Performance Analysis of Wormhole Routing in Multicomputer Interconnection Networks

A Thesis Submitted

by

Hamid Sarbazi-Azad

for

The Degree of Doctor of Philosophy

to

The Faculty of Science
University of Glasgow

To my parents, my wife and my son

for their love, support and encouragement.

Abstract

Perhaps the most critical component in determining the ultimate performance potential of a multicomputer is its interconnection network, the hardware fabric supporting communication among individual processors. The message latency and throughput of such a network are affected by many factors of which topology, switching method, routing algorithm and traffic load are the most significant. In this context, the present study focuses on a performance analysis of k-ary n-cube networks employing wormhole switching, virtual channels and adaptive routing, a scenario of especial interest to current research.

This project aims to build upon earlier work in two main ways: constructing new analytical models for k-ary n-cubes, and comparing the performance merits of cubes of different dimensionality. To this end, some important topological properties of k-ary n-cubes are explored initially; in particular, expressions are derived to calculate the number of nodes at/within a given distance from a chosen centre. These results are important in their own right but their primary significance here is to assist in the construction of new and more realistic analytical models of wormhole-routed k-ary n-cubes.

An accurate analytical model for wormhole-routed k-ary n-cubes with adaptive routing and uniform traffic is then developed, incorporating the use of virtual channels and the

effect of locality in the traffic pattern. New models are constructed for wormhole k-ary n-cubes, with the ability to simulate behaviour under adaptive routing and non-uniform communication workloads, such as hotspot traffic, matrix-transpose and digit-reversal permutation patterns. The models are equally applicable to unidirectional and bidirectional k-ary n-cubes and are significantly more realistic than any in use up to now. With this level of accuracy, the effect of each important network parameter on the overall network performance can be investigated in a more comprehensive manner than before.

Finally, k-ary n-cubes of different dimensionality are compared using the new models. The comparison takes account of various traffic patterns and implementation costs, using both pin-out and bisection bandwidth as metrics. Networks with both normal and pipelined channels are considered. While previous similar studies have only taken account of network channel costs, our model incorporates router costs as well thus generating more realistic results. In fact the results of this work differ markedly from those yielded by earlier studies which assumed deterministic routing and uniform traffic, illustrating the importance of using accurate models to conduct such analyses.

Acknowledgements

First of all, I must thank the almighty Allah (SWT) who has again helped me to pass another important passage in my life.

I would like to thank my supervisors, Dr Lewis Mackenzie for his support and guidance during the course of my research, and Dr Mohamed Ould-Khaoua for his continuous effort and the time he spent for numerous discussions on the issues arisen during the course of this study. I also thank all my colleagues, especially my officemates for their comments and help, and the staff of the Department of Computing Science for their kind and friendly support. I am indebted to the Iranian government for the financial support I have been provided with for staying and studying here in Glasgow.

Finally, I must thank my wife and our sweet son Mostafa, for putting up with many husbandless and fatherless evenings and weekends when I was doing my research. Without their tolerance and continuous support and encouragement this work would not be finished.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the modern world, there is an apparently insatiable demand for ever-greater processing power, particularly in science and engineering. Although designers have had considerable success in increasing the performance of individual processors using advanced micro-architectures, this has limitations and large-scale parallel processor systems have become increasingly popular for high-end applications [90, 99]. Indeed, arguably, such machines are possibly the only feasible way of achieving the enormous computational power [27] required in these areas.

Parallel systems may be based on either a *shared-memory* or *distributed-memory* model. In shared-memory architectures, known as *multiprocessors,* all processors may access a shared memory while in those based on the distributed-memory model, known as *multicomputers*, processors communicate by means of interchanging messages. The latter, in particular, have experienced rapid development during the last decade [19] because of their superior scalability. Such systems are organized as an ensemble of nodes, each having its own processor, local memory and other supporting devices [129], comprising a

processing element (PE) and a router or switching element (SE) which communicates with other nodes via an *interconnection network.*

1.1 Interconnection networks

An interconnection network is a crucial component of a multicomputer because the overall system performance is very sensitive to network latency and throughput [129, 137]. It may employ a variety of topologies that can be classified into two broad categories: *indirect* and *direct* [129]. In indirect networks, the nodes are connected to other nodes (or memory banks in a shared-memory architecture) through multiple intermediate stages of switching elements (SE). Many experimental and commercial parallel machines have employed indirect interconnection networks [59], such as Hitachi SR2201 [75, 182], Cedar [105], Cray X/Y-MP, DEC GIGA switch and Cenju-3, IBM RP3 [148] and SP2 [22], Thinking Machine CM-5 [111] and Meiko CS-2. Examples of indirect networks include crossbar [75], bus [68] and multistage interconnection networks (MINs) [105]. Figure 1(a) illustrates a Butterfly MIN constructed from 2x2 switches. In direct networks (also called *point-to-point* networks) each node has a point-to-point or direct connection to some of the other nodes (known as its *neighbours*) allowing for direct communication between processors. Direct interconnection networks have been widely employed by recent machines [59]. Figure 1(b) shows a 4x4 mesh and associated node structure.

In a MIN multicomputer with N processing nodes, there are typically $O(N \log N)$ switching elements while in a direct network there are N such elements. From the scalability point of view, direct networks are preferred. Moreover, direct networks can exploit locality in traffic more effectively. Consequently, most recent multicomputers employ these networks, including the Intel iPSC [16, 91, 140], Intel Delta [93], Intel Paragon [92], Cosmic Cube [170], nCUBE [131-133], MIT Alewife [7] and J-machine [138, 139], iWarp [147], Stanford DASH [112], Stanford FLASH [107], Cray T3D [98],

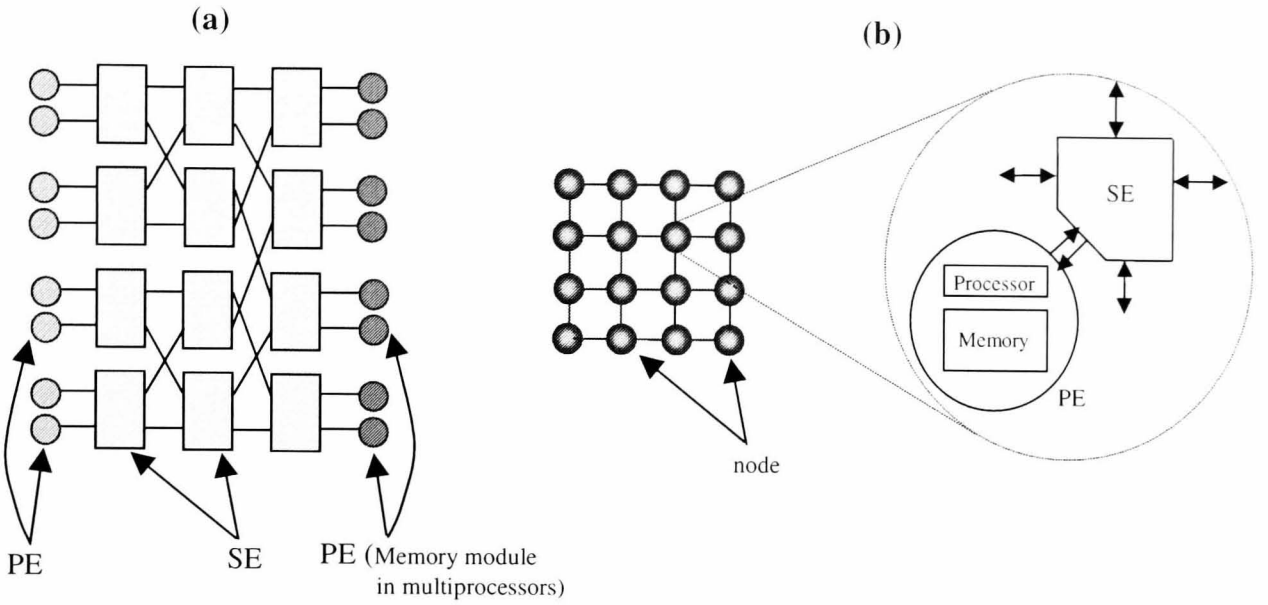


Figure 1.1: Examples of indirect and direct networks, (a) A Butterfly MIN constructed from 2x2 switches, (b) A 4x4 mesh.

Cray T3E [13, 45], and SGI Origin [108]. In this study we focus on direct interconnection networks. From this point by "interconnection network" we mean "direct interconnection network" unless otherwise mentioned.

In addition to the technology in which the hardware is implemented (considered in Section 1.2) several key factors influence network performance: *topology, switching method, routing algorithm* and the *traffic pattern* generated by the application program being executed. These are now considered in turn.

1.1.1 Network topology

Network topology defines the way nodes are connected and can be described using an interconnection graph. The vertices of this graph are the nodes and the edges are the physical channels that connect the nodes [110]. The network *diameter* is the maximum

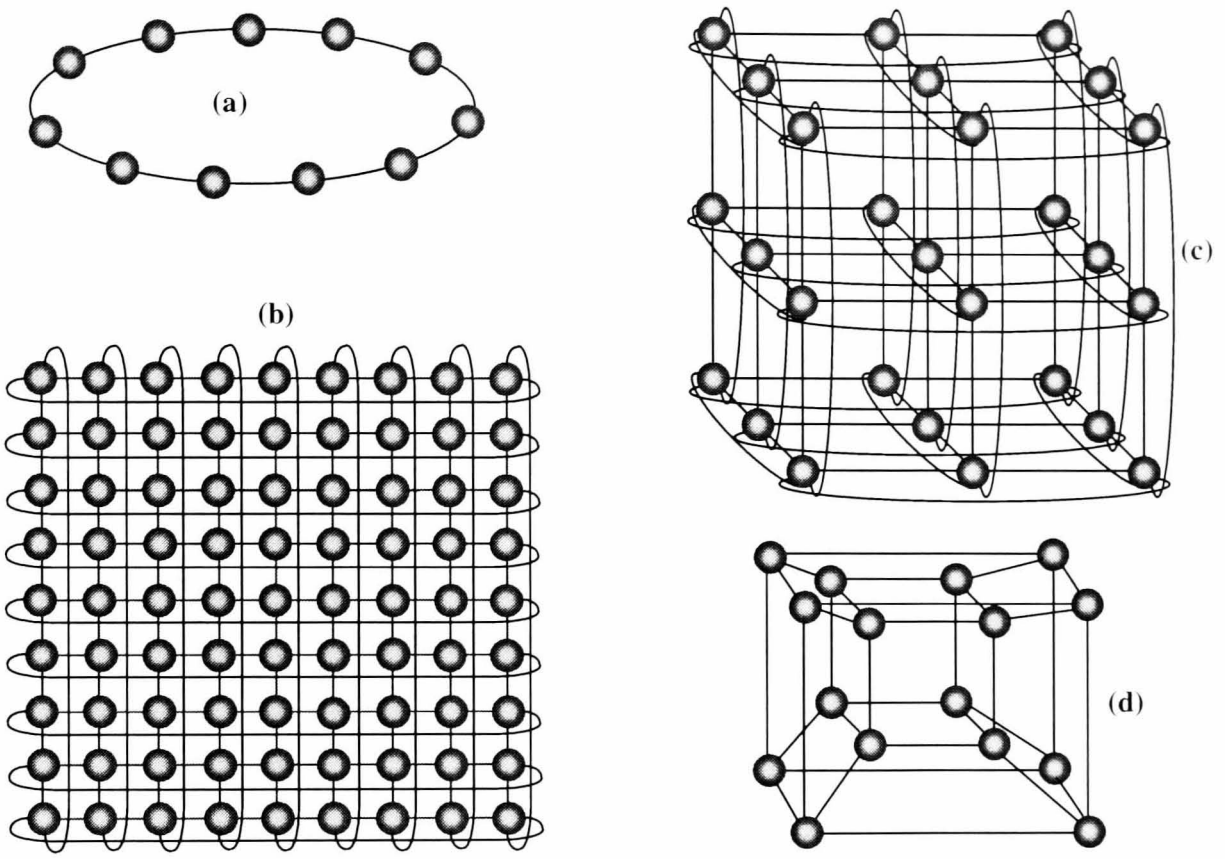


Figure 1.2: Some popular topologies. (a) A ring (with 11 nodes), (b) 2-D torus (9x9 nodes), (c) 3-D torus (3x3x3 nodes), and (d) 4-D hypercube or 4-cube.

value (in hops) of the minimum distances between any two nodes in network. The number of links adjacent to a node is called *node degree* and *network degree* is the maximum node degree in all network. A network is *regular* if all nodes have the same degree. Finally, a network is *symmetric* if it is isomorphic to itself with any node labelled as origin. The best topology is one which is regular and symmetric with small diameter and node degree [110].

Many topologies have been proposed for multicomputers [110] including the star [8, 9], cube-connected cycles [154], generalised hypercube [26], pyramid, and k-ary n-cube. Figure 1.2 illustrates some of the most commonly used direct networks, the ring (employed by KSR 1st-level ring [59]), 2-dimensional torus (used in iWARP [147]), 3-

dimensional torus (used by Cray T3D [98] and Cray T3E [13, 45]) and hypercube (employed in iPSC [16, 91, 140] and nCUBE [131-133]). These all belong to a major family of networks, called k-ary n-cubes, which have many desirable topological properties including ease of implementation, modularity, symmetry, low diameter and node degree, plus an ability to exploit locality exhibited by many parallel applications [160]. *K-ary* n-cubes are suited to a variety of applications including matrix computation, image processing and problems whose task graphs can be embedded naturally into the topology [130].

In a k-ary n-cube $N=k^n$ nodes are arranged in n dimensions, with k nodes per dimension. A node belongs to all n dimensions and is connected to two neighbours in each. Nodes located at physical boundaries of a dimension are wrapped around with a link in each (see Figure 1.2 for some examples). Links can be bidirectional or unidirectional; if bidirectional the wrap-around links may be omitted. Examples of k-ary n-cubes include, the ring (n=1), the 2-D and 3-D torus (n=2, 3), the hypercube or binary n-cube (k=2). The 2-D and 3-D meshes are examples of k-ary n-cubes with bidirectional links but without wrap-around connections, used in several real machines including the Intel Paragon [92] and MIT J-machine [138, 139] and M-machine [69]. In this study, the term k-ary n-cube will be used to mean k-ary n-cubes with wrap-around unless otherwise indicated.

1.1.2 Switching method

Switching method determines the way messages visit intermediate nodes. Several methods have been described in the literature, of which the two most important in multicomputers are *store-and-forward* [176] and *wormhole switching* [47]. In store-and-forward switching a node will not forward an incoming message till it has the entire message stored in its channel buffer. While most first generation multicomputers employed store-and-forward switching (of which the commonest form is packet switching), wormhole switching (also

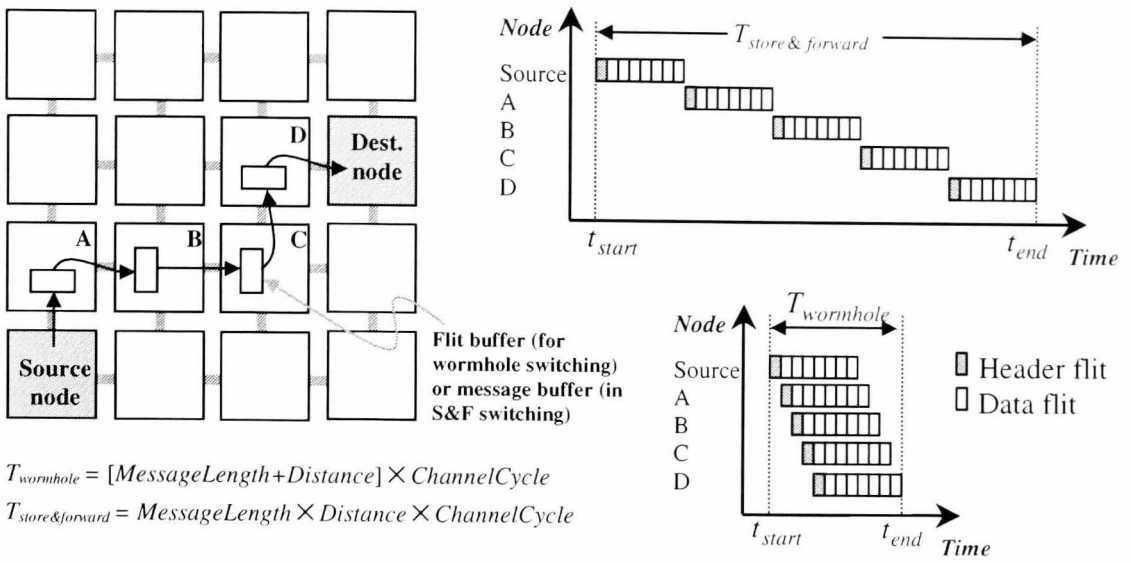


Figure 1.3: Transmission of an 8-flit message from the source node to the destination node destined 5 hops away in a 4x4 mesh (left) using store-and-forward (right up) and wormhole switching methods (right down) via intermediate nodes A, B, C and D.

called wormhole routing) has been widely used in recent multicomputers [59] due to its low buffering requirement and good performance. Here, a message is divided into *flits* (flow control unit) for transmission and flow control and each channel buffer need be only one flit in length[1]. The first flit of a message, the *header* flit, includes the routing information and is followed by the data flits in pipelined fashion. If the header cannot be routed in the network due to contention for resources (buffers and channels), the data flits are also blocked in situ, keeping all the allocated buffers occupied. Since wormhole routing uses pipelining, it can perform well even in a high diameter network. Figure 1.3 illustrates the transmission of an 8-flit message from a source node to a destination node destined 5 hops away in a 4x4 mesh using store-and-forward and wormhole switching methods. Many experimental machines, such as iWarp [147], J-Machine [139], and Caltech Mosaic [172], and commercial ones including Intel Paragon [92], Cray T3D [98], Cray T3E [13, 45], CM-5 [111], and nCUBE 2/3 [132, 133] use wormhole routing.

[1] Wormhole switching has been derived from *virtual cut-through* [93] switching where each channel buffer is enough big to keep an entire message.

