *(n+1)V*-way crossbar switch, with *V* being the number of virtual channels which can simultaneously connect multiple input to multiple output channels in the presence of channel connection.

3. The analytical model for hypercube with PCS

This section describes the assumptions and notations used in the derivation of the analytical model.

3.1 Assumptions

The model uses the following assumptions, which are commonly used in the past literature [9,10, 14, 15] and used in the construction of the analytical model.

- Nodes generate traffic independently of each other, which follows a Poisson process with a mean arrival rate of λ_g messages per node per cycle.
- There is only a single hot spot node in the network. This restriction is to keep the notation used for description the model at a manageable level. However, this model can easily be extended to deal with the case of multiple hot spots. The traffic model is proposed by Pfister and Norton [16] is used to generate hot spot traffic pattern. In this model, each generated message has a finite probability θ of being directed to the hot spot node and probability $(1-\theta)$ of being directed to other network nodes. We usually refer to these types of messages as *hot spot* and *regular*, respectively.
- Message destination nodes are not uniformly distributed across the network nodes. A regular message crosses *i* dimension $(1 \le i \le n)$ to reach its destination with probability P_{r_i}. Different traffic patterns yield different values of P_{r_i}. For example, under the uniform traffic pattern $P_{r_i} = \binom{n}{i} / (N-1)$.
- Message length is *M* flits, each of which requires one cycle to cross from one node to the next.
- The queue in the source node has infinite capacity. Furthermore, messages at the destination node are transferred to the local processing element as soon as they arrive at their destinations.
- Each physical channel has *V* virtual channels $(V \ge 1)$. When there is more than one virtual channel available that bring a message closer to its destination, one is chosen at random. So, there is no distinction between the virtual channels when computing the different virtual channels occupancy probabilities.
- In the event of message blocking, the message header releases the last reserved virtual channel and backtracks to the preceding node, then searches for an alternative virtual channel to advance towards its destination [9].

3.2. Derivation of the model

The average message latency composed of the average network latency, $\overline{T}$, that is the average time to pass the network, and the average waiting time seen by messages at the source node, $\overline{W}$. However, to model

the effects of virtual channel multiplexing, the average message latency has to be scaled by a factor, $\overline{V}$, representing the average degree of virtual channel multiplexing. Therefore, the average message latency can be written as [17]:

$$\text{Average message latency} = (\overline{T} + \overline{W})\overline{V} \quad (1)$$

Since the topology of hypercube is symmetric, averaging the network latencies seen by the messages generated by only one node and destined to all the other nodes in the network yields the average network latency [10]. In the following section, we calculate the quantities $\overline{T}$, $\overline{W}$ and $\overline{V}$.

3.2.1. Calculation of the average network latency

Let $\overline{T}$ denotes the average network latency seen by a message i.e., a message that needs to cross from source node to destination node. Note that, the regular and hot spot messages see different network latencies as they cross-different number of hops to reach their destination. Let $\overline{T_r}$ and $\overline{T_h}$ denote the average network latency for regular and hot spot messages, respectively. The average network latency taking into account both types of messages is given by:

$$\overline{T} = (1-\theta)\overline{T_r} + \theta\overline{T_h} \quad (2)$$

In PCS, the network latency for a regular (or a hot spot) message consists of two parts: one the time to setup a path and other, the time to transmit a message from source to destination. Therefore, the network latency of an *i*-hop regular message can be written as:

$$\overline{T}_{r_i} = M + i + \overline{d}_{r_i} \quad (3)$$

In the above equation, M is the message length and $\overline{d}_{r_i}$ is the average time needed to setup a path for a regular *i*-hop message header. The probability that a newly generated regular message makes *i* hops to reach its destination is P_{r_i} $(1 \le i \le n)$. Averaging over all the possible *i* hop made by a regular message results the average network latency for regular messages:

$$\overline{T_r} = \sum_{i=1}^{n} P_{r_i}\overline{T}_{r_i} \quad (4)$$

Similarly, network latency, $\overline{T}_{h_j}$, seen by a hot spot message that is *j* hops away $(1 \le j \le n)$ from the hot spot node is given by:

$$\overline{T}_{h_j} = M + j + \overline{d}_{h_j} \quad (5)$$

Where $\overline{d}_{h_j}$ is the average time to setup a path for a hot spot message that is *j* hops away $(1 \le j \le n)$ from the hot spot node. The probability that a newly generated hot spot message which is a *j*-hop message is given by:

$$P_{h_j} = \binom{n}{j} \Big/ (N-1) \quad (6)$$

Averaging over all the possible hops made by a hot spot message, we can write the average network latency for hot-spot messages as:

$$\overline{T_h} = \sum_{j=1}^{n} P_{h_j}\overline{T}_{h_j} \quad (7)$$

3.2.2 Calculation of the average time to setup a reserved path

When header flit finds all feasible virtual channels at a given intermediate node busy, it releases the last reserved virtual channel and backtracks to the previous node, then resumes searching for an available virtual channel from the node. We model the header behaviour from the source node to the destination node as a Random Walk problem [18], which is associated with Markov chain that illustrated by Figure 1.

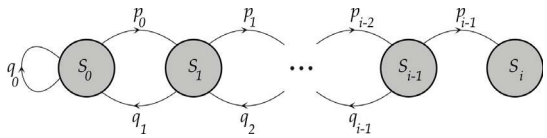

Figure 1: The Markov chain diagram for calculating the average path setup time

As illustrated in Figure 1, each state represents the current location of the message header along its network path. States S_0 and S_i correspond to the case where header flit is in the source and destination nodes, respectively. Therefore, state S_j $(1 \le j \le i-1)$ denotes the case where the header flit is at intermediate node that is *j* hop away from the source node; or *i-j+1* hops remain to reach to the hot spot node. Any transition of state S_j to S_{j+1} implies that the header flit succeeds in acquiring virtual channel and brings it one hope closer to its destination. Also, each transition from state S_j to state S_{j-1} means that the header flit has encountered blocking situation at the node corresponding to state S_j and it has to backtrack to

the previous node. We assume that the probability when the header flit is blocked in the intermediate node is q_j. Hence, p_j (equals to $1-q_j$) denotes the transition probability of advancing across the reserved path. State S_i is a final state and known as *"absorbing state"* since once the header flit reaches its destination, a full path is reserved. State S_0 is a *"partially reflecting"* state, with the reflecting probability q_0 because once the header backtracks to its source node due to blocking this implies that it has not succeeded in establishing a path and has to make a new attempt. We want to calculate the expected duration time interval that header flit starting from state S_0 to reach the absorbing state in two situations both normal and hot spot messages. This time corresponds to the average time that the header flit needs to reserve a desired path from the source node to the destination node.

Calculation of the average time to setup a reserved path for messages

Let $\bar{d}_j$ be the average time interval to reach the absorbing state originating from state S_j. $\bar{d}_j$ is always finite [18], and $\bar{d}_{j+1}$ denotes the header flit at state S_j succeeds in acquiring a virtual channel and it can proceeds to state S_{j+1}. On the other hand, when the header flit encounters situation of blocking, it backtracks to preceding node corresponding to state S_{j-1} and the residual average time would be $\bar{d}_{j-1}$. It is assumed that the header needs one cycle to move from one node to another. The above argument reveals that the average time, $\bar{d}_j$, satisfies the following equations:

$$\begin{cases} \bar{d}_j = p_j \bar{d}_{j+1} + q_j \bar{d}_{j-1} + 1 & (1 \le j \le i-1) \\ \bar{d}_0 = p_0 (d_1 + 1) + q_0 \bar{d}_0 \\ \bar{d}_i = 0 \end{cases} \quad (8)$$

Once the header reaches its destination node, an acknowledgment flit is transmitted back to the source node through the reserved path. Thus, the average time to setup a path for an *i*-hop regular message can be written as:

$$\overline{d_r}_i = \bar{d}_0 + i \quad (9)$$

Similarly, the time is needed to setup a reserved path for hot spot message that is *j* hops away $(1 \le j \le n)$ from the hot spot node is given by:

$$\bar{d}_{hj} = \bar{d}_0 + j \quad (10)$$

Finally, by averaging all possible values of $\bar{d}_r$ and $\bar{d}_h$ yields the overall average time to set up a path, $\bar{d}$, as the following equation:

$$\bar{d} = (1-\theta) \cdot \sum_{i=1}^{n} P_{r_i} . \bar{d}_{r_i} + \theta \cdot \sum_{j=1}^{n} P_{h_j} . \bar{d}_{h_j} \quad (11)$$

Note that in equations 9 and 10 the terms *i* and *j* both accounts for *i* and *j* cycles that are required to send the acknowledgement flit back to the source node.