Because wormhole flits can be blocked in the network and then occupy switching resources, the method requires careful deadlock control [51]. One solution to this problem is the use of *virtual channel flow control* [49]. Flow control concerns techniques for dealing with contention by multiple messages for the same channels and buffers [158]. A good flow control policy should reduce congestion, be fair and retain low latency. Flow control techniques are very dependent on the switching scheme employed [74]; in wormhole routing, the commonest flow control strategy is the use of *virtual channels.*

Virtual Channel

The preceding switching techniques assume that messages or parts of messages are buffered at the input and output of each buffer channel. Buffers are commonly operated as first-in-first-out (FIFO) queues. Therefore, once a message occupies a buffer for a channel, no other message can access the physical channel even if the message is blocked [59]. Due to the chained blocking property of wormhole switching, the bandwidth of interconnection networks is then limited to a fraction (20%-50%) of the total available physical bandwidth [6, 48]. However, it is also possible to multiplex several communications on a flit-by-flit basis by decoupling the allocation of buffers and physical channels [64], thus dividing a physical channel into several logical or virtual sub-channels. A virtual channel consists of a buffer—together with associated state information—capable of holding one or more flits of a message [49]. Virtual channels were first introduced in [49] to prevent deadlocks in wormhole networks based on the torus routing chip [50]. In [49, 59], it has been shown that virtual channels can also be used to improve network performance and latency by relieving contention. It is often observed that increasing the number of virtual channels will increase the network performance [59]. The advantages and disadvantages of using virtual channels have been thoroughly investigated (e.g. see [15, 46, 67, 74, 106, 158, 166]).

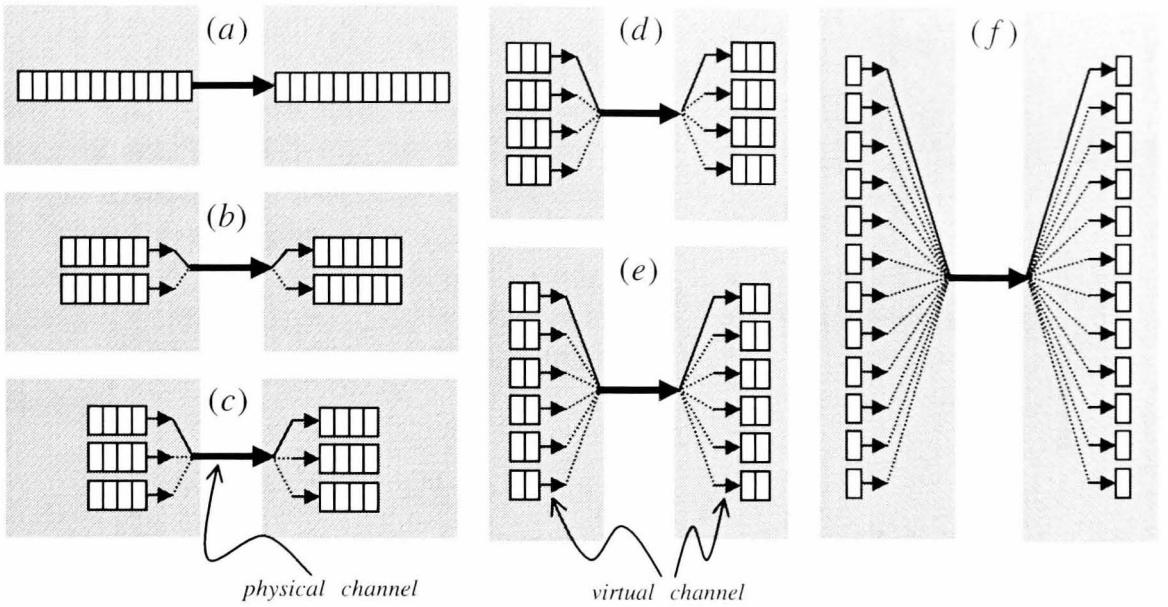


Figure 1.4: Organizing a 12-flit buffer dedicated to a physical channel; (*a*) Conventional routers organize it into one FIFO queue, while a network using virtual channels may organize it into several independent lanes resulting in different number of virtual channel each with different queue length, namely (*b*) 2×6 flits, (*c*) 3×4 flits, (*d*) 4×3 flits, (*e*) 6×2 flits, and (*f*) 12×1 flit.

Virtual channels dedicated to a physical channel may be organised in different ways. Figure 1.4 shows several organisations of virtual channels for a physical channel with a 12-flit buffer. The architectures differ in terms of performance, hardware requirements, and particularly arbiter complexity. Performance is dependent on network parameters and there is an optimal number of virtual channels where the network performance is maximised [49, 59].

The flit-level flow control discussed in [49] utilizes virtual channels efficiently as it permits the messages to cross the network channels in a time-multiplexed fashion. The header flit of a message directly determines the physical route (channel) required to reach its destination but a channel allocation algorithm selects the virtual route. When a header flit arrives at an input port, an available virtual channel is assigned to it. A header flit advances through a switch if: 1) it gains access through the crossbar; and 2) an available

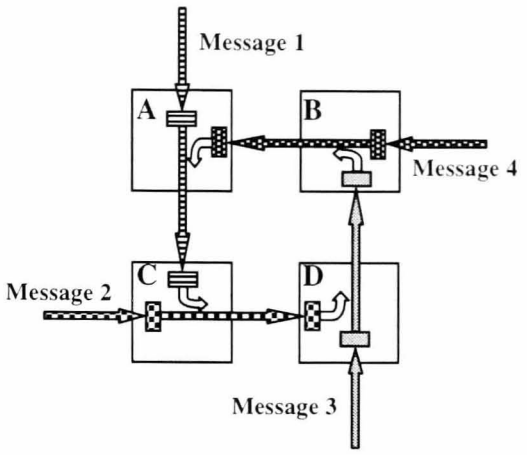


Figure 1.5: A deadlock situation in wormhole routing where no message can advance towards its destination. Messages 1, 2, 3 and 4 are destined respectively to nodes D, B, A, and C. Patterned paths are occupied by messages and whole arrows show the desired directions to be passed by messages.

virtual channel exists at the receiving end. The virtual channels acquired by the header flit at each routing step are used by the remaining flits to advance in the network. Once a message is allocated a virtual channel, the channel is not relinquished until blocking conditions arise. When the blocking conditions are removed, the message competes with other virtual channels to access the physical channel. When the last flit of a message leaves a router, the virtual channel that is hosting that message is de-allocated.

When virtual channel flow control is employed, a transmission policy is needed to mediate access to each physical channel. One simple policy is *work conserving round robin* in which unblocked messages occupying virtual channels on the same physical channel are alternately selected for transmission of a single flit [74]. As another example, a policy that could potentially reduce variance in message latencies is *oldest flit first* where flits belonging to the oldest unblocked message are given transmission priority [74].

1.1.3 Routing algorithm

Most interconnection networks, including k-ary n-cubes, provide multiple physical paths for routing a message between two given nodes. This introduces the problem of choosing a best route between many possible alternatives. Routing is a means used to achieve this.

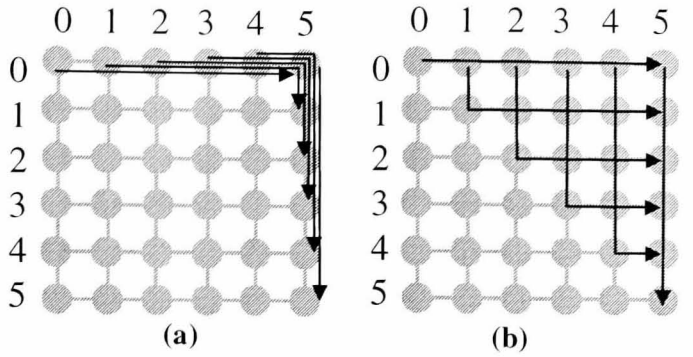


Figure 1.6: Routing messages in an 6x6 mesh from node (0,*i*) to node (*i*,5) (for *i*=0,1,...,5); (a) Using dimension order routing, five messages must traverse the channel from (0,4) to (0,5), (b) Using adaptive routing, all messages proceed simultaneously.

An important requirement for any routing algorithm is to ensure deadlock freedom; deadlock situations occur when no message can advance towards its destination because of occupied channels and buffers [59]. Many studies [61, 62, 78, 80, 118, 119, 128, 178] have addressed this issue in multicomputer networks. Figure 1.5 illustrates a deadlock situation where four messages are blocked and each one wants to acquire a channel being used by another.

Many practical multicomputers have used deterministic routing [65] with virtual channels to ensure deadlock avoidance. This is achieved by forcing messages to visit the virtual channels in a strict order [59]. Consequently, messages always take the same path between a given pair of nodes. This form of routing has the advantage of being simple, but is unable to adapt to conditions such as congestion or failures. *Dimension-ordered routing* [59] is a typical example of deterministic routing where messages visit network dimensions in a pre-defined order. However, if any channel along the message path is heavily loaded, the message experiences large delays and if any channel along the path is faulty the message cannot be delivered at all.

Adaptive routing improves both the performance and fault tolerance of an interconnection network and, more importantly, it has the ability to provide performance which is less sensitive to the communication pattern [51]. In this case, the paths can be chosen

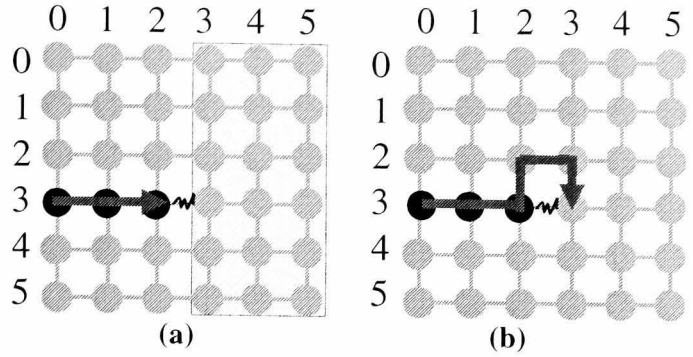


Figure 1.7: A 6x6 mesh with a faulty link from node (3,2) to node (3,3). (a) With dimension order routing messages from dark nodes to the shaded area cannot be delivered. (b) With adaptive routing, messages can be delivered between all pairs of nodes.

according to the degree of congestion of the node where the routing decision is taken. Figure 1.6 shows a 6x6 mesh in which the node $(0, i)$ sends a message to the node $(i, 5)$ for i=0,1,...,5. With dimension order deterministic routing five of the six messages must traverse the channel from (0, 4) to (0, 5), as shown in Figure 1.6(a). Thus only one of these five messages can proceed at a time. With adaptive routing (Figure 1.6(b)) all of the messages can proceed simultaneously using alternate paths. Figure 1.7 shows the same network with a faulty channel from (3, 2) to (3, 3). With dimension-ordered routing messages from node $(3, i)$ to node (k, j) where $0 \le i \le 2 < j \le 5$, $0 \le k \le 5$, cannot be delivered. With adaptive routing all messages can be delivered by routing around the faulty channel. In this example, it is necessary for the messages initially to be routed away from the destination node, resulting in a non-optimal distance route [51].

With a *non-minimal* routing algorithm, the selected path may not always be a shortest path while a *minimal adaptive* routing algorithm limits the path selection to the shortest paths between any given pair [51]. Many adaptive routing algorithms (minimal and non-minimal) have been developed for k-ary n-cubes [59]. These algorithms display interesting tradeoffs between their degree of adaptivity and the number of virtual channels needed. Introducing more adaptivity usually requires an increase in the number of virtual channels [145]. For example, Linder and Harden [116] have described fully adaptive minimal algorithms for k-ary n-cubes with unidirectional and bidirectional links. For the

unidirectional case, n+1 virtual channels are needed for each physical channel. For k-ary n-cubes with bidirectional links, 2^{n-1} virtual channels are needed per physical link in each direction, if the network has no wrap around links. With toroidal cubes the number increases to $(n+1)2^{n-1}$.

Boppana and Chalasani [30] have proposed another approach to design deadlock-free adaptive routing algorithms, based on the idea of the *structured buffer pool* method, traditionally used in store-and-forward networks [176]. Each physical channel is split into D virtual channels, where D is the diameter of the network. To guarantee deadlock-freedom, messages cross virtual channels according to the number of hops they have made in the network. Upon reaching an intermediate node, a message uses the h^{th} virtual channel to complete its h^{th} hop. Again, the high number of virtual channels required in this routing algorithm makes it impractical in large diameter networks, e.g. k-ary n-cubes.

Partially adaptive routing as an approach to trade off adaptivity against the number of virtual channels has gained much attention [35, 40, 51, 77]. Chien and Kim [40] have presented an algorithm, called *planar adaptive routing*, which is minimal and partially adaptive. This approach involves examining the routing dimensions in pairs, and constraining the routing choices at any time to one or two dimensions. This, in general, is less flexible than the fully adaptive routing algorithm of Linder and Harden, but requires only a constant number of virtual channels, regardless of the network dimension. For example, in a k-ary n-cube without wrap around connections, only three virtual channels for each physical link are required. The *turn model* [77] prevents some of the transitions between dimensions, and generalizes to multidimensional meshes and binary n-cubes. This scheme requires only a virtual channel per physical channel, is non-minimal, and partially adaptive. The approach for two-dimensional meshes works by disallowing two of the eight possible turns a packet may take. The turn model can be used in conjunction with virtual channels to increase adaptivity and to generalize to k-ary n-cubes. Thus, instead of

prohibiting some turns, the packet can be switched to a different virtual channel upon taking such a turn. Dally and Aoki [51] used this idea to design partially adaptive non-minimal routing algorithms for the class of k-ary n-cubes. In their algorithms, each packet carries with it a dimension reversal number which keeps track of the number of times the packet has been routed from a channel in one dimension to a channel in a lower dimension.

Most of the proposed adaptive routing algorithms require a high number of virtual channels. This high number of virtual channels results in increased hardware complexity, which can reduce router speed, causing significant degradation in network performance [39]. The high cost of adaptivity has motivated researchers to develop adaptive routing algorithms that require a smaller number of virtual channels. Several authors like Duato [63], Lin *et al* [115], and Su and Shin [174] have proposed fully adaptive routing algorithms, which can achieve deadlock-freedom with a minimal requirement for virtual channels, allowing for an efficient router implementation. Duato's fully adaptive routing algorithm is the best known of these and has been widely studied, having sufficient and necessary conditions for deadlock freedom with a minimum number of virtual channels.

1.1.4 Traffic pattern

One of the most important factors influencing network performance is the traffic pattern generated by the applications being executed on a machine. Different algorithms can generate very different traffic patterns. One way of optimising performance is to develop efficient algorithms that generate traffic compatible with the network's other properties. For example, we have described the design of efficient algorithms for a parallel numerical interpolation method on the k-ary n-cube [163], the star [164], and the cube-connected cycles [165] interconnection networks, by studying the nature of the Lagrange

interpolation method itself as well as the topological properties and communication capabilities of the target networks.

Interconnection network research must incorporate good models of message traffic. The traffic model is basically defined by three parameters: *message injection time*, *message length* and *message destination address distribution* [88]. The most frequently used, simplest and most elegant model is the classical *uniform traffic* model where processors target each other (or memory modules) with equal probability, at some rate per cycle. Successive requests are independent and processors also make requests independently [87, 104]. This simple model has widely been used [3, 4, 5, 6, 12, 32-34, 42, 43, 48, 49, 58, 81, 84-86, 99, 120, 142, 143] to drive most queuing-theory-based studies of interconnection network.

Based on observations of locality in message traffic, researchers have proposed extensions to the uniform traffic model. Some prefer a *sphere of locality* model [6, 160], in which a processor is more likely to send messages to a small number of destinations within a so-called "sphere of locality". Destinations outside the sphere are requested less frequently. Several ways to define the sphere exist; it can be based on physical network distances [1], network partitions [54, 89], or combinations of the two.

A message traffic model that has attracted much attention is the *hotspot* model studied in [149] where all (or a large number of) processors attempt to send messages to a single destination with relatively high probability. This may lead to extreme network congestion resulting in serious performance degradation due to the tree saturation phenomenon first observed in multi-stage networks. Hotspot traffic behaviour may be exhibited in many applications such as cache coherency protocols, synchronisation and many operating system functions [88]. It can be produced directly by certain collective communication operations including *gathering* and *barrier synchronisation* [127].

Permutation traffic is exhibited in many parallel applications such as computing multidimensional FFT (Fast Fourier Transform), matrix problems, finite elements and fault-tolerant routing [90]. In a permutation traffic pattern the destination address is solely determined by the source address using a permutation function [79]. Examples of permutation functions are *bit reversal, matrix transposition, shuffle, unshuffle, butterfly,* and *exchange* (see [59, 79, 90] for more examples of permutation routing and applications employing them). A permutation function, f, maps any address $X = x_1 x_2 \cdots x_n$ to a destination address $f(X) = x'_1 x'_2 \cdots x'_n$ where x'_i, $1 \le i \le n$, can be any of the x_i or their complements. Although most of permutation routing functions have been defined for binary input addresses (where x_i= 0 or 1), they may also be defined for radix-k addresses ($0 \le x_i < k$), e.g. the digit reversal permutation which deals with addresses (of radix-k digits) in several applications like radix-k FFT and related transforms [59].

1.2 Implementation constraints

In comparative evaluation of interconnection networks one must take account of implementation constraints to give a meaningful evaluation. Two such implementation constraints have been chosen to enable a fair comparison between networks of constant cost: pin-out constraints [4, 6] and wiring density constraints [48, 52].

The wiring complexity of the system is an important issue since the silicon area is limited and in general networks are wiring intensive [21, 73, 159]. In [48] Dally tried to quantify the implementation costs of different multidimensional networks in order to compare their performance under a constant cost constraint. He chose network bisection width [177] as cost measure and concluded that at fixed bisection width lower dimensional structures are best. A constant bisection width constraint is a measure of wiring complexity and is particularly relevant when the network wiring is implemented on a single chip or board. Dally's results have influenced the migration of commercial multicomputer networks from

hypercubes to 2D meshes and tori over several years. The J-machine [139], iWarp [147], and Stanford DASH [112] are examples of low-dimensional torus and mesh multicomputers.

Abraham and Padmanabhan [4], however, applied a constant pin-out constraint, which may be a more relevant cost constraint for today's pin-limited chips or where connector costs for cabling between cabinets is a consideration. With such a constraint, higher dimensional networks look attractive again although the analysis of [4] does not consider the longer wiring delays that might provide an additional penalty to such networks.

Agarwal [6] has examined the cut-through switching method in k-ary n-cubes under both constant pin-out and bisection width constraints and obtained results similar to Dally's, although favouring networks of somewhat higher dimensionality than Dally. Other authors have compared the performance of multidimensional tori using both of the above cost constraints, and also considered different wire delay models to account for the longer lengths required to implement higher dimensional topologies [23, 60, 167].