Calculation of the probability of message header blocking

A regular header message is blocked at a given node when all the virtual channels of the remaining dimensions still to be visited are busy. When blocking occurs, the header has not to wait for acquiring a virtual channel and immediately backtracks to its previous node. The probability of blocking for message header depends on the number of output channels, and thus on the number of virtual channels that a message header can use at its next hop. Therefore, the waiting time in the event of blocking equals to zero either for hot spot or regular case.

Consider a regular message header that has to cross *i* hops to reach its destination, which is different from the hot spot node. Suppose that the message header has reached the k^{th}-hop channel along its network path. This channel can be between *1* and *n* hops away from the hot spot node. Let $p_{i,j,k}$ denote the probability of blocking when the k^{th}-hop channel is *j* hops away from the hot spot node. The reason behind using two subscripts, *k* and *j*, for $p_{r,j,k}$ is that the probability of message header is blocked at a given node depends on its current network position and its location with respect to the hot spot node. A message header crosses exactly one dimension after making each hope in the hypercube network. Now we compute the probability $p_{i,j,k}$. A message is blocked when all the possible virtual channels it can use, are busy. Hence, the probability that blocking occurs can be written as:

$$p_{i,j,k} = (p_{L_j})^{i-k+1} \quad (12)$$

Where p_{L_j} is the probability that all virtual channels are busy. Similarly, the probability of blocking of the hot spot header message that is *j*-hop away the hot spot node can be calculated as:

$$p_{h,j} = (p_{L_j})^{j} \quad (13)$$

Calculation of the rate of regular messages on a given channel

Given that a newly generated regular message makes *i*-hops to reach its destination with probability P_{r_i} $(1 \le i \le n)$, the average number of hops that a regular message makes across the network, $\overline{D_r}$, is given by [14]:

$$\overline{D_r} = \sum_{i=1}^{n} i.P_{r_i} = \frac{nN}{2(N-1)} \quad (14)$$

Fully adaptive routing allows a regular message to use any channel that brings it closer to its destination, resulting in an evenly distributed regular traffic rate on all network channels. A router in the hypercube has *n* output channels and a node generates, on average, $(1-\theta)\lambda_g$ regular messages in a cycle. Since each regular message travels, on average, $\overline{D_r}$ hops to cross the network, the rate of regular messages received by each channel, λ_r, can be written as:

$$\lambda_r = \frac{(1-\theta)\lambda_g \overline{D_r}}{n} = \frac{(1-\theta)\lambda_g \overline{d}}{4n} \quad (15)$$

Calculation of the rate of hot spot messages on a channel j hops away from the hot spot node

Hot spot traffic is not uniformly distributed across the network channels. Consider a channel that is *j* hops away from the hot spot node $(1 \le j \le n)$. The probability that a hot spot message has used during its network journey this channel can be derived as follows [10]. Consider the set *J* of all the channels located *j* hops away from the hot spot node. The number of source nodes for which an element of *J* can act as intermediate channel to reach the hot-spot node is $N - \sum_{k=0}^{j-1} \binom{n}{k}$. Since there are $(n-j+1)\binom{n}{j-1}$ such intermediate channels, the probability, P_{c_j}, that a hot spot message has used during its network trace a channel which is *j* hops away from hot spot is given by:

$$P_{c_j} = \left(N - \sum_{k=0}^{j-1} \binom{n}{k} \right) \Big/ \left((n-j+1)\binom{n}{j-1} N \right) \quad (16)$$

Given that each of the *N* nodes generates, on average, $\theta\lambda_g$ hot spot messages in a cycle, the rate of hot spot traffic, λ_{h_j}, received by a channel located *j* hops away from hot spot node is simply given by:

$$\lambda_{h_j} = \theta N \lambda_g P_{c_j} = \theta \lambda_g \sum_{k=j}^{n} \binom{n}{k} \Big/ \left((n-j+1)\binom{n}{j-1} \right) \quad (17)$$