For instance Basak and Panda [23] have introduced another constraint for studying interconnection networks which they term a *packaging constraint*. As large systems require several levels of packaging, they have used the bisection width and pin-out constraints at each level. A typical hierarchy used in packaging a large system consists of multiple chips on a board and multiple such boards in a card-cage. A large system may require multiple card-cages, multiple cabinets and so on. The modules at each level of this packaging hierarchy: chips, boards, card-cage etc. have their own characteristics in terms of maximum capacity, bisection width, available pin-out and channel width. Considering a wide range of parameters, they have concluded that the best configurations are achieved with cluster-based systems with up to 8 processors per cluster with a 3- to 5-dimensional inter-cluster interconnection networks [23].

Scott and Goodman [167] have considered line pipelining which reduces the effect of long wire delays. The majority of their analysis is for a non-contention model, although they also provide extensive network simulations for contention situations. They conclude that the optimal radix in unconstrained situations is 2 (hypercube), and that for a constant pin-out constraint the optimal radix is 4 to 10, and that under constant bisection width the optimal radix is 16 to 32.

It can be concluded that the topology that looks best is clearly highly dependent on the constraint chosen and the wire delay model used [2, 117]. However, other network parameters (routing algorithm, switching method, router hardware cost, ...) and the traffic pattern used are of great importance. All of above studies assume deterministic routing, uniform message traffic and networks without virtual channels.

1.3 Performance modelling

The popularity of multicomputers is exemplified by the proliferation of a variety of parallel machines with diverse design philosophies. This range of architectural design has created a need for developing performance models for multicomputers not only to analyse the effectiveness of their design but also to reduce the design space [99].

Analytical models are cost-effective and versatile tools for evaluating system performance under different design alternatives. Since simplifications are often made to reduce the complexity of models, there is a need to validate the models through simulation. Validation is typically carried out for test cases, which require reasonable computation time and resources. The significant advantage of analytical models over simulation is that they can be used to obtain performance results for large systems and behaviour under network configurations and working conditions which may not be feasible to study using simulation on conventional computers due to the excessive computation demands [141].

Using analytical models, one can see the effect of each parameter on the system performance including those parameters related to the network configuration, implementation choices and traffic load. Realising such detailed investigations through simulation may take months or years, depending on the machine employed, and an efficient analytical model often provides a great reduction in the time required for such investigations. Most analytical models proposed for the performance evaluation of interconnection networks use results from queuing theory, providing a relatively straightforward derivation of performance expressions.

1.4 Related work

Within the literature on networks, the work on analytical performance modelling and constraint-based performance comparison of interconnection networks is most related to this thesis.

The first direct network stochastic performance study considered only the hypercube [3], assuming store-and-forward switching method with uniform message traffic and deterministic routing.

In [48], Dally introduced a mathematical model for predicting the average message latency in a unidirectional k-ary n-cube. This study assumes wormhole switching without virtual channels, deterministic routing and uniform message traffic. Since the model does not take account of blocking behaviour in the network, it is most applicable for light to moderate traffic and loses its accuracy when applied to a network with higher loads.

Abraham and Padmanabhan's [4] model for bidirectional k-ary n-cubes considers store-and-forward and virtual cut-through switching methods, deterministic routing and uniform

message traffic. They have used this model to compare the performance merits of multi-dimensional k-ary n-cubes.

The analytical model described in [6] for both unidirectional and bidirectional k-ary n-cubes uses virtual cut-through switching, deterministic routing, and uniform message traffic and considers locality. However, this model again omits the effect of virtual channels and the approximations used for incorporating message contention lead to under-estimation or over-estimation of performance depending on the network configuration [13].

Draper and Gosh [58] introduced an accurate model for unidirectional k-ary n-cubes using wormhole switching, deterministic routing, and uniform traffic but their study does not consider the effect of virtual channels.

Adve and Vernon [5] proposed a model for k-ary n-cubes employing virtual cut-through and wormhole switching methods without virtual channels. The model considers deterministic routing and uniform traffic and is approximately accurate.

Anderson and Abraham [13] proposed some models for unidirectional and bidirectional k-ary n-cubes employing store-and-forward and virtual cut-through switching, deterministic routing and uniform traffic. These models are fairly accurate in all traffic regions.

Cincinani, Colajanni and Paolucci [42] introduced a model for the 3D torus with wormhole switching and deterministic routing. It does not consider virtual channels and uses uniform message traffic to approximate mean message latency fairly accurately.

Greenberg and Guan [81, 83] proposed a model for the two-dimensional mesh and torus employing wormhole switching, deterministic routing, and assuming a uniform message traffic. The model loses its accuracy slightly when approaching heavy traffic loads.

Kim and Das [99] introduced an analytical model for hypercubes using wormhole and virtual cut-through switching without virtual channels using both deterministic and random routing and uniform message traffic. Their model takes account of the blocking effect and displays good accuracy.

Another model for hypercubes is proposed by Hady and Menezes [85, 86] using wormhole switching, deterministic routing and uniform traffic. Results obtained through simulation show close agreement to those produced by the analytical model.

Boura and Das [33, 36] have proposed an accurate analytical model for wormhole-routed hypercubes employing virtual channels and adaptive routing with uniform traffic. The model considers blocking behaviour and is fairly accurate for all traffic loads. A similar model was developed in [120, 121] for wormhole-routed tori deriving necessary expressions for computing the probability of message blocking. Computing this probability becomes more complicated for a general k-ary n-cube. Ould-Khaoua [142] has introduced a model for unidirectional k-ary n-cubes. His model considers wormhole switching and virtual channels, adaptive routing, uniform message traffic and locality. While the model exhibits a good degree of accuracy under light and moderate traffic loads, it loses its accuracy as the network enters the heavy traffic region. The discrepancy between the model and simulation is more noticeable as the number of alternative paths increases in the network, for example when n is moderately big. This is due to approximations when computing the probability of message blocking.

1.5 Motivations and outline of the thesis

In order to design high-performance multicomputers it is essential that the performance capabilities of their interconnection networks be completely understood. As discussed earlier, one approach to such performance studies is to build analytical models.

Table 1.1: Proposed models for *k*-ary *n*-cubes.

Model	Network	Switching method	Dealing With virtual channels	Routing Algorithm	Traffic Pattern
Dally [48]	Unidirectional *k*-ary *n*-cubes	Wormhole	No	Deterministic	Uniform
Abraham and Padmanabhan [4]	Bidirectional *k*-ary *n*-cubes	Store&Forward and Virtual Cut-through	No	Deterministic	Uniform
Agarwal [6]	Unidirectional and Bidirectional *k*-ary *n*-cubes	Virtual Cut-through	No	Deterministic	Uniform and Locality
Draper and Ghosh [58]	Unidirectional *k*-ary *n*-cubes	Wormhole	No	Deterministic	Uniform
Adve and Vernon [5]	Unidirectional and Bidirectional *k*-ary *n*-cubes	Virtual Cut-through and Wormhole	No	Deterministic	Uniform
Anderson and Abraham [13]	Unidirectional and Bidirectional *k*-ary *n*-cubes	Store&Forward and Virtual Cut-through	No	Deterministic	Uniform
Cinciani, Colajanni and Paolucci [42]	Unidirectional and Bidirectional *k*-ary *n*-cubes	Wormhole	No	Deterministic	Uniform
Ould-Khaoua [142]	Unidirectional *k*-ary *n*-cubes	Wormhole	Yes	Adaptive	Uniform

Most recent multicomputers use k-ary n-cubes as their underlying topology and employ wormhole switching [59]. Several analytical models [4, 6, 13, 42, 48, 58] have been proposed in the literature for deterministic routing in wormhole-routed k-ary n-cubes. Although deterministic routing algorithms are simpler to implement and many machines use deterministic routing [65], they cannot exploit network channels efficiently since messages cannot use alternative paths to avoid congested channels and thus reduce message communication latency. Fully adaptive routing overcomes this limitation by enabling messages to explore all the available paths between source and destination nodes. Several recent machines have used adaptive routing (e.g. Cray T3E [13, 45] and Reliable router [53]), and it continues to be a favoured routing algorithm in multicomputers. However, there have been few analytical models proposed for adaptive routing. Table 1.1 summarises the major analytical models of k-ary n-cubes so far introduced, addressing their main characteristics. As can be seen in the table, most of models deal with deterministic routing and do not consider the effect of virtual channels. The only model that uses adaptive routing and deals with virtual channels is Ould-Khaoua's; however, this loses accuracy as the network is subjected to heavy traffic load, although it exhibits a good degree of accuracy in light and moderate traffic. The discrepancy between the model and simulation is more noticeable as the number of alternative paths increases in the network and, therefore, it is more applicable to low-dimensional high-radix k-ary n-cubes (small n and large k). All these models assume a uniform traffic pattern, which is generally not a suitable choice for a typical load generated by a real application.

If adaptive routing is to be widely adopted in practical machines, it is necessary to assess its behaviour and suitability for different workload characteristics. Although it is very difficult to define a typical real-world traffic pattern, we may use some non-uniform traffic patterns created by known communication algorithms and applications (e.g. permutation traffics) or observed in practice (e.g. hotspot and locality). The most commonly

encountered of these are hotspot traffic and the permutation traffic characteristic of digit-reversal (known as bit-reversal for the hypercube) and matrix-transpose permutations. These patterns have previously mostly been studied via simulation experiments [59, 74, 158].

In this study, we first construct an accurate and exhaustive model for general wormhole-routed k-ary n-cubes, employing adaptive routing both with uniform traffic and including an element of locality. No such general model, capable of coping with arbitrary network size and traffic load, has yet been developed. Then, we introduce new models that deal with three important non-uniform traffic patterns: hotspot, matrix-transpose and digit-reversal. The performance merits of k-ary n-cubes will be analysed using these models and the effect of different parameters on that performance will be assessed. Finally, we shall apply our models to adaptive wormhole routing under uniform and non-uniform traffic, taking into account the effect of virtual channels, to reassess the performance merits of multi-dimensional k-ary n-cubes under constant pin-out and bisection-width constraints.

The rest of the thesis is organized as follows. Chapter 2 gives the preliminaries required for understanding the next chapters and reports some important findings on topological properties of k-ary n-cubes. It starts with defining the k-ary n-cube structure and deriving some topological properties of k-ary n-cubes; in particular, it gives some expressions for calculating the number of nodes which are at (and within) a given distance from a given node in the unidirectional and bidirectional k-ary n-cubes. Theses expressions will then be used in the next chapters for developing some analytical models. Finally, a methodology for designing adaptive routing algorithms in multi-computer networks is described and the methodology is applied to design some adaptive routing algorithm for wormhole-switched k-ary n-cubes.

Chapter 3 proposes an accurate analytical model of adaptive routing in wormhole-routed k-ary n-cubes with uniform traffic and validates it through simulation experiments. It starts with modelling a unidirectional network and then extends it for the bidirectional k-ary n-cube. The model is also described with locality in the traffic pattern.

In Chapter 4, a model is introduced for adaptive wormhole routing for k-ary n-cubes in the presence of hotspot traffic. The model uses the method employed by Pfister and Norton [149] for producing hotspot traffic. The model for the unidirectional k-ary n-cube is firstly described and then extended to consider bidirectional networks.

Analytical models of adaptive routing in wormhole-routed unidirectional k-ary n-cubes for digit-reversal and matrix-transpose permutation traffics are introduced in Chapter 5. Like previous models, these are then extended to bidirectional k-ary n-cubes.

The structure of Chapter 3, 4 and 5 for development, validation and extension of the analytical models is as follows. We first develop the model for unidirectional k-ary n-cubes. The required changes in the model equations are then discussed for the hypercube as a special case of the unidirectional k-ary n-cubes. We then validate the model through simulation experiments. The bidirectional extension of the model is then developed and, finally, some analysis is realised using the proposed model.

In Chapter 6, using the analytical models constructed in Chapters 3, 4 and 5, performance merits of hypercube and torus interconnection networks are compared under uniform, hotspot, digit-reversal and matrix-permutation traffic patterns. The comparison considers both pin-out and bisection bandwidth constraints with both normal and pipelined channels.

Finally, Chapter 7 concludes the thesis and outlines some directions for future work considering areas which might be worthy of further research.

Chapter 2

The *k*-Ary *n*-Cube: Structure, Properties and Routing

In this chapter, we derive some results on the topological properties of *k*-ary *n*-cubes which will be used when constructing our models in the next chapters. We then examine *k*-ary *n*-cubes, their node structure and Duato's adaptive routing algorithm which are necessary for understanding the models developed in the following chapters.

2.1 The *k*-ary *n*-cube

The *k*-ary *n*-cube is probably the most widely deployed [7, 13, 16, 93, 98, 107, 108, 112, 131, 139, 147, 170] multicomputer network topology [59] with many desirable properties including, symmetry, regularity, good node degree and diameter. It is well-suited to many applications such as matrix computation, image processing, and many others that can directly be mapped onto grid structures [130]. The three commonest instances of the *k*-ary *n*-cube are the 2 and 3-dimensional tori (*k*-ary 2-cube and *k*-ary 3-cube), both widely employed in recent machines, and the *n*-dimensional hypercube (2-ary *n*-cube), popular in early multicomputers.

A k-ary n-cube, Q_n^k, where n and k are referred to as *dimension* and *radix* respectively, has k^n identical nodes arranged in n dimensions with k nodes in each dimension. Node **A** in Q_n^k is labelled with a distinct n-digit radix k vector $[a_{n-1}, a_{n-2}, \ldots, a_0]$, where a_i, $0 \le i \le n-1$, indicates the position of the node in the i^{th} dimension.

The k-ary n-cube can be either *bidirectional* or *unidirectional*. In the unidirectional network, $\vec{Q}_n^k$, there is a unidirectional link from node **A** = $[a_{n-1}, a_{n-2}, \ldots, a_0]$ to node **B** = $[b_{n-1}, b_{n-2}, \ldots, b_0]$ iff there is an i, $0 \le i \le n-1$, such that $a_i = (b_i + 1) \bmod k$ and $a_j = b_j$, $0 \le j \le n-1$, $j \ne i$. Thus, in $\vec{Q}_n^k$ each node is adjacent with one node in each dimension, hence n nodes in total.

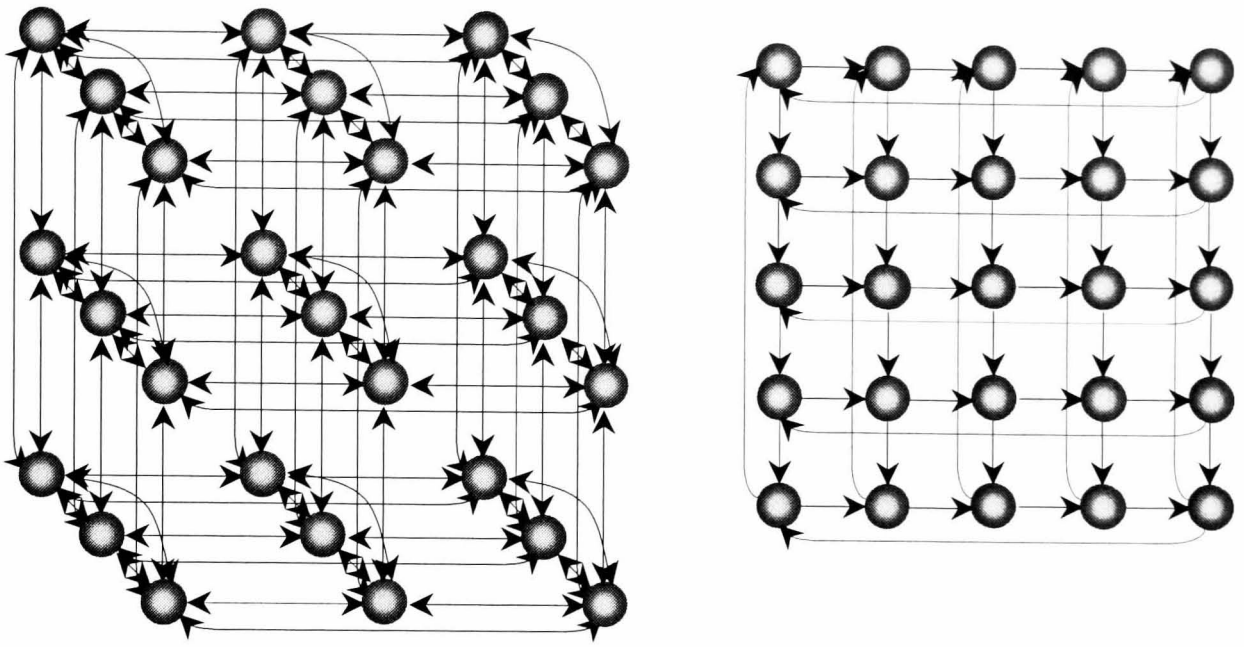
In the bidirectional k-ary n-cube, on the other hand, the concept of Lee distance is useful.

DEFINITION 2.1. *Lee weight* [31, 37]. Let **A**=$[a_{n-1}, a_{n-2}, \ldots, a_0]$ be an n-digit radix k vector. The Lee weight of **A** is defined as

$$W_L(\mathbf{A}) = \sum_{i=0}^{n-1} \|a_i\|, \text{ where } \|a_i\| = \min(a_i, k - a_i).$$

DEFINITION 2.2. *Lee Distance* [31, 37]. The Lee distance between two vectors **A** and **B** is denoted by $D_L(\mathbf{A},\mathbf{B})$ and is defined to be $W_L(\mathbf{A}-\mathbf{B})$. That is, the Lee distance between two vectors is the Lee weight of their bit-wise difference (mod k).

For example, when k=4, $W_L([3,2,1]) = (4-3)+2+1 = 4$, and $D_L([1,2,3],[3,2,1]) = W_L([1,2,3]-[3,2,1]) = W_L([2,0,2]) = 4$. Just as the Hamming distance may be used to define the hypercube and generalised hypercube graphs, the Lee distance may be used to define the bidirectional k-ary n-cube [31, 37] as follows.



(a) (b)

Figure 2.1: Examples of *k*-ary *n*-cubes; (a) A $\vec{Q}_3^3$ and (b) A $\vec{Q}_2^5$.

A bidirectional *k*-ary *n*-cube, $\vec{Q}_n^k$, has k^n nodes where any two nodes **A** and **B** are interconnected if and only if $D_L(\mathbf{A},\mathbf{B})=1$. Each node in a $\vec{Q}_n^k$ is connected to $2n$ adjacent nodes through $2n$ bidirectional channels, two at each dimension.

Alternatively, both unidirectional and bidirectional *k*-ary *n*-cubes can be defined as a cross product of n cycles of length k [55]. The cross product of the graphs $G_1=(V_1,E_1)$, $G_2=(V_2,E_2),\ldots,$ and $G_n=(V_n,E_n)$, denoted by $G_1\otimes G_2\otimes\cdots\otimes G_n$, is a new graph $G=(V,E)$, where $V=\{(v_1,v_2,\ldots,v_n)\,|\,v_i\in V_i,\quad (1\le i\le n)\}$ and $E=\{[(v_1,v_2,\ldots,v_n),(u_1,u_2,\ldots,u_n)]\mid \exists i,1\le i\le n,$ such that $[v_i,u_i]\in E_i$ and $v_j=u_j$ for $j\ne i\}$. Hence, Q_n^k can be defined as a product of cycles as [31, 37, 55]

$$Q_n^k=\overbrace{C_k\otimes C_k\otimes\cdots\otimes C_k}^{n\text{ times}}$$

with C_k being a cycle of k nodes which can be unidirectional or bidirectional resulting in a product graph of $\vec{Q}_n^k$ or $\tilde{Q}_n^k$, respectively. Figure 2.1 shows two examples of the unidirectional and bidirectional k-ary n-cubes.

2.2 Some topological properties of k-ary n-cubes

Many aspects of the k-ary n-cube have been extensively studied in the past, including its topological properties [17, 18, 25, 31, 37, 76], routing [18, 56, 61, 62, 76, 80, 145, 180], load balancing [84], performance analysis [4-6, 12, 23, 24, 39, 42, 43, 48, 58, 60, 66, 81, 83, 100, 102, 120, 121, 141-143, 158, 167], resource placement [20, 156], etc. In this section, we derive some fundamental properties of the k-ary n-cube. In particular, we furnish exact expressions that compute the number of nodes located at/within distance i from a given node in both the unidirectional and bidirectional k-ary n-cube. These results are interesting in their own right and can help us better understand the k-ary n-cube and explore properties that may be used in other problems. For instance, when a node is viewed as a root of a spanning tree in the k-ary n-cube, such expressions determine the number of nodes at level i in the tree. Therefore, the results of this study are useful in the study of spanning tree structures, which have been widely employed in the design of efficient multicast and broadcast operations, e.g. see [31, 70-72]. Furthermore, our results can also be used in the study of the *resource placement* problem in the k-ary n-cube [17, 146]. We shall use them in the following chapters when developing our models.

2.2.1 Problem definition

In $\vec{Q}_n^k$, node $\mathbf{A} = [a_{n-1}, a_{n-2}, \ldots, a_0]$ is at distance i from node $\mathbf{B} = [b_{n-1}, b_{n-2}, \ldots, b_0]$ if $\sum_{j=0}^{n-1}(a_j - b_j) \bmod k = i$. In the bidirectional case, $\tilde{Q}_n^k$, node $\mathbf{A} = [a_{n-1}, a_{n-2}, \ldots, a_0]$ is at distance i from node $\mathbf{B} = [b_{n-1}, b_{n-2}, \ldots, b_0]$ if $D_L(\mathbf{A}, \mathbf{B}) = i$.

Table 2.1: The surface area in a bidirectional 4-ary *n*-cube

	n=2	*n*=3	*n*=4	*n*=5	*n*=6	*n*=7	*n*=8
***i*=1**	4	6	8	10	12	14	16
***i*=2**	6	15	28	45	66	91	120
***i*=3**	4	20	56	120	220	364	560
***i*=4**	1	15	70	210	495	1001	1820
***i*=5**		6	56	252	792	2002	4368
***i*=6**		1	28	210	924	3003	8008
***i*=7**			8	120	792	3432	11440
***i*=8**			1	45	495	3003	12870
***i*=9**				10	220	2002	11440
***i*=10**				1	66	1001	8008
***i*=11**					12	364	4368
***i*=12**					1	91	1820
***i*=13**						14	560
***i*=14**						1	120
***i*=15**							16
***i*=16**							1

DEFINITION 2.3. *Surface Area* [31, 37]: The surface area $A_n^k(i)$ is the number of nodes in Q_n^k whose distance from a given node is exactly i. That is, $A_n^k(i)$ is the surface area of a sphere of radius i.

For instance, Tables 2.1 and 2.2 give the number of nodes at distance i from a given node in $\vec{Q}_n^4$ and $\vec{Q}_n^5$, respectively, for $2 \le n \le 8$, i.e., $\overleftrightarrow{A}_n^4(i)$ and $\overleftrightarrow{A}_n^5(i)$.

DEFINITION 2.4. *Volume* [31, 37]: The volume $V_n^k(i)$ is the number of nodes in Q_n^k whose distance from a given node is less than or equal to i. That is, $V_n^k(i)$ is the volume of a sphere of radius i. The volume can be written in terms of surface area as $V_n^k(i) = 1 + \sum_{j=1}^{i} A_n^k(j)$.

2.2.2 Related work

The objective of this study is to find expressions for computing $A_n^k(i)$ and $V_n^k(i)$ in both the unidirectional and bidirectional k-ary n-cube. Previous similar studies have considered

Table 2.2: The surface area in a bidirectional 5-ary *n*-cube

	n=2	*n*=3	*n*=4	*n*=5	*n*=6	*n*=7	*n*=8
***i*=1**	4	6	8	10	12	14	16
***i*=2**	8	18	32	50	72	98	128
***i*=3**	8	32	80	160	280	448	672
***i*=4**	4	36	136	360	780	1484	2576
***i*=5**		24	160	592	1632	3752	7616
***i*=6**		8	128	720	2624	7448	17920
***i*=7**			64	640	3264	11776	34176
***i*=8**			16	400	3120	14896	53344
***i*=9**				160	2240	15008	68352
***i*=10**				32	1152	11872	71680
***i*=11**					384	7168	60928
***i*=12**					64	3136	41216
***i*=13**						896	21504
***i*=14**						128	8192
***i*=15**							2048
***i*=16**							256

other networks. For instance, in [1], an expression for calculating surface area in a *n*-dimensional hypercube was given. An expression for calculating surface area in the generalised hypercube is derived by Bhuyan and Agrawal in [26]. Calculations have also been carried out by Qiu and Akl [155] for the star graph, and by Corbett [44] for the rotator graph. The authors in [31, 37] have described only partial results for the problem of finding the surface area in the *k*-ary *n*-cube. They have provided an expression for the surface area in the bidirectional *k*-ary *n*-cube, but have restricted their analysis only to the simple case where the radius, *i*, is smaller than *k*/2 as this is easier to deal with than the general case where *i* may be any number from 1 to the network diameter. To the best of our knowledge, the present study is the first to report comprehensive general results for both the unidirectional and bidirectional *k*-ary *n*-cube.

2.2.3 The surface area and volume in the unidirectional *k*-ary *n*-cube

Let G define the set of all nodes in the unidirectional k-ary n-cube. To determine the number of nodes at distance i from a given node, **B**, in $\vec{Q}_n^k$, we will make use of the function *uniCOUNT* outlined below.

Function *uniCOUNT* (k, n, i);
BEGIN
 $Counter \leftarrow 0$;
 for all $\mathbf{A} \in G$-{**B**} **do**
  if $\sum_{j=0}^{n-1}(a_j - b_j) \bmod k = i$ **then** $Counter \leftarrow Counter + 1$;
 return *Counter*;
END;

Note that the position of the given node **B** does not change the result. Using the above function we can calculate the volume of the sphere of radius i as follows.

Function *uniVOLUME* (k, n, i);
BEGIN
 $Sum \leftarrow 1$;
 for $j \leftarrow 1$ **to** i **do** $Sum \leftarrow Sum + uniCOUNT(k,n,j)$;
 return *Sum*;
END;

THEOREM 2.1. The number of ways to make j hops over n_H dimensions such that the number of hops made in each dimension i, $1 \le i \le n_H$, is at most the i-th element of an n_H-tuple $H = (h_1, h_2, \cdots, h_{n_H})$, that is h_i, is given by

$$\psi(H,j)=\begin{cases}1, & j=0\\ 0, & j<0 \text{ or } n_H<1\\ \sum_{m=0}^{h_{n_H}}\psi(H-h_{n_H},j-m), & \text{otherwise}\end{cases}.$$

where $H-h_{n_H}$ results in an (n_H-1)-tuple which is H without the last element h_{n_H}.

PROOF. One way to distribute j hops over n_H dimensions such that no dimension i is assigned more than h_i hops is as follows. Suppose that one hop is assigned to dimension n_H. The remaining (j-1) hops are distributed over the (n_H-1) remaining dimensions resulting in $\psi(H-h_{n_H},j-1)$ ways of distribution. The same approach may be taken where dimension n_H is assigned 2, 3,..., or h_{n_H} hops resulting in, respectively, $\psi(H-h_{n_H},j-2)$, $\psi(H-h_{n_H},j-3)$, ..., or $\psi(H-h_{n_H},j-h_{n_H})$ ways for distributing the remaining j-2, j-3, ..., and j-h_{n_H} hops over the n_H-1 remaining dimensions. Therefore, the total number of ways to distribute j hops over n_H dimensions is the sum of all the cases, i.e. $\sum_{m=1}^{h_{n_H}}\psi(H-h_{n_H},j-m)$. When the number of the remaining hops is zero this means that all the hops have already been distributed over the dimensions in one possible way. When the number of remaining hops, excluding the hops made in dimension n_H, is negative this means that the particular way of distributing hops is impossible to achieve. Finally, considering all these combinations together with the case where no hop is assigned to dimension n_H yields the above equation. □

COROLLARY 2.1. The number of nodes, $\vec{A}_n^k(i)$, at distance i from a given node in $\vec{Q}_n^k$ is given by $\vec{A}_n^k(i)=\psi(H_0,i)$, where $H_0=(\overbrace{k-1,k-1,\cdots,k-1}^{n \text{ times}})$ is the initial tuple.

PROOF. It follows directly from Theorem 2.1, given that the maximum possible hops that a message may spend in each dimension in $\vec{Q}_n^k$ is k-1. □

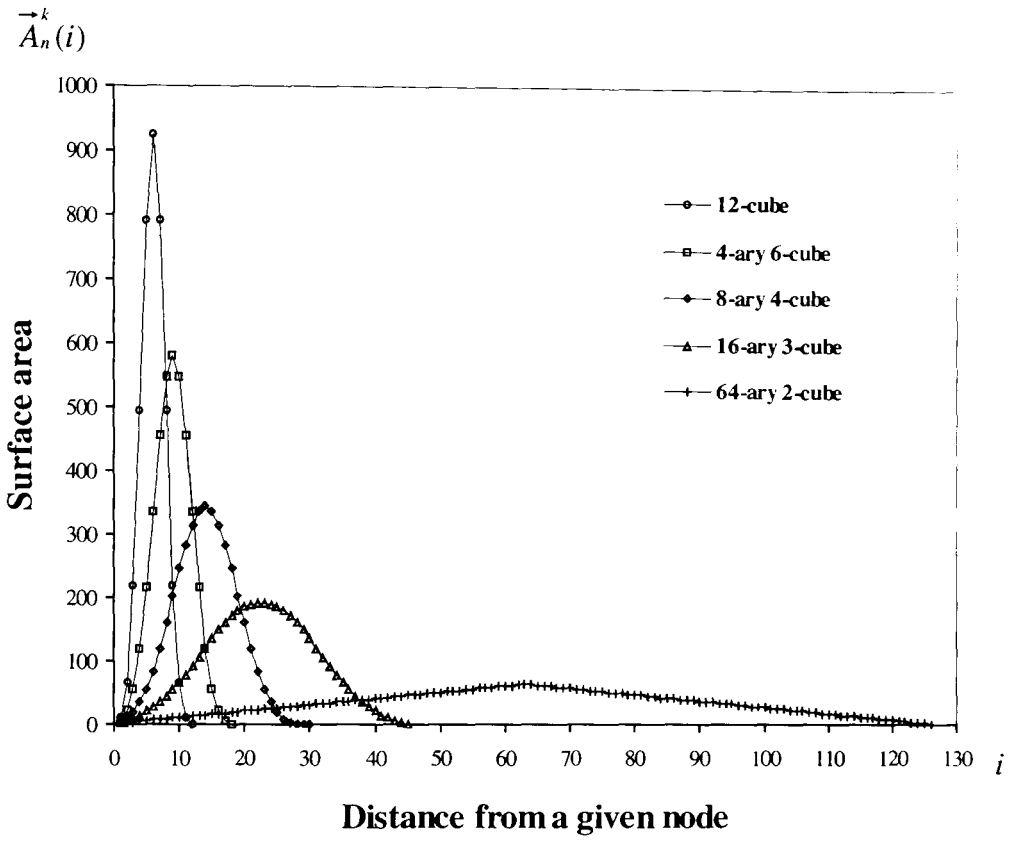


Figure 2.2: Number of nodes at distance i from a given node in a 4096-node $\bar{Q}_n^k$.

Now, a non-recursive expression is derived to count the number of nodes at distance i from a given node in the unidirectional k-ary n-cube. Let us first refer to the following result from combinatorial theory.

PROPOSITION 2.1. The number of ways to distribute r like objects (or indistinguishable object) into m different cells, such that no cell contains less than p objects and not more than $p+q-1$ objects is the coefficient of x^{r-pm} in the expansion of the polynomial $(1-x^q)^m(1-x)^{-m} = (1+x+x^2+.....+x^{q-1})^m$ [134].

Let the coefficient of x^{r-pm} be denoted as $\Gamma_p^{p+q-1}(r,m)$. In [161, 179], the expression for $\Gamma_p^{p+q-1}(r,m)$ is given by

$$\Gamma_p^{p+q-1}(r,m) = \sum_{l=0}^{m}(-1)^l\binom{m}{l}\binom{r-mp-lq+m-1}{m-1}.$$

THEOREM 2.2. The number of nodes at distance i from a given node in $\bar{Q}_n^k$ is given by

$$\vec{A}_n^k(i) = \sum_{l=0}^{n}(-1)^l\binom{n}{l}\binom{i-lk+n-1}{n-1}.$$

PROOF. The number of nodes located i hops away from a given node is equal to the number of possible paths via which an i-hop journey can be realized, starting from a given node, such that no channel and no node is traversed more than once and hops are always made to go further from the starting node. Such a journey is taken by an i-hop message routed across a unidirectional k-ary n-cube with a *minimal* routing algorithm, i.e. a routing algorithm that enables a message to select a shortest path to cross from source to destination [59]. The order at which the hops are made among dimensions is not important in our present calculation since we are primarily interested in determining the number of hops made at each dimension that leads to different ending nodes.

If the hops made by a message are treated as indistinguishable objects and the visited dimensions as different cells, the above proposition can be used to compute the number of nodes, $\vec{A}_n^k(i)$, which are i hops away from a given node in $\bar{Q}_n^k$. Taking into account the fact that a message may spend at least zero and at most k-1 hops at each dimension, we can write

$$\vec{A}_n^k(i) = \Gamma_0^{k-1}(i,n) = \sum_{l=0}^{n}(-1)^l\binom{n}{l}\binom{i-lk+n-1}{n-1}.\square$$

COROLLARY 2.2. In $\bar{Q}_n^k$, the volume, $\vec{V}_n^k$, of the sphere of radius i is given by

$$\vec{V}_n^k(i) = 1 + \sum_{j=1}^{i} \sum_{l=0}^{n} (-1)^l \binom{n}{l} \binom{j - lk + n - 1}{n - 1}.$$

PROOF. The claim follows directly from Theorem 2.2 and Definition 2.4. □

Figure 2.2 illustrates some results for $\vec{A}_n^k(i)$ in $\vec{Q}_n^k$ when the values of k and n are varied while keeping the total number of nodes fixed at 4096. Note that the surface area surrounded by each curve and the horizontal axis is equal to the total number of nodes in the network, i.e. 4096. The diagram shows that the hypercube is the richest network in the k-ary n-cube family from the connectivity point of view, and has the smallest diameter as its spanning tree is thick and short, compared to other equivalent k-ary n-cubes like the 2-dimensional torus.

2.2.4 The surface area and volume in the bidirectional k-ary n-cube

As with the function *uniCOUNT*, outlined in Section 2.2.3, for computing the surface area in the unidirectional k-ary n-cube, we can simply use function *biCOUNT*, shown below, to calculate the surface area in a bidirectional k-ary n-cube.

```
Function biCOUNT (k, n, i);
BEGIN
 Counter ← 0;
 for all A∈ G-{B} do
     if D_L(A,B) = i then Counter ← Counter + 1;
 return Counter;
END;
```

Using the above function we calculate the volume of the sphere of radius i using

Definition 2.4 as follows.