To determine the total input traffic rate for the network, we calculate the traffic rate on the channel is located *j* hops from the hot spot node and add this value to the rate of messages arriving at the channel consists of the traffic rate of regular messages. Therefore, the overall traffic rate is computed as:

$$\lambda_j = \lambda_r + \lambda_{h_j} \quad (18)$$

Since adaptive routing distributes regular traffic evenly across the network channels, and due to the symmetry of the hypercube topology, the mean service time seen by a regular message is the same across all channels. When a regular or hot spot message reaches a channel that is *j* hops away form the hot spot node, the mean service time at the channel, considering both regular and hot spot message with their appropriate weights, can be written as:

$$\overline{T}_j = \frac{\lambda_c}{\lambda_j}\overline{T_r} + \frac{\lambda_{h_j}}{\lambda_j}\overline{T}_{h_j} \quad (19)$$

3.3. Calculation of the mean waiting time at the source

The mean waiting time in the source node is the average time that a message encounters in the source node before entering into the network. A physical channel is treated as an M/G/1 queue with a mean time waiting of [19]:

$$\overline{W} = \rho\overline{T}(1 + C_{\overline{T}}^2)/2(1-\rho) \quad (20)$$

$$\rho = \lambda\overline{T} \quad (21)$$

$$C_{\overline{T}}^2 = \sigma_{\overline{T}}^2 / \overline{T}^2 \quad (22)$$

Where λ is the traffic rate on the network, $\overline{T}$, is the mean service time, and $\sigma_{\overline{T}}^2$ is the variance of the service distribution. A message in the source node can

enter the network through any of the V virtual channels. A regular message originating from a given source node that is j hops away from the hot spot node sees a network latency of $\overline{T_r}$ (given by equation 4), whereas a hot spot message sees a latency of T_h (given by equation 7) to reach the hot spot node. So, the average latency taking into accounts both regular and hot spot messages is simply calculated by $(1-\theta)\overline{T_r} + \theta\overline{T_h}$. As we mentioned above, modelling the local queue in the source node that is j hops away from the hot spot node as an M/G/1 queue, and the mean arrival rate on each virtual channel is λ_g / V and service time $\overline{T}$ with an approximated variance $\left(\overline{T} - M - 3\overline{D} + 1\right)^2$ yields the mean waiting time as:

$$\overline{W} = \frac{\overline{T}^2 (\lambda_g / V)[1 + (\overline{T} - M - 3\overline{D} + 1)^2 / \overline{T}^2]}{2[1 - \overline{T}(\lambda_g / V)]} \qquad (23)$$

3.3.1. Calculation of the average degree of virtual channels multiplexing

The probability, Pb_{v_j}, that v virtual channels are busy at the physical channel that is j hops away from the hot spot node, can be determined using a Markovian model [15]. In the steady state, the model yields the following probability:

$$Pb_{v_j} = \begin{cases} (1 - \lambda_j \overline{T_j})(\lambda_j \overline{T_j})^v, & 0 \le v < V \\ (\lambda_j \overline{T_j})^v, & v = V \end{cases} \qquad (24)$$

Hence, the average degree of multiplexing of virtual channels located j hops away from hot spot can be approximated by [15]:

$$\overline{V_j} = \sum_{v=1}^{V} v^2 P_{v_j} \Big/ \sum_{v=1}^{V} v P_{v_j} \qquad (25)$$

After averaging over all the values of j $(1 \le j \le n)$ the average degree of multiplexing of virtual channels, that takes place at a given physical channel, can be given by:

$$\overline{V} = \sum_{j=1}^{n} P_{h_j} \overline{V}_j \qquad (26)$$

Examining the above equations of the analytical model reveals that it is very difficult to give close-form solutions to the various variables of the model. Therefore, these equations are solved iteratively.

4. Validation of the model

The analytical model has been validated through a discrete-event simulator that mimics the behaviour of PCS at the flit level in the hypercubes. The mean message latency is defined as the mean amount of time from the generation of a message until the last data flit reaches the local PE at the destination node. The other measures include the mean network latency, the time taken to cross the network, the mean waiting time at the source node, and the time spent at the local queue before crossing the first network channel. Numerous validation experiments have been performed for several combinations of network sizes, message lengths, and number of virtual channels to validate the model. However, for the sake of specific illustration, validation results are presented for the following cases only:

- Network size $N = 2^8$ nodes.
- Message length M=32 and 64 flits.
- Number of virtual channels V=3, 6 and 10.
- Fraction of hot spot traffic is $\theta = 0.05, 0.2$. The hot spot node is assumed to be the node with address N-1.