Function *biVOLUME* (*k, n, i*);
BEGIN
Sum ← 1;
for *j* ← 1 **to** *i* **do** *Sum* ← *Sum* + *biCOUNT*(*k*,*n*, *j*);
return *Sum*;
END;

THEOREM 2.3. In $\overleftrightarrow{Q}_n^k$, the number of ways to distribute j hops over $n_H\,(=n_{H^+}=n_{H^-})$ dimensions such that the number of hops made in each dimension $i, (1 \le i \le n_H)$, is at most the i-th element of either $H^+ = (h_1^+, h_2^+, \cdots, h_{n_{H^+}}^+)$ or $H^- = (h_1^-, h_2^-, \cdots, h_{n_{H^-}}^-)$, that is either h_i^+ or h_i^-, is given by

$$\Theta(H^+, H^-, j) = \begin{cases} 1, & j = 0 \\ 0, & j < 0 \text{ or } n_{H^+} < 1 \text{ or } n_{H^-} < 1 \\ \sum_{m=0}^{h_{n_{H^+}}} \Theta(H^+ - h_{n_{H^+}}, H^- - h_{n_{H^-}}, j-m) + & \\ \quad \sum_{m=0}^{h_{n_{H^-}}} \Theta(H^+ - h_{n_{H^+}}, H^- - h_{n_{H^-}}, j-m), & \text{otherwise} \end{cases}.$$

PROOF. Suppose that one hop is assigned to dimension n_H either in the positive direction or in the negative direction (n_{H^+} or n_{H^-}). The remaining j-1 hops are distributed over the n_H-1 remaining dimensions resulting in $\Theta(H^+ - h_{n_{H^+}}, H^- - h_{n_{H^-}}, j-1) + \Theta(H^+ - h_{n_{H^+}}, H^- - h_{n_{H^-}}, j-1)$ ways where the first term counts the case where the hop is made at dimension n_{H^+} in the tuple H^+ (in the positive direction) and the second term counts for the case where the hop is made in the negative direction. The same approach

may be taken where dimension n_{H^+} (or n_{H^-}) is assigned 2, 3, ..., $h_{n_{H^+}}$ (or $h_{n_{H^-}}$) hops resulting in, respectively, $\Theta(H^+ - h_{n_{H^+}}, H^- - h_{n_{H^-}}, j-2) + \Theta(H^+ - h_{n_{H^+}}, H^- - h_{n_{H^-}}, j-2)$, $\Theta(H^+ - h_{n_{H^+}}, H^- - h_{n_{H^-}}, j-3) + \Theta(H^+ - h_{n_{H^+}}, H^- - h_{n_{H^-}}, j-3)$, ... , or $\Theta(H^+ - h_{n_{H^+}}, H^- - h_{n_{H^-}}, j - h_{n_{H^+}}) + \Theta(H^+ - h_{n_{H^+}}, H^- - h_{n_{H^-}}, j - h_{n_{H^-}})$ ways for distributing j-2, j-3, ..., and j-$h_{n_{H^+}}$ (j-$h_{n_{H^-}}$) or hops over the n_H-1 remaining dimensions. Therefore, the total number of ways to distribute j hops over n_H dimensions is the sum of all cases, i.e. $\sum_{m=0}^{h_{n_{H^+}}} \Theta(H^+ - h_{n_{H^+}}, H^- - h_{n_{H^-}}, j-m) + \sum_{m=0}^{h_{n_{H^-}}} \Theta(H^+ - h_{n_{H^+}}, H^- - h_{n_{H^-}}, j-m)$. When the number of the remaining hops is zero this means that all the hops have already been distributed over the dimensions in one possible way. When the number of remaining hops, excluding the hops made in dimension n_H, is negative, this means that the current particular way taken for distributing hops is impossible to achieve. Finally, considering all these combinations together yields the recurrence equation claimed. □

In the above expression, the tuple $H^+ = (h_1^+, h_2^+, \cdots, h_{n_{H^+}}^+)$ with n_{H^+} elements and $H^- = (h_1^-, h_2^-, \cdots, h_{n_{H^-}}^-)$ with n_{H^+} elements are the limit vectors (tuples) for the network in the positive and negative directions, where h_i^+, $(1 \le i \le n_{H^+})$, and h_i^-, $(1 \le i \le n_{H^-})$, are the maximum possible distance from a given node at dimension i in the positive and negative directions.

COROLLARY 2.3. The number of nodes at distance i from a given node in $\overleftrightarrow{Q}_n^k$ is given by

$$\overleftrightarrow{A}_n^k(i) = \Theta(H_0^+, H_0^-, i),$$

where $H_0^+ = (\lfloor \frac{k-1}{2} \rfloor, \lfloor \frac{k-1}{2} \rfloor, \ldots, \lfloor \frac{k-1}{2} \rfloor)$ and $H_0^- = (\lceil \frac{k-1}{2} \rceil, \lceil \frac{k-1}{2} \rceil, \ldots, \lceil \frac{k-1}{2} \rceil)$ are two initial n-tuple vectors.

PROOF. The claim follows directly from Theorem 2.3, having in mind that the maximum possible hops made by a message in each dimension is $\lfloor \frac{k-1}{2} \rfloor$ in one direction (say positive) and $\lceil \frac{k-1}{2} \rceil$ in the other direction (negative).□

Now, a non-recursive expression is derived to count the number of nodes at distance i from a given node in the bidirectional k-ary n-cube. In order to use Proposition 2.1 which deals with cells of equal capacity, we have to consider the problem separately for odd and even k. Let us first consider the problem for the simpler case where the network radix k is odd.

LEMMA 2.1. The number of nodes at distance i from a given node in $\vec{Q}_n^k$ when k is odd is given by

$$\overleftrightarrow{A}_n^k(i) = \sum_{m=0}^{n} \sum_{l=0}^{m} (-1)^l 2^m \binom{n}{m} \binom{m}{l} \binom{i - \frac{kl}{2} - 2l - 1}{m-1}.$$

PROOF. We should count the number of ways that i like objects can be distributed over two groups, each of n cells, say $G = \{C_1, C_2, \cdots, C_n\}$ and $G' = \{C'_1, C'_2, \cdots, C'_n\}$, such that each cell contains not more than (k-1)/2 objects and no two corresponding groups, C_i and C'_i for $i = 0,1,...,n$, can be assigned objects at the same time. The problem can be thought of as that of finding the number of different destinations that an i-hop message can choose from a given source node in $\vec{Q}_n^k$, when k is odd, using a minimal routing algorithm. It is apparent that a message can take at most (k-1)/2 hops in each dimension as the network is bidirectional.

Let us consider the case that an i-hop message makes some hops over m, ($m = 0,1,2,....n$), fixed dimensions (each in one direction) so that the message has made at least one hop in each dimension. This can be realized in $\Gamma_1^{\frac{k}{2}}(i,m)$ ways. Each of n dimensions could be in

these m dimensions resulting in $\binom{n}{m}\Gamma_1^{\frac{k}{2}}(i,m)$ possible combinations that m dimensions are passed (each in one direction). Recalling that each of two directions in one dimension can be chosen yields the total number of ways to pass m dimension with at least one hop in each dimension as $2^m\binom{n}{m}\Gamma_1^{\frac{k}{2}}(i,m)$. Summing up all the combinations for $m=0,1,2,\ldots,n$ gives the total number of nodes at distance i from a given node in $\ddot{Q}_n^k$ as

$$\overleftrightarrow{A}_n^k(i)=\sum_{m=0}^{n}2^m\binom{n}{m}\Gamma_1^{\frac{k}{2}}(i,m)=\sum_{m=0}^{n}\left[2^m\binom{n}{m}\sum_{l=0}^{m}(-1)^l\binom{m}{l}\binom{i-\frac{kl}{2}-2l-1}{m-1}\right]$$

$$=\sum_{m=0}^{n}\sum_{l=0}^{m}(-1)^l 2^m\binom{n}{m}\binom{m}{l}\binom{i-\frac{kl}{2}-2l-1}{m-1}. \square$$

Now, we consider the problem when the radix k is even.

LEMMA 2.2. *The number of nodes at distance i from a given node in a* $\ddot{Q}_n^k$ (with even k) *is given by*

$$\overleftrightarrow{A}_n^k(i)=\sum_{t=0}^{n}\sum_{m=0}^{n-t}\sum_{l=0}^{m}(-1)^l 2^m\binom{n}{t}\binom{n-t}{m}\binom{m}{l}\binom{i-\frac{k(l+t)}{2}-2l-1}{m-1}.$$

PROOF. We should count the number of ways that i like objects can be distributed over two groups, each of n cells, say $G=\{C_1,C_2,\cdots,C_n\}$ and $G'=\{C'_1,C'_2,\cdots,C'_n\}$, such that each cell in G contains not more than $k/2$-1 and each cell in G' contains not more than $k/2$ and no two correspondent cells, C_i and C'_i, for all $i=0,1,\ldots,n$, can be assigned objects at the same time. Suppose that t, $t=0,1,2,\ldots,n$, cells in G' have received $k/2$ objects, in $\binom{n}{t}$ ways. The remaining objects may be distributed over the remaining n-t dimensions using the equation given in Lemma 2.1, since each dimension in G and G' now receives at most $k/2$-1 objects. Therefore, we can write $\overleftrightarrow{A}_n^k(i)=\sum_{t=0}^{n}\binom{n}{t}\overleftrightarrow{A}_{n-t}^{k-1}(i-tk/2)$.

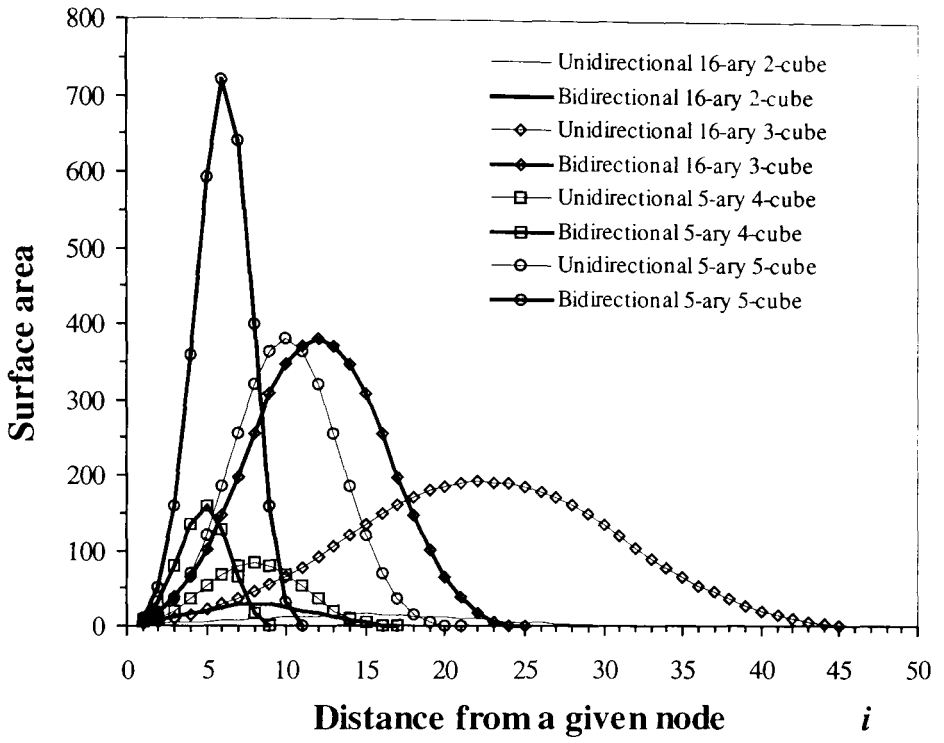


Figure 2.3: The surface area in some unidirectional and bidirectional *k*-ary *n*-cubes versus radius *i*.

Substituting $\overleftrightarrow{A}^{k-1}_{n-t}(i - tk/2)$ from Lemma 2.1 derives the equation claimed by the Lemma, i.e.

$$\overleftrightarrow{A}^{k}_{n}(i) = \sum_{t=0}^{n}\sum_{m=0}^{n-t}\sum_{l=0}^{m}(-1)^{l}2^{m}\binom{n}{t}\binom{n-t}{m}\binom{m}{l}\binom{i-\frac{k(l+t)}{2}-2l-1}{m-1}. \square$$

THEOREM 2.4. *The number of nodes at distance i from a given node in* $\bar{Q}^{k}_{n}$ *is given by*

$$\overleftrightarrow{A}^{k}_{n}(i) = \begin{cases} \sum_{m=0}^{n}\sum_{l=0}^{m}(-1)^{l}2^{m}\binom{n}{m}\binom{m}{l}\binom{i-\frac{kl}{2}-2l-1}{m-1}, & k \text{ is odd} \\ \sum_{t=0}^{n}\sum_{m=0}^{n-t}\sum_{l=0}^{m}(-1)^{l}2^{m}\binom{n}{t}\binom{n-t}{m}\binom{m}{l}\binom{i-\frac{k(l+t)}{2}-2l-1}{m-1}, & \text{otherwise} \end{cases}.$$

PROOF. The theorem follows directly from Lemmas 2.1 and 2.2. □

Figure 2.3 illustrates the surface area in some unidirectional and bidirectional k-ary n-cubes versus radius i.

COROLLARY 2.4. *The volume, $\overleftrightarrow{V}_n^k$, of the sphere of radius i in $\vec{Q}_n^k$ is given by*

$$\overleftrightarrow{V}_n^k(i)=\begin{cases}1+\sum_{j=1}^{i}\sum_{m=0}^{n}\sum_{l=0}^{m}(-1)^l 2^m\binom{n}{m}\binom{m}{l}\binom{j-\frac{kl}{2}-2l-1}{m-1} & k \text{ is odd}\\ 1+\sum_{j=1}^{i}\sum_{t=0}^{n}\sum_{m=0}^{n-t}\sum_{l=0}^{m}(-1)^l 2^m\binom{n}{t}\binom{n-t}{m}\binom{m}{l}\binom{j-\frac{k(l+t)}{2}-2l-1}{m-1} & \text{otherwise}\end{cases}$$

PROOF. It follows directly from Theorem 2.4 and Definition 2.4. □

As an illustration of the application of these results, let us calculate the mean distance, $\bar{d}$, traveled by a message in both $\vec{Q}_n^k$ and $\ddot{Q}_n^k$, assuming that destinations are uniformly distributed; the mean distance is the mean number of hops that a message makes when the destination address is chosen randomly. Using Theorems 2.2 and 2.4, we can calculate the mean message distance in $\vec{Q}_n^k$ and $\ddot{Q}_n^k$, respectively, as

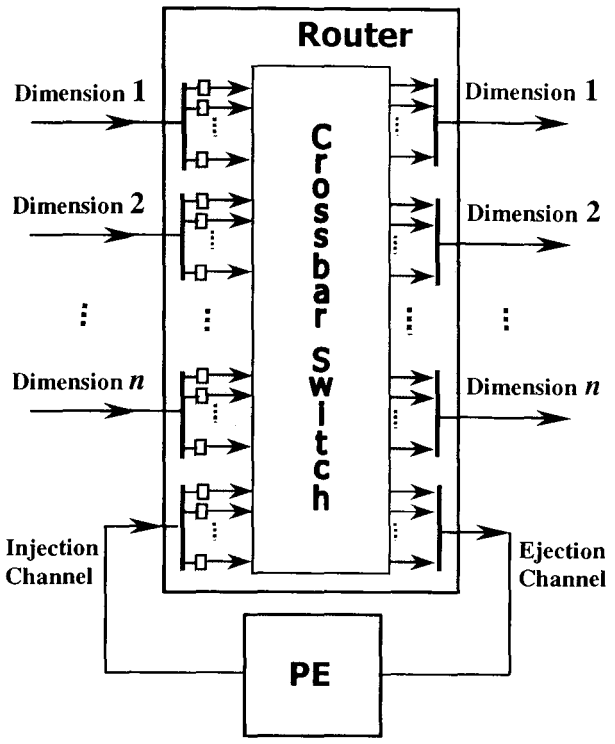
$$\vec{d}=\frac{1}{N-1}\sum_{i=1}^{n(k-1)} i\cdot\vec{A}_n^k(i)=\frac{1}{N-1}\sum_{i=0}^{n(k-1)}\sum_{l=0}^{n} i(-1)^l\binom{n}{l}\binom{i-lk+n-1}{n-1},$$

and

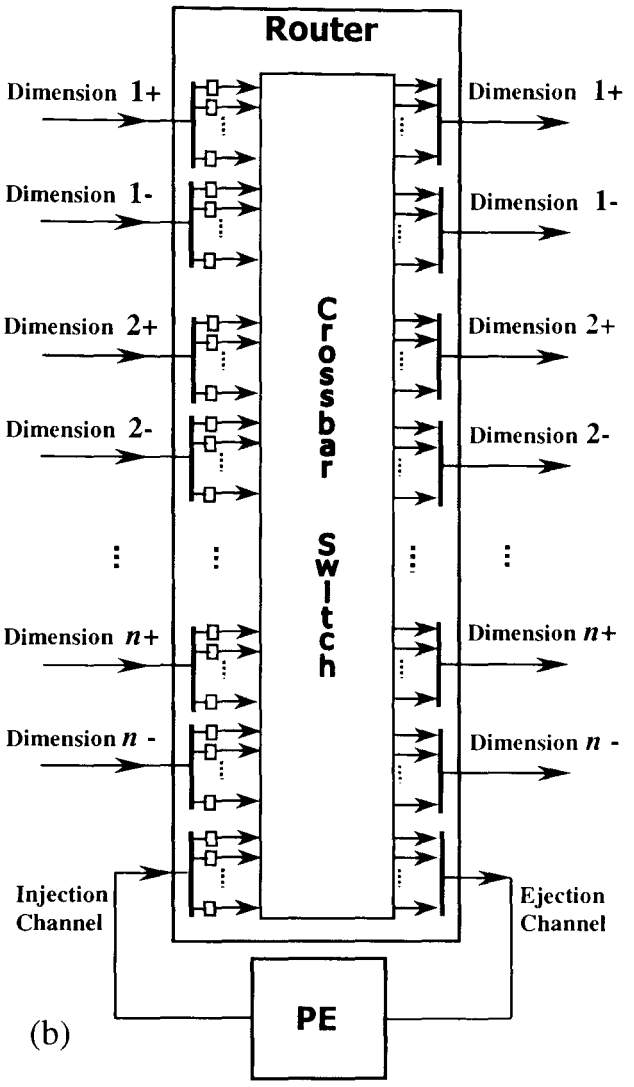
$$\ddot{d}=\frac{1}{N-1}\sum_{i=1}^{n\left\lceil\frac{k-1}{2}\right\rceil} i\cdot\overleftrightarrow{A}_n^k(i)$$

$$=\begin{cases}\frac{1}{N-1}\sum_{i=1}^{n\left\lceil\frac{k-1}{2}\right\rceil}\sum_{m=0}^{n}\sum_{l=0}^{m}(-1)^l 2^m\binom{n}{m}\binom{m}{l}\binom{i-\frac{kl}{2}-2l-1}{m-1} & k \text{ is odd}\\ \frac{1}{N-1}\sum_{i=1}^{n\left\lceil\frac{k-1}{2}\right\rceil}\sum_{t=0}^{n}\sum_{m=0}^{n-t}\sum_{l=0}^{m}(-1)^l 2^m\binom{n}{t}\binom{n-t}{m}\binom{m}{l}\binom{i-\frac{k(l+t)}{2}-2l-1}{m-1} & \text{otherwise}\end{cases}$$

Note that the terms $n(k-1)$ and $n\left\lceil\frac{k-1}{2}\right\rceil$ are the diameters in $\vec{Q}_n^k$ and $\ddot{Q}_n^k$, respectively.



(a)



(b)

Figure 2.4: The node structure in the (a) unidirectional *k*-ary *n*-cube, and (b) bidirectional *k*-ary *n*-cube.

2.3 Adaptive routing in *k*-ary *n*-cubes

This section first describes the node structure in a *k*-ary *n*-cube. It then introduces Duato's methodology for designing adaptive routing algorithms. The application of this methodology for designing adaptive routing algorithm for *k*-ary *n*-cubes is described.

2.3.1 Node structure

A *k*-ary *n*-cube has $N=k^n$ identical nodes each consisting of a processing element (PE) and

router, as shown in Figure 2.4. The PE contains a processor and some local memory. The router has $2n+1$ input channels and $2n+1$ output channels ($n+1$ input and $n+1$ output channels in a unidirectional cube). A node is connected to its neighbours through $2n$ inputs and $2n$ output channels (n input and n output channels in unidirectional cube); there are two channels (only one in a unidirectional cube) in each dimension corresponding to the positive and negative direction respectively. The remaining channels are used by the PE to inject messages into or absorb messages from the network. Messages generated by the PE are injected into the network through the injection channel. Messages at the destination node are transferred to the PE through the ejection channel. The router contains flit buffers for each input virtual channel. The input and output channels are connected by a crossbar switch that can simultaneously connect multiple input to multiple output channels given that there is no contention over the output channels.

2.3.2 Duato's adaptive routing algorithm

Many adaptive routing algorithms have been discussed in literature, each with their own special requirements [59]. Amongst these, Duato's algorithm [63] is attractive since it requires a limited number of virtual channels to ensure deadlock freedom. It has therefore been widely studied and is accepted as a practical approach to adaptive routing with minimal resource requirements. The Cray T3E [13, 45] and the reliable router [53] are two examples of recent practical systems that have adopted Duato's routing algorithm.

2.3.2.1 The general methodology

The chain of channels a message may pass to reach its destination is called a route or path. When a message reserves a channel, and later requests the use of another channel possibly several hops further on, there is a dependency between those channels. Also, at a given node, a message may request the use of several channels, then select one of them (adaptive

routing). With deterministic routing, messages have a single routing option at each node. In this situation, it is necessary to remove all the cyclic dependencies between channels to prevent deadlocks [51] or messages may indefinitely hold some channels while waiting for other channels that are held by competitors. When adaptive routing is considered, on the other hand, messages typically have several choices at each node and it is not necessary to eliminate all the cyclic dependencies, provided that every message can always find a path towards its destination whose channels are not in such dependencies. The channels of these paths can be considered as *escape channels* from the cycles [63]. A *routing subfunction* is a restriction of a routing algorithm which supplies escape channels.

THEOREM 2.5. A connected and adaptive routing function for an interconnection network is deadlock-free if there are no cycles in its channel dependency graph [63].

THEOREM 2.6. A connected and adaptive routing function for an interconnection network is deadlock-free if there exists a subset of network channels that defines a deadlock-free routing subfunction [63].

2.3.2.2 The algorithm for *k*-ary *n*-cubes

As discussed above, to design an adaptive routing algorithm for a k-ary n-cube, we need a deadlock-free routing subfunction. However, such a routing subfunction can itself be deterministic. In k-ary n-cubes, deadlock-free deterministic routing algorithms can be designed using virtual channels [59]. The authors in [51] have shown that the wrap-around connections in k-ary n-cubes can lead to deadlock situations due to the cyclic dependencies that can occur within a dimension but propose the use of an additional virtual channel to transform dependency *cycles* into *spirals*. On this basis, two virtual channels are needed to implement a deadlock-free deterministic routing algorithm in a k-ary n-cube network.

Applying Duato's methodology for k-ary n-cubes (k >2) requires V, (V >2), virtual channels per physical channel. The V virtual channels ($v_1, v_2, \cdots, v_V$) associated with a given physical channel are split into two sets: $VC_1 = \{v_1, v_2\}$ and $VC_2 = \{v_3, v_4, \cdots, v_V\}$. The two virtual channels in VC_1 (also called deterministic virtual channels) are used to implement a deadlock-free routing subfunction (escape paths), e.g. dimension order deterministic routing. The other virtual channels in VC_2 (called adaptive virtual channels), can be visited by a message in any order that brings it closer to its destination (as required by any minimal routing scheme). A message firstly checks all adaptive virtual channels of the remaining dimensions to be negotiated. If some adaptive virtual channels are free, one of them is chosen randomly to route through. If all adaptive virtual channels of the remaining dimensions are busy, the message is routed through the deterministic virtual channels of the lowest dimension to be passed. If the deterministic virtual channels are also busy the message is blocked and waits for a proper virtual channel becoming free.

Figure 2.5 illustrates Duato's minimal adaptive routing algorithm for unidirectional k-ary n-cubes in pseudo code. For a bidirectional k-ary n-cube, lines 2, 5 and 6 in the algorithm should change as follows.

2. $\mathbf{P}= \{ j^X \mid 1 \le j \le n, c_j \ne d_j,\ X = \begin{cases} + & \text{if } (c_j - d_j) \bmod k < (d_j - c_j) \bmod k \\ - & \text{otherwise} \end{cases},\ \exists v \in VC_2^{(j^X)} \text{ and } v \text{ is free} \};$

5. else $\{ q = max \{ i \mid 1 \le i \le n,\ c_i \ne d_i \};\ pc = \begin{cases} q+ & \text{if } (c_q - d_q) \bmod k < (d_q - c_q) \bmod k \\ q- & \text{otherwise} \end{cases};$

and

6. if $(c_q < d_q)$ then vc=1 else vc=2;

As mentioned before, when k=2 the k-ary n-cube collapses to the familiar hypercube. According to Duato's methodology, when designing adaptive routing for hypercubes with

ALGORITHM adaptive routing in the unidirectional *k*-ary *n*-cube;

INPUT: Destination node address vector $D = d_n d_{n-1} \cdots d_1$ and current node address vector $C = c_n c_{n-1} \cdots c_1$.

OUTPUT: The virtual channel *vc*, of the physical channel *pc*, to be taken by the message in the next hop. A returned value of zero in *pc* means that there is not any available virtual channel to take and the message is blocked.

BEGIN

1. if ($C = D$) then return *ejection channel*;
   // message has arrived at destination.
2. $\mathbf{P} = \{ j \mid 1 \le j \le n, c_j \ne d_j, \exists v \in VC_2^{(j)}$ and v is free $\}$;
3. *pc* = *random* **P**; // *random* gives an element of set **P** at random
   // it returns 0 if **P**={}.
4. if ($pc \ne 0$) then vc = *random* $\{v \mid v \in VC_2^{(pc)}$ and v is free $\}$;
5. else $pc = max \{i \mid 1 \le i \le n, c_i \ne d_i\}$;
   // *max* **A** returns the maximum element in set **A**
6. if ($c_i < d_i$) then vc=1 else vc=2;
7. if (virtual channel *vc* of physical channel *pc* is not free) then pc=0;

endif;

return *vc*, *pc*;

END;

Figure 2.5: Adaptive routing algorithm in the unidirectional *k*-ary *n*-cube.

V, (V>1), virtual channels per physical channel, the virtual channels are split into two sets: $VC_1 = \{v_1\}$ and $VC_2 = \{v_2, v_3, \cdots, v_V\}$. This is because only one virtual channel (here v_1 in VC_1) is needed to implement a deadlock-free deterministic sub-routing algorithm, e.g. e-cube routing algorithm [59], since with two nodes per dimension and minimal routing there is no cyclic dependency between channels of each dimension. As described earlier, the remaining virtual channels, those in VC_2, associated to all usable physical channels are adaptively used by message to get closer to their destinations. If all adaptive virtual

channels of usable physical channels are busy, the message is routed through the deterministic virtual channel of the physical channel associated to the highest dimension remaining to be traversed. The routing algorithm for the hypercube would be the same as unidirectional k-ary n-cube except for line 6 which should change to

6. vc=1;

as there are no wraparound connections in hypercubes and the deterministic routing subfunction for such networks requires only one virtual channel, according to e-cube routing (a dimension order routing algorithm) [59].

2.4 Summary

We have drawn a careful distinction between the unidirectional and bidirectional k-ary n-cube interconnection networks and then derived expressions for calculating the number of nodes at some distance from a given centre in each. These results are used in the models developed in the next chapters, but they are also interesting in their own right. For example they are very useful in the study of the spanning trees widely used in the design of collective communication (multicast and broadcast) and resource placement algorithms [20, 31, 70-72, 156].

We have described the basic router structure which supports wormhole switching in k-ary n-cubes and selected Duato's routing methodology for its ability to provide full adaptivity with a minimum requirement of virtual channels. We have applied this methodology to define a minimal adaptive routing algorithm for k-ary n-cubes. In the next chapters, we shall construct analytical models of k-ary n-cubes using this algorithm for routing with the aim of studying behavior under different traffic patterns and loads.

Chapter 3

Performance Modelling of Adaptive Wormhole Routing in k-Ary n-Cubes with Uniform Traffic

Several researchers have recently proposed analytical models of fully adaptive routing [33, 43, 120, 142]. For instance, Boura *et al* [33] have proposed a model of fully adaptive routing in the hypercube. More recently, the authors in [43, 120] have extended the model for the 2-dimensional torus while the author in [142] has generalised it for high-radix k-ary n-cubes.

The most difficult part in developing any analytical model of adaptive routing is the computation of the probability of message blocking at a given router due to the number of combinations that have to be considered when enumerating the paths that a message may have used to reach its current position in the network. The problem is further exacerbated when the network dimensionality increases since the number of alternative paths then also increases. While the model in [142] exhibits a good degree of accuracy under light and moderate traffic loads, it loses accuracy as the network enters the heavy traffic regions. The discrepancy between the model and simulation is more noticeable as the number of

alternative paths increases because the model resorts to approximations when computing the probability of message blocking.

This chapter describes a new analytical model for wormhole-switched k-ary n-cubes. The proposed model exhibits a good degree of accuracy in light, moderate and heavy traffic conditions. It achieves this because it computes the exact expressions for the probability of message blocking at a given router by considering all the possible paths that enable a message to cross from its source to its current position in the network. The model determines the value of different components that make up the average message latency, including the message transfer time and the blocking delay for messages, in the network. It also considers the use of virtual channels with the adaptive routing algorithm described in Chapter 2.

The analytical model is described first for unidirectional k-ary n-cubes. It is then extended for bidirectional k-ary n-cubes and for traffic patterns that exhibit communication locality. The remainder of this chapter is organised as follows. Section 3.1 outlines the assumptions used in the analysis. Section 3.2 presents the model for the unidirectional k-ary n-cube while Section 3.3 validates it through simulation experiments. Extensions of the model for the bidirectional k-ary n-cube and for capturing the effects of communication locality are described in Section 3.4. Section 3.5 uses the proposed model to analyse the performance of k-ary n-cubes for different network parameters. Finally, Section 3.6 concludes this chapter.

3.1 Assumptions

The model uses the following assumptions that are widely accepted in the literature [3-6, 12, 32-34, 42, 43, 48, 49, 58, 81, 84-86, 99, 120, 142, 143].

a) Nodes generate traffic independently of each other, following a Poisson process

with a mean rate of λ_g messages per cycle.

b) The arrival process at a given channel is approximated by an independent Poisson process. This approximation has often been invoked to determine the arrival process at channels in store-and-forward networks [103, 104]. Although wormhole routing differs from store-and-forward in various aspects (e.g. flit buffering and advancing as opposed to message buffering and forwarding), simulation experiments from previous studies have revealed that this is still a reasonable approach to determine the arrival process [36, 83, 85, 121, 141]. Therefore, the rate of the process arrival at a channel can be calculated using formulae borrowed from *Jackson's queueing networks* [104].

c) Message destinations are uniformly distributed across network nodes.

d) Message length is fixed and equal to M flits, each of which is transmitted in one cycle from one router to the next using wormhole switching.

e) The local queue at the injection channel in the source node has infinite capacity. Moreover, messages are transferred to the local processor as soon as they arrive at their destinations through the ejection channel.

f) V virtual channels are used per physical channel, as shown in the router structure illustrated in Figure 2.2(a). These are divided into two classes: VC_1 and VC_2. According to the adaptive routing algorithm, described in Section 2.2.2, Class VC_2 contains $(V-2)$ virtual channels that are crossed adaptively. On the other hand, class VC_1 contains two virtual channels that are crossed deterministically. Let the virtual channels belonging to class VC_2 and VC_1 be called the adaptive and deterministic virtual channels, respectively. When there is more than one adaptive virtual channel available a message chooses one at random. To simplify the model derivation no distinction is made between the deterministic and adaptive virtual channels when computing virtual channel occupancy probabilities [142].

3.2 The analytical model

The notation used to describe the model is briefly listed in Table 3.1. The model computes the mean message latency as follows.

3.2.1 Outline of the model

The mean network latency, $\overline{S}$, that is the time to cross the network, is first determined. Then, the mean waiting time seen by a message in the source node, $\overline{W_s}$, is evaluated. Finally, to consider the effects of virtual channels multiplexing, the mean message latency is scaled by a factor, $\overline{V}$, representing the average degree of virtual channels multiplexing that takes place at a given physical channel. Thus the mean message latency can be written as

$$Latency = (\overline{S} + \overline{W_s})\overline{V} \,. \tag{3.1}$$

The average number of hops that a message makes across one dimension and across the network, $\overline{k}$ and $\overline{d}$ respectively, are given by [6]

$$\overline{k} = \frac{k-1}{2}, \tag{3.2}$$

$$\overline{d} = n\overline{k} \,. \tag{3.3}$$

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. A router in the k-ary n-cube has n output channels and the PE generates, on average, λ_g messages in a cycle. Since each message travels, on average, $\overline{d}$ hops to cross the network the rate of messages received by each channel, λ_c, can be written as [6]

Table 3.1: Notation used in the model for uniform traffic

Symbol	Description
B_j	blocking time seen by a message at the j-th hop during in its journey
D	destination node
$\overline{d}$	average hops that a message takes in the network
H	set denoting the distance between S and D in each dimension i for $1 \le i \le n$
$\|H\|$	distance between the source node S and the destination node D
k	network radix
$\overline{k}$	average hops that a message takes in one dimension
Latency	average latency seen by a message
λ_g	message generation rate at a node
λ_c	messages arrival rate on a channel
M	message length
n	network dimension
n_H	number of elements in set H
N	network size
$\psi(h, j)$	number of ways to distribute j hops over n_H dimensions with at most h_i hops in dimension i for $1 \le i \le n$
P_a	probability that all adaptive virtual channels at a physical channel are busy
$P_{a\&d}$	probability that all adaptive and deterministic virtual channels at a physical channel are busy
P_{block_j}	probability that a message is blocked at the j-th hop channel
P_v	probability that v virtual channels at a physical channel are busy
$Pass_j^z$	probability of passing z dimensions after j-th hop in a journey towards D
π_v	state that v virtual channels of a physical channels are occupied
Q_v	temporary variable used for calculating P_v
S_H	message latency to cross the network from source node S to destination node D
$\overline{S}$	mean message latency
V	number of virtual channels per physical channel
$\overline{V}$	average multiplexing degree of the virtual channels at a physical channel
w	mean waiting time seen by a message at a given physical channel
W_s	mean waiting time seen by a message at the source node before entering the network

$$\lambda_c = \frac{\lambda_g \bar{d}}{n}. \tag{3.4}$$

Since the k-ary n-cube is symmetric, averaging the network latencies seen by the messages generated by a given node for all other nodes, gives the mean message latency in the network. Let $S = s_1 s_2 \cdots s_n$ be the source node and $D = d_1 d_2 \cdots d_n$ denotes a destination node such that $D \in G - \{S\}$ where G is the set of all nodes in the network. Let us define the set $H = \{h_i\}$, $(1 \le i \le n)$, where each element h_i denotes the number of hops that the message makes along dimension i when it traverses the network from the source node to the destination node, that is $(s_i + h_i) \bmod k = d_i$. The network latency, S_H, seen by the message crossing from node S to node D consists of two parts: one is the delay due to the actual message transmission time, and the other is due to the blocking time in the network. Therefore, S_H can be written as

$$S_H = |H| + M + \sum_{j=1}^{|H|} B_j, \tag{3.5}$$

where M is the message length, $|H|$ is the distance (in terms of the number of hops made by the message) between the source and the destination node, and B_j is the blocking time seen by a message on its j-th hop. The terms $|H|$ and B_j are given by

$$|H| = \sum_{i=1}^{n} h_i, \tag{3.6}$$

$$B_j = P_{block_j} w, \tag{3.7}$$

with P_{block_j} being the probability that a message is blocked on its j-th hop during its network journey and w is the mean waiting time to acquire a channel in the event of blocking. Let us now calculate the blocking probability P_{block_j}. To do so, let n_H denote the number of elements in the set $H = \{h_1, h_2, \cdots, h_n\}$, calculated for source and

destination nodes S and D as shown earlier. Recall that $\psi(H, j)$, defined by Theorem 2.1, can give the number of ways that j hops can be distributed over n_H dimensions such that the number of hops made in each dimension i, $(1 \le i \le n_H)$, can be at most the i-th element of the set H, that is h_i.