Figure 2 shows the results of the overall message latency for hot spot case with traffic fractions $\theta = 0.05, 0.2$ and V=3, 6 and 10 virtual channels. The horizontal axis presents the traffic rate (λ_g) and the vertical axis illustrates the average latency in crossing from the source to the destination node. This figure reveals that the analytical model predicts the average message latency with a good degree of accuracy under low, moderate traffic and when network enters to heavy traffic region. However, some discrepancies are noticeable between results provided by the analytical model and simulation under heavy traffic and approach the saturation point. This is due to approximations that have been made in the derivations of the equations to ease the implementation of the model. However, it can be concluded that the model produces accurate results in the steady state regions, which are the regions of interests in most evaluation network studies.

5. Conclusions

Many studies, based on simulation experiments, have revealed the performance advantages of Pipelined Circuit Switching (PCS) over the Wormhole Switching under uniform traffic. However, the uniform traffic assumption is not always justifiable in practice as there are many parallel applications that exhibit non-uniform behaviour. In this paper we presented an analytical

model to compute message latency of hypercubes with PCS under hot spot traffic. Our analytical results show that this model has a good agreement with simulation experiments. Our next objective is to extend our modelling approach to consider other fault-tolerant routing algorithms and network topologies.

References

[1] S.F. Nugent, "The iPSC/2 direct-connect communication technology", *Proc. Conf. on Hypercube Concurrent Computers & Applications*, Vol. 1, 1988, pp. 51-60.

[2] N-Cube Company, *NCUBE-2 Processor Manual*, 1990.

[3] L. M. Ni, P. K. McKinley, "A survey of wormhole routing techniques in direct networks", *IEEE Computer*, Vol. 26, 1993, pp. 62-76.

[4] P.T. Gaughan and S. Yalamanchili, "A family of fault-tolerant routing protocols for direct multiprocessor networks", *IEEE Trans. Parallel & Distributed Systems,* Vol. 6, No.5, 1995, pp. 482-497.

[5] J. Duato, S. Yalamanchili, L.M. Ni, *Interconnection networks: An engineering approach*, IEEE Computer Society Press, Los Alamitos, CA, 1997.

[6] J.D. Allen, P.T. Gaughan, D.E. Schimmel, S. Yalamanchili, "Ariadne-- An adaptive router for fault-tolerant multicomputers"*, Proc. ACM/IEEE 21st Int. Symp. Computer Architecture (ISCA-21)*, ACM Press, 1994, pp. 278–288.

[7] A. DeHon, F. Chong, M. Becker, E. Egozy, H. Minsky, S. Peretz, T.F. Knight, "METRO: A router architecture for high-performance, short-haul routing networks"*, Proc. ACM/IEEE 21st Int. Symp. Computer Architecture (ISCA-21)*, ACM Press, 1994, pp. 266-277.

[8] J. Duato, S. Yalamanchili, M.B. Caminero, D. Love, F. Quiles, "MMR: A highperformance multimedia router: architecture and design trade-offs"*, Proc. 5th Int. Symp. High Performance Computer Architecture (HPCA'99)*, IEEE Computer Society Press, 1999, pp. 300-309.

[9] G. Min, *Performance Modelling and Analysis of Multicomputer Interconnection Networks*, PhD Thesis, Computing Science Department, Glasgow University, 2003.

[10] M. Ould-Khaoua, H. Sarbazi-Azad, "An analytical model of adaptive wormhole routing in hypercubes in the presence of hot spot traffic", *IEEE Trans. On Parallel and Distributed Systems*, Vol. 12, No. 3, March 2001, pp. 283-292.

[11] W. Hsu, *Performance issues in wire-limited hierarchical networks*, PhD Thesis, University of Illinois-Urbana Champaign, 1992.

[12] P.K. Mckinley, D.F. Robinson, "Collective communication in wormhole-routed massively parallel computers", *IEEE Computer*, Dec. 1995, pp. 39-50.

[13] Y. Saad, M. Schultz, "Topological Properties of Hypercubes", *IEEE Trans. On Computers,* Vol. 37, No. 7, July 1988, pp.867-872.

[14] S. Abraham, K. Padmanabhan, "Performance of the direct binary *n*-cube networks for multiprocessors", *IEEE Trans. Computers,* Vol. 37, No. 7, 1989, pp. 1000-1011.

[15] W.J. Dally, "Virtual channel flow control", *IEEE Trans. Parallel & Distributed Systems* , Vol. 3, No. 2, 1992, pp. 194-205.