The probability that a message has entirely crossed one dimension on its j-th hop is therefore given by

$$Pass_j^1 = \frac{\sum_{l=1}^{n} \psi(H'(l), j - h_l)}{\psi(H, j)}, \tag{3.8}$$

where $H'(l) = \{h_1 - 1, h_2 - 1, \cdots, h_{l-1} - 1, 0, h_{l+1} - 1, \cdots, h_n - 1\}$. Similarly, the probability that a message has entirely crossed two dimensions on its j-th hop can be expressed as

$$Pass_j^2 = \frac{\sum_{l_1=1}^{n} \sum_{l_2=l_1+1}^{n} \psi(H'(l_1, l_2), j - h_{l_1} - h_{l_2})}{\psi(H, j)}, \tag{3.9}$$

where $H'(l_1, l_2) = \{h_1 - 1, h_2 - 1, \cdots, h_{l_1 - 1} - 1, 0, h_{l_1 + 1} - 1, \cdots, h_{l_2 - 1} - 1, 0, h_{l_2 + 1} - 1, \cdots, h_n - 1\}$. More generally, the probability that a message has entirely crossed z dimensions can be written as

$$Pass_j^z = \frac{\sum_{l_1=1}^{n} \sum_{l_2=l_1+1}^{n} \cdots \sum_{l_z=l_{z-1}+1}^{n} \psi(H'(l_1, l_2, \cdots, l_z), j - \sum_{i=1}^{z} h_{l_i})}{\psi(H, j)}, \tag{3.10}$$

where

$$H'(l_1, l_2, \cdots, l_n) = \{h'_1, h'_2, \cdots, h'_n\}, \tag{3.11}$$

$$h'_i = \begin{cases} 0 & i = l_1 \text{ or } i = l_2 \text{ or } \cdots \text{ or } i = l_z \\ h_i - 1 & \text{otherwise} \end{cases}. \tag{3.12}$$

A message is blocked at a given channel when all the adaptive virtual channels of the remaining dimensions to be visited and also the deterministic virtual channels of the lowest dimension still to be visited are busy. The probability of blocking depends on the number of output channels, and thus on the virtual channels that a message can use at its next hop. When a message has entirely crossed z dimensions it can select any of the available $(n\text{-}z)(V\text{-}2)$ adaptive virtual channels and one deterministic virtual channel to make its next hop. The probability of blocking, P_{block_j}, can therefore be written as

$$P_{block_j} = \sum_{z=0}^{n-1} Pass_j^z (P_a)^{n-z-1} P_{a\&d}, \tag{3.13}$$

with P_a being the probability that all adaptive virtual channels of a physical channel are busy and $P_{a\&d}$ being the probability that all adaptive and deterministic virtual channels of a physical channel are busy. To compute P_a three cases should be considered, and are as follows [142].

a) V virtual channels are busy which means all adaptive virtual channels are busy as well.

b) $(V\text{-}1)$ virtual channels are busy. The number of combinations where $(V\text{-}1)$ out of V virtual channels are busy is $\binom{V}{V-1}$ of which only two combinations result in all adaptive virtual channels being busy.

c) $(V\text{-}2)$ virtual channels are busy. The number of combinations where $(V\text{-}2)$ out of V virtual channels are busy is $\binom{V}{V-2}$ of which only one combination results in all adaptive virtual channels being busy.

Similarly, to compute $P_{a\&d}$, two cases should be considered, as follows [142].

a) V virtual channels are busy, that is all adaptive and deterministic virtual channels are busy.

b) (V-1) virtual channels are busy. In this case only two combinations out of $\binom{V}{V-1}$ result in all adaptive and deterministic virtual channels being busy.

Let P_v, $(0 \le v \le V)$, represent the probability that v virtual channels at a physical channel are busy. Taking into account the different cases mentioned above, P_a and $P_{a\&d}$ are given in terms of P_v by

$$P_a = P_V + \frac{2P_{V-1}}{\binom{V}{V-1}} + \frac{P_{V-2}}{\binom{V}{V-2}}, \tag{3.14}$$

$$P_{a\&d} = P_V + \frac{2P_{V-1}}{\binom{V}{V-1}}. \tag{3.15}$$

To determine the mean waiting time, w, to acquire a virtual channel a physical channel is treated as an M/G/1 queue with a mean waiting time of [104]

$$w = \frac{\rho \overline{S}(1+C_{\overline{S}}^2)}{2(1-\rho)}, \tag{3.16}$$

$$\rho = \lambda_c \overline{S}, \tag{3.17}$$

$$C_{\overline{S}}^2 = \frac{\sigma_{\overline{S}}^2}{\overline{S}^2}, \tag{3.18}$$

where λ_c is the traffic rate on the channel, $\overline{S}$ is its service time, and $\sigma_{\overline{S}}^2$ is the variance of the service time distribution. While λ_c is given by Equation 3.4 above, the quantities $\overline{S}$ and $\sigma_{\overline{S}}^2$ are computed as follows. Since adaptive routing distributes traffic evenly among

all channels, the mean service time at each channel is the same regardless of its position, and is equal to the mean network latency, $\overline{S}$. Equation 3.5 gives the network latency, S_H, seen by a message in crossing from the source node S to the destination node D. Averaging over the (N-1) possible destination nodes in the network yields the mean network latency as

$$\overline{S} = \frac{1}{N-1} \sum_{D \in G-\{S\}} S_H \,. \tag{3.19}$$

Since the minimum service time at a channel is equal to the message length, M, following a suggestion proposed in [58], the variance of the service time distribution can be approximated as $\sigma_{\overline{S}}^2 = (\overline{S} - M)^2$. Hence, the mean waiting time becomes

$$w = \frac{\lambda_c \overline{S}^2 (1 + \frac{(\overline{S} - M)^2}{\overline{S}^2})}{2(1 - \lambda_c \overline{S})} \,. \tag{3.20}$$

A message originating from a given source node sees a network latency of $\overline{S}$ (given by Equation 3.19). Modelling the local queue in the source node as an M/G/1 queue, with the mean arrival rate λ_g / V (recalling that a message in the source node can enter the network through any of the V virtual channels) and service time $\overline{S}$ with an approximated variance $(\overline{S} - M)^2$ yields the mean waiting time seen by a message at source node as

$$W_s = \frac{\frac{\lambda_g}{V} \overline{S}^2 \left(1 + \frac{(\overline{S} - M)^2}{\overline{S}^2} \right)}{2(1 - \frac{\lambda_g}{V} \overline{S})} \,. \tag{3.21}$$

The probability, P_v, that v virtual channels are busy at a physical channel can be determined using a Markovian model as shown in Figure 3.1. State π_v, $(0 \le v \le V)$,

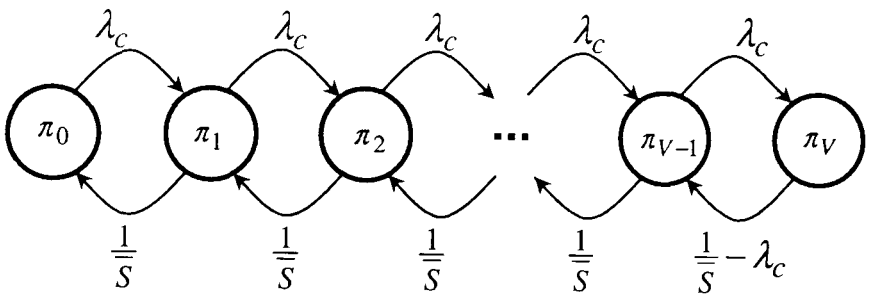


Figure 3.1: The Markov chain used for computing the probability of virtual channel occupancy of a physical channel.

corresponds to v virtual channels being busy. The transition rate out of state π_v to state π_{v+1} is the traffic rate λ_c (given by Equation 3.4) while the rate out of state π_v to state π_{v-1} is $\frac{1}{\overline{S}}$ ($\overline{S}$ is given by Equation 3.19). The transition rate out of state π_V is reduced by λ_c to account for the arrival of messages while a channel is in this state. The steady-state solutions of the Markovian model yield the probability P_v, $(1 \le v \le V)$, as [49]

$$P_v = \begin{cases} \dfrac{1}{\sum_{i=0}^{V} Q_i}, & v = 0 \\ P_{v-1}\lambda_c \overline{S}, & 0 < v < V, \\ P_{v-1}\dfrac{\lambda_c}{\frac{1}{\overline{S}} - \lambda_c}, & v = V \end{cases} \tag{3.22}$$

where

$$Q_v = \begin{cases} 1, & v = 0 \\ Q_{v-1}\lambda_c \overline{S}, & 0 < v < V. \\ Q_{v-1}\dfrac{\lambda_c}{\frac{1}{\overline{S}} - \lambda_c}, & v = V \end{cases} \tag{3.23}$$

When multiple virtual channels are used per physical channel they share the bandwidth in

a time-multiplexed manner. The average degree of multiplexing of virtual channels, that takes place at a given physical channel, can be estimated by [49]

$$\overline{V} = \frac{\sum_{v=1}^{V} v^2 P_v}{\sum_{v=1}^{V} v P_v}. \tag{3.24}$$

3.2.2 The hypercube case

When the network is a hypercube (a k-ary n-cube with k=2), some of equations derived above are modified as follows. The average distance, $\overline{d}$, traversed by a message crossing the network is given by [1]

$$\overline{d} = \sum_{i=1}^{n} i \frac{\binom{n}{i}}{N-1} = \frac{nN}{2(N-1)}. \tag{3.25}$$

Since in the hypercube, when a message makes one hop it has consequently passed one dimension, the probability that the message is blocked at its j-th hop is simply given by

$$P_{block\ j} = (P_a)^{|H|-j} P_{a\&d}. \tag{3.26}$$

Since only one deterministic virtual channel per physical channel is sufficient to ensure deadlock-free fully adaptive routing in the hypercube, as described in Section 2.3, the probability that all adaptive virtual channels associated to a physical channel are busy, P_a, and the probability that all adaptive and deterministic virtual channels associated with a physical channel are busy, $P_{a\&d}$, are expressed as [36]

$$P_a = P_V + \frac{P_{V-1}}{\binom{V}{V-1}}, \tag{3.27}$$

$$P_{a\&d} = P_V. \tag{3.28}$$

3.2.3 Implementation issues

The above equations reveal that there are several inter-dependencies between the different variables of the model. For instance, Equations 3.5, 3.7 and 3.19 reveal that $\overline{S}$ is a function of w while Equation 3.20 shows that w is a function of $\overline{S}$. Given that closed-form solutions to such inter-dependencies are very difficult to determine the different variables of the model are computed using iterative techniques for solving equations [104]. The procedure for computing the mean message latency using the above model is as follows.

Step 1) Let $\overline{S}$ be initialized to M.

Step 2) Compute P_v, P_a, $P_{a\&d}$ and w, using Equations 3.14, 3.15, 3.22, and 3.20.

Step 3) Compute S_H, for all $H \in N - \{S\}$, using Equation 3.5.

Step 4) Compute new $\overline{S}$ using Equation 3.19. If it is different from the old value by greater than ε (a predefined error limit) then go to Step 2.

Step 5) Compute W_s, $\overline{V}$ and *Latency* using Equations 3.21, 3.24 and 3.1.

To ease the implementation of the above procedure we can assume that node 0 (with address pattern 0, 0, 0, ..., 0) is the source node (S) and nodes 1, ..., k^n-1 are destination nodes ($D \in N - \{S\}$). This will simply result in H=D.

Note that we may avoid computing S_H for some H vectors recalling that, due to adaptive routing and network symmetry, the latency S_H is the same for all permutations of h_is, $1 \le i \le n$, in $H = (h_1, h_2, \cdots, h_n)$. Using this property the run time of the model will be lower. For example, for a 3-ary 3-cube, and assuming a source node at (000), we have $H = D \in$ {001, 002, 010, 011, 012, 020, 021, 022, 100, 101, 102, 110, 111, 112, 120, 121, 122, 200, 201, 202, 210, 211, 212, 220, 221, 222} for which we have

$S_{(001)} = S_{(010)} = S_{(100)}$, $S_{(002)} = S_{(020)} = S_{(200)}$, $S_{(011)} = S_{(110)} = S_{(101)}$, $S_{(022)} = S_{(220)} = S_{(202)}$, $S_{(112)} = S_{(121)} = S_{(211)}$, $S_{(122)} = S_{(221)} = S_{(212)}$, and $S_{(012)} = S_{(120)} = S_{(201)} = S_{(102)} = S_{(021)} = S_{(210)}$. Therefore, $\overline{S}$ can be computed about 26/9=2.88 times faster since we may consider calculating the 9 different latency factors (listed above), instead of computing all 3^3-1 message latencies corresponding to the 26 different destination nodes, as $\overline{S} = [S_{(111)} + S_{(222)} + 3S_{(001)} + 3S_{(002)} + 3S_{(011)} + 3S_{(022)} + 3S_{(112)} + 3S_{(122)} + 6S_{(012)}]/26$. Such equivalent latency factors would appear even more when the network size increases.

3.3 Model validation

The analytical model has been validated through a discrete-event simulator that mimics the behaviour of Duato's fully adaptive routing at the flit level in *k*-ary *n*-cubes. In each simulation experiment, a total number of 100000 messages is delivered. Statistics gathering was inhibited for the first 10000 messages to avoid distortions due to the initial startup conditions. The simulator uses the same assumptions as the analysis, and some of these assumptions are detailed here with a view to making the network operation clearer. The network cycle time is defined as the transmission time of a single flit from one router to the next. Messages are generated at each node according to a Poisson process with a mean inter-arrival rate of λ_g messages/cycle. Message length is fixed at M flits. Destination nodes are determined using a uniform random number generator. The mean message latency is defined as the mean amount of time from the generation of a message until the last data flit reaches the local PE at the destination node. The other measures include the mean network latency, the time taken to cross the network, and the mean queueing time at the source node, the time spent at the local queue before entering 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, Figures 3.2, 3.3 and 3.4 depict latency results predicted by the above models plotted against those provided by the simulator for a 8-ary 2-cube ($N = 8^2$), 8-ary 3-cube ($N = 8^3$), and 8-dimensional hypercube ($N = 2^8$), respectively, and for different message lengths, M=32, 64 and 100 flits. Moreover, the number of virtual channels per physical channel was set to V=3 and 5.

The horizontal axis in the figures shows the traffic generation rate at each node (λ_g) while the vertical axis shows the mean message latency. The figures reveal that in all cases, the analytical model predicts the mean message latency with a good degree of accuracy in the steady state regions. Moreover, the model predictions are still good even when the network operates in the heavy traffic region and when it starts to approach the saturation region. However, some discrepancies around the saturation point are apparent. These can be accounted for by the approximation made to estimate the variance of the service time distribution at a channel. This approximation greatly simplifies the model as it allows us to avoid computing the exact distribution of the message service time at a given channel, which is not a straightforward task due to the interdependencies between service times at successive channels (since wormhole routing relies on a blocking mechanism for flow control). However, the main advantage of the proposed model is its simplicity which makes it a practical evaluation tool for assessing the performance behaviour of fully adaptive routing in k-ary n-cubes.

3.4 Extension of the model

This section outlines briefly the modifications that have to be made in order to extend the

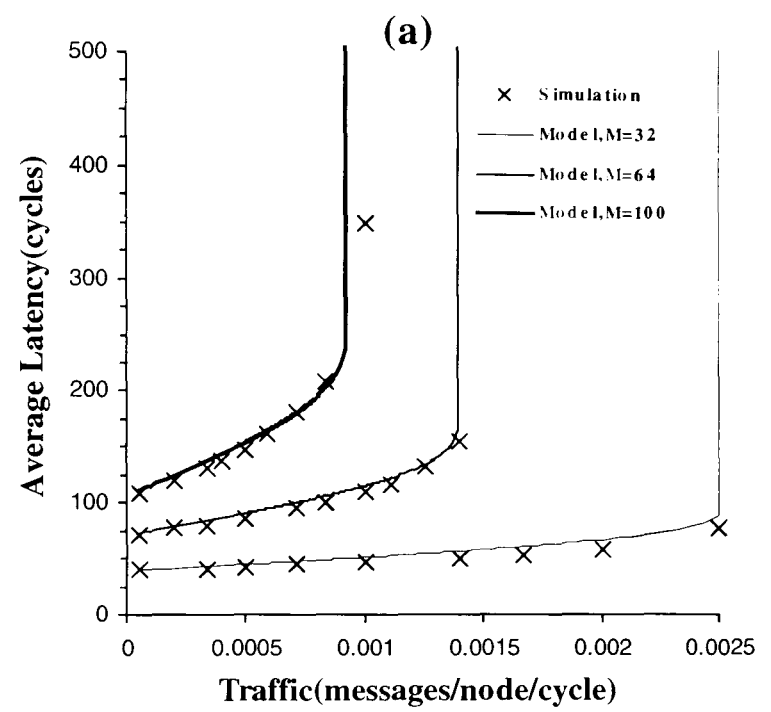
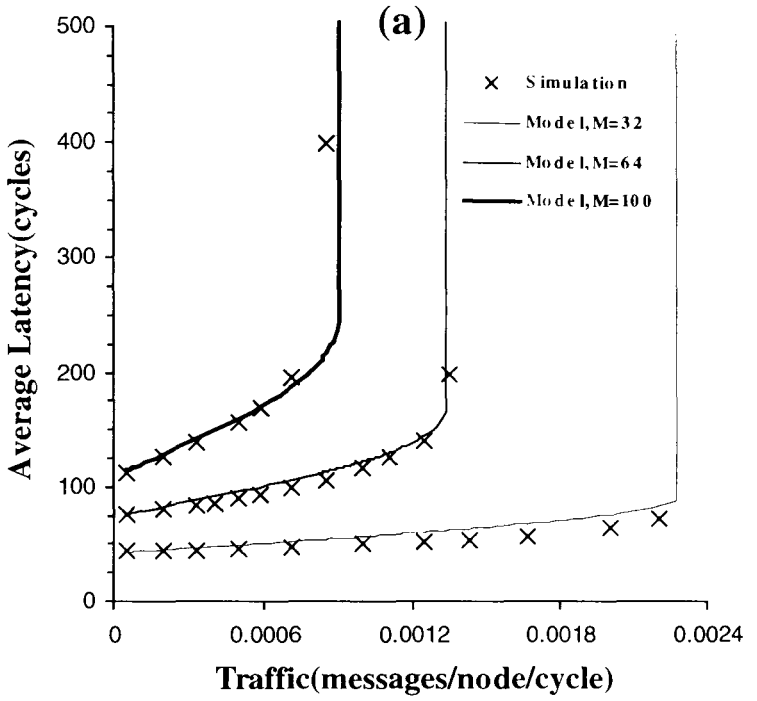


Figure 3.2: The average message latency predicted by the model against simulation results for an 8-ary 2-cube with message length M=32, 64 and 100 flits and (a) V=3 and (b) V=5 virtual channels per physical channel.