[16] G.J. Pfister, V.A. Norton, "Hot-spot contention and combining in multistage interconnection networks", *IEEE Trans. Computers*, Vol. 34, No. 10, 1985, pp. 943-948.

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

[18] W. Feller, *An introduction to probability theory and its applications*, Vol. 1, John Wiley & Sons, New York, 1967.

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

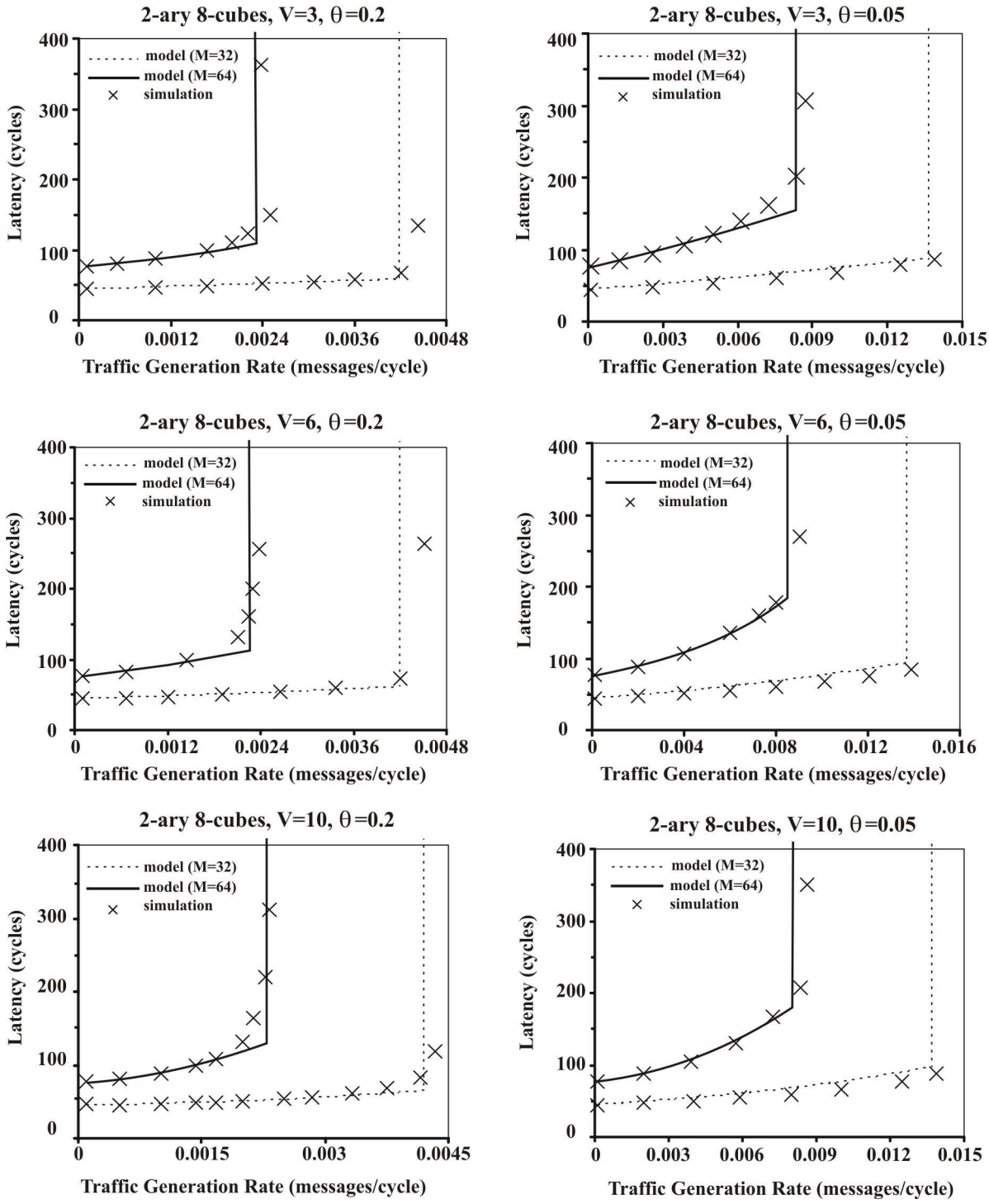

Figure 2: Average message latency calculated by the model versus simulation in 2-ary 8-cubes for different number of virtual channels V=3, 6 and 10, message lengths M=32 and 64 flits, and different hot spot fractions $\theta = 0.05, 0.2$.