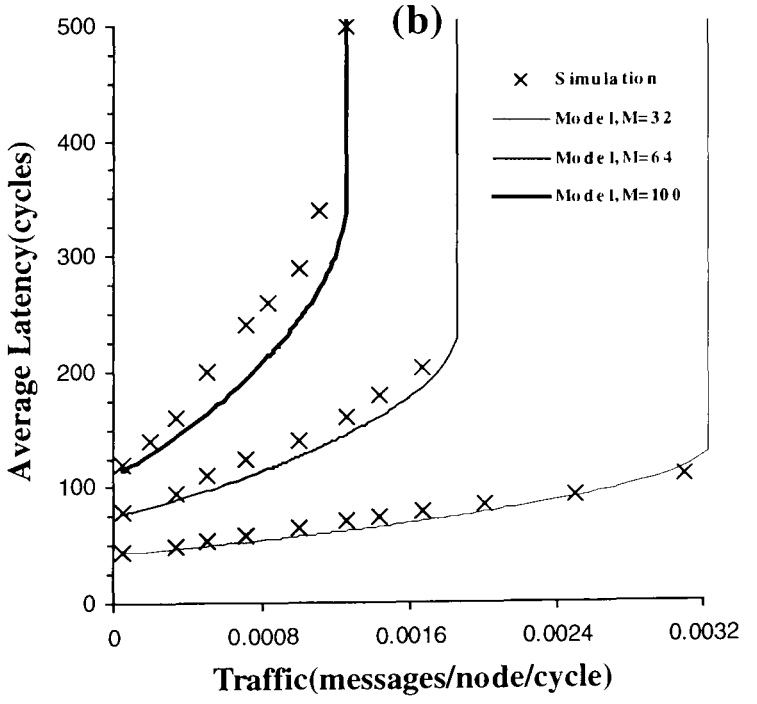
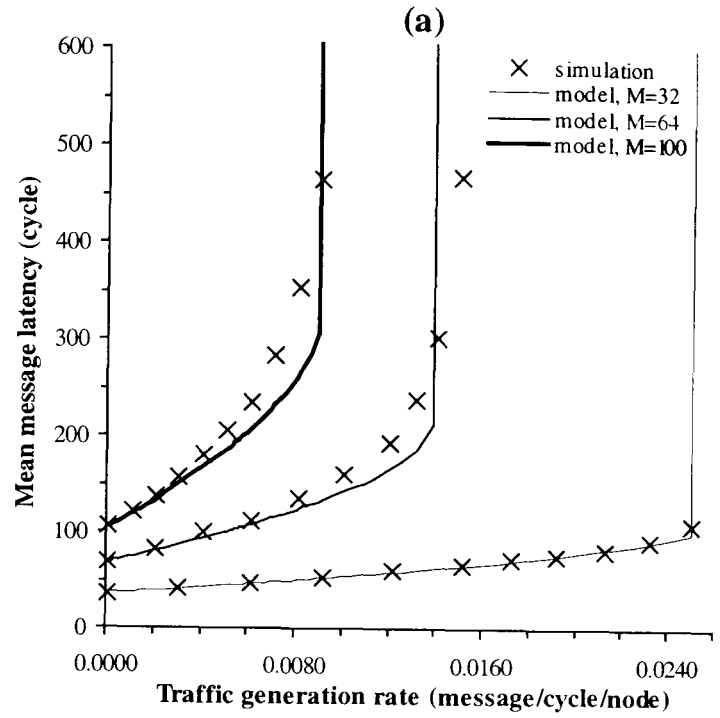


Figure 3.3: The average message Latency predicted by the model against simulation results for an 8-ary 3-cube with message length M=32, 64 and 100 flits and (a) V=3 and (b) V=5 virtual channels per physical channel.



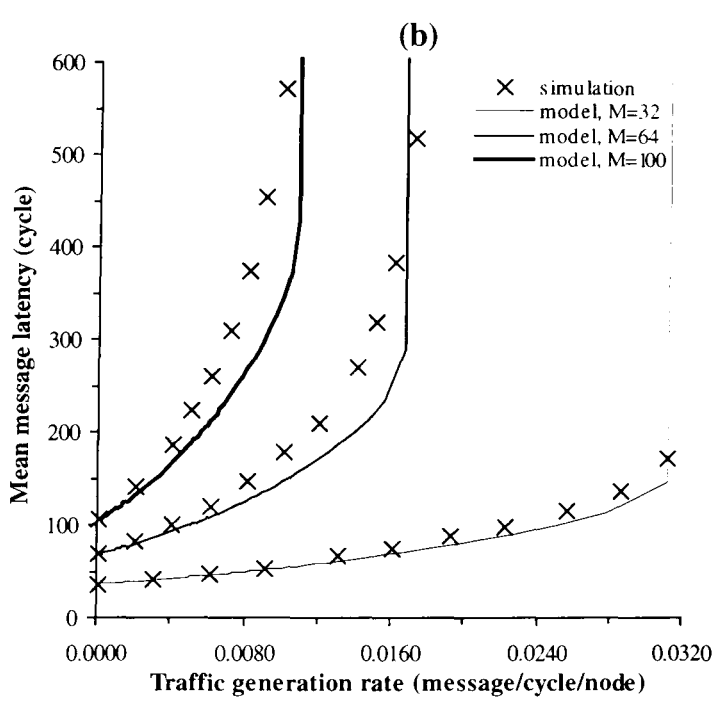


Figure 3.4: The average message Latency predicted by the model against simulation results for an 8-dimensional hypercube with message length M=32, 64 and 100 flits and (a) V=3 and (b) V=5 virtual channels per physical channel.

above model for bidirectional k-ary n-cubes and for capturing the effects of communication locality.

3.4.1 The model for the bidirectional k-ary n-cubes

The average number of hops that a message makes across one dimension, $\bar{k}$, given by Equations 3.1, should be changed to [6]

$$\bar{k} = \begin{cases} \dfrac{k}{4}, & \text{if } n \text{ is even} \\ \dfrac{1}{4}(k - \dfrac{1}{k}), & \text{if } n \text{ is odd} \end{cases} . \tag{3.29}$$

Since a router in the k-ary n-cube has $2n$ output channels, the rate of messages received by each channel, λ_c, is now given by [6]

$$\lambda_c = \frac{\lambda_g \bar{d}}{2n} . \tag{3.30}$$

Let set $H = \{h_i\}$, $(1 \le i \le n)$, denote the number of hops that the message makes along each dimension when it traverses the network from the source node $S = s_1 s_2 \cdots s_n$ to the destination node $D = d_1 d_2 \cdots d_n$ as discussed above for the unidirectional case. We have

$$h_i = \min(h_i^+, h_i^-), \tag{3.31}$$

where h_i^+ and h_i^- are distance between the source and destination nodes in i-th dimension using channels in, respectively, increasing and decreasing directions. These numbers, h_i^+ and h_i^-, are the smallest integer numbers satisfying the following equations

$$(s_i + h_i^+) \bmod k = d_i , \tag{3.32}$$

$$(s_i - h_i^{-}) \bmod k = d_i . \tag{3.33}$$

3.4.2 Capturing the effects of communication locality

Locality has an important impact on network performance [96]. Thus, deriving an analytical model to study the effect of locality on the overall performance of a network would be very beneficial. The above modelling approach can be easily extended to account for the case when traffic contains communication locality. Using a simple locality model proposed by Agarwal [6] allows the probability of blocking to be determined in a similar way to that of the uniform traffic case. Let f denote the locality factor, which is the fraction of nodes that are potential candidates to receive a message from a source node. Moreover, for a given source node, destination node is chosen randomly among the nodes in an n-dimensional sub-cube with fN nodes centred at the source. Note that f=1 refers to a pure uniform traffic without locality. For a given fraction of locality, f, destination nodes for messages originating at a source node with address $x = (x_1 x_2 \cdots x_n)$, $0 \le x_i \le k-1$, are randomly chosen from the set of nodes $y = (y_1 y_2 \cdots y_n)$ where $x_i \le y_i \le x_i + \sqrt[n]{fN} - 1$ (modulo k). With such a locality model, $\overline{k}$ given by Equation 3.2 should be replaced in the model by $\overline{k_f}$ given by

$$\overline{k_f} = \frac{\sqrt[n]{fN} - 1}{2} . \tag{3.34}$$

The destination node $D = d_1 d_2 \cdots d_n$ for a message generated at a given source node $S = s_1 s_2 \cdots s_n$, can be any node in the set G_f - S where

$$G_f = \{ X = x_1 x_2 \cdots x_n \mid (s_i + h_i)_{\bmod k} = x_i,\ 0 \le h_i \le \overline{k_f} - 1 \ \text{ for } 1 \le i \le n \} . \tag{3.35}$$

The vector H for each destination node D is now made up from new h_is, $(1 \le i \le n)$, derived to meet the condition used in Equations 3.32 and 3.33. Since the number of

possible destination nodes D for the given source node is $fN-1$ (the number of elements in G_f excluding the source node), Equation 3.19 should change to

$$\bar{S} = \frac{1}{fN-1} \sum_{D \in G_f - S} S_H \,. \tag{3.36}$$

3.5 Performance analysis

The proposed analytical model is used to study the performance merits of the k-ary n-cube with adaptive routing and virtual channels. In this section, the 10-ary 3-cube is often used for the sake of a concrete example, but the conclusions reached here have been found to be similar when other network configurations are considered.

Figure 3.5 shows, for the case of a unidirectional 10-ary 3-cube with message length M=50 flits and V=4 virtual channels, as a function of offered traffic by each node, the main components of the message latency, *Latency*: average network latency, $\bar{S}$; average waiting time at a source node to inject a message into the network, W_s; and average degree of virtual channels multiplexing, $\bar{V}$. As can be seen from the figure, the waiting time at the source and average degree of virtual channels multiplexing grow almost linearly when the offered traffic increases, whereas the average network latency remains almost fixed when the network has not approached the heavy traffic region. The network latency starts to increase rapidly when the traffic is around $\lambda_g = 0.0015$.

Figure 3.6 illustrates latency results against traffic generation rate when the message length is M = 50 flits for different numbers of virtual channels. The results show that increasing the number of virtual channels improves network performance especially when the increase in the number of virtual channels is relatively considerable (compared to the number of existing virtual channels). That is, the performance improvement from V=3 to

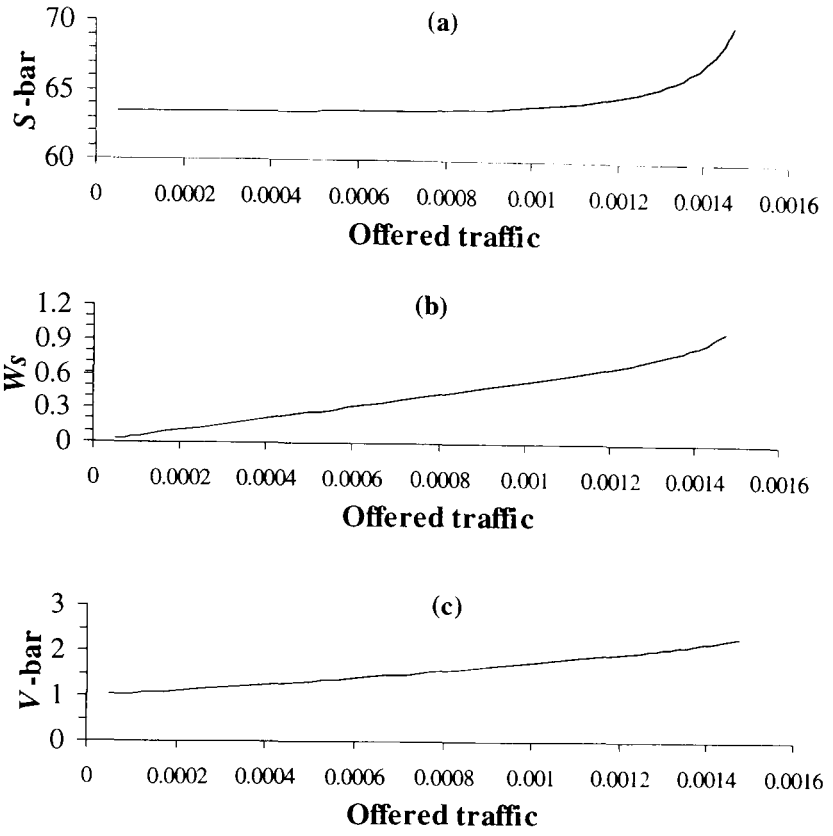


Figure 3.5: Main components making the average message latency, (a) $\overline{S}$, (b) W_s, and (c) $\overline{V}$, versus traffic generation rate in a unidirectional 10-ary 3-cube with message length M=50 flits and V=4 virtual channels per physical channel.

V=4 (with a 4/3 increase ratio) is much more noticeable than that from V=6 to V=7 (with a 7/6 ratio). As we add more and more virtual channels, the achieved performance improvement is reduced, since the network approaches the actual limits imposed by the total physical bandwidth of its channels. This is better shown in Figure 3.7, which reveals the effects of the number of virtual channels on network performance by plotting the offered traffic when the network is saturated (saturation traffic) against the number of virtual channels. The network enters the saturation region when $\rho \geq 1$ (given by Equation 3.17); the corresponding λ_g for which the condition $\rho \geq 1$ is satisfied is the saturation traffic rate. As can be seen from the figure, increasing the number of virtual channels increases the saturation traffic rate. However, the increase of the saturation traffic rate

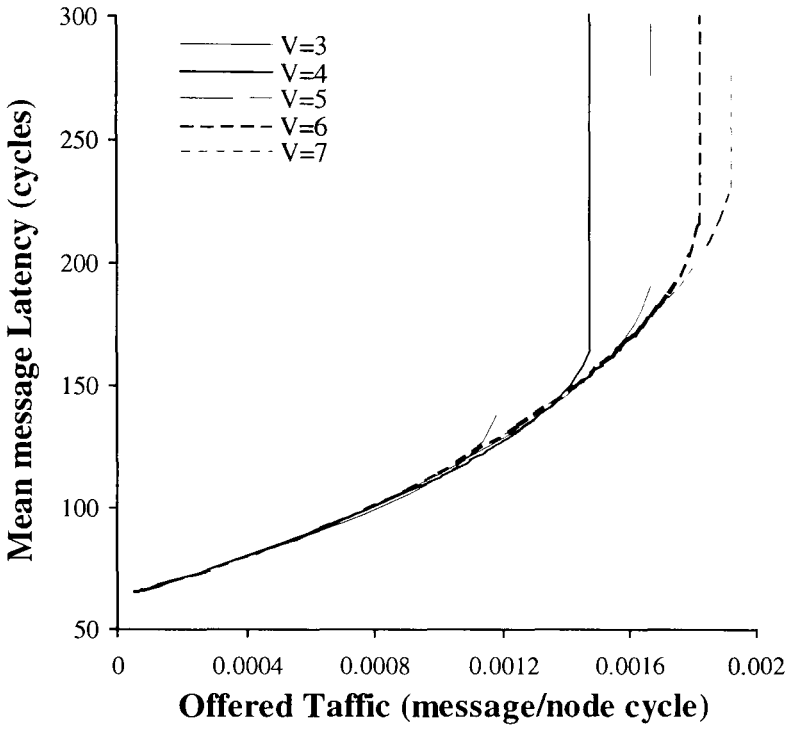


Figure 3.6: The average message latency versus traffic generation rate in a unidirectional 10-ary 3-cube with message length M=50 flits and V=3, 4, 5, 6, and 7 virtual channels per physical channel.

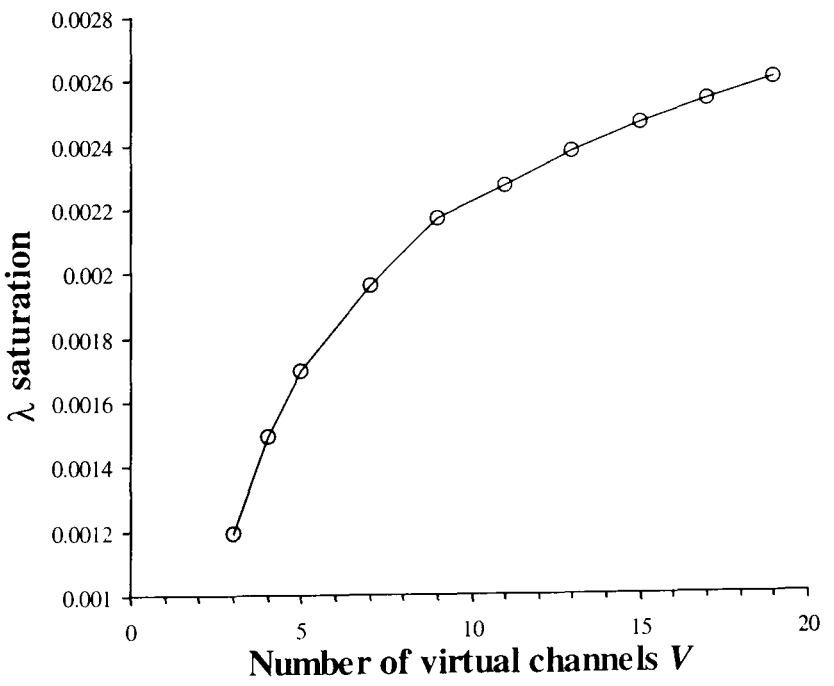


Figure 3.7: The saturation traffic rate versus the number of virtual channels per phsical channel in a unidirectional 10-ary 3-cube with message length M=50 flits.

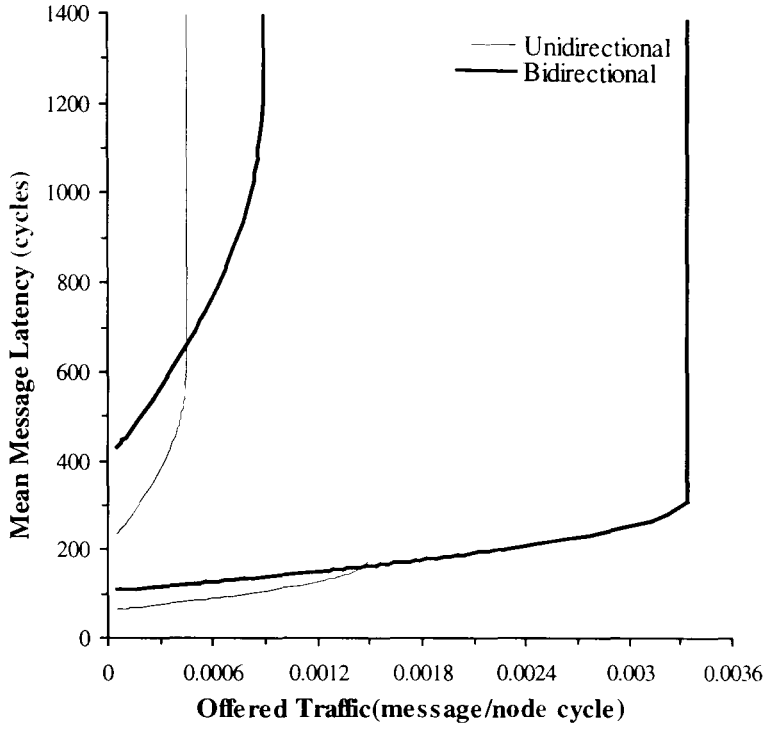


Figure 3.8: The average message latency versus traffic generation rate in a unidirectional and bidirectional 10-ary 3-cube with message length M=50 and 200 flits and V=4 virtual channels per physical channel.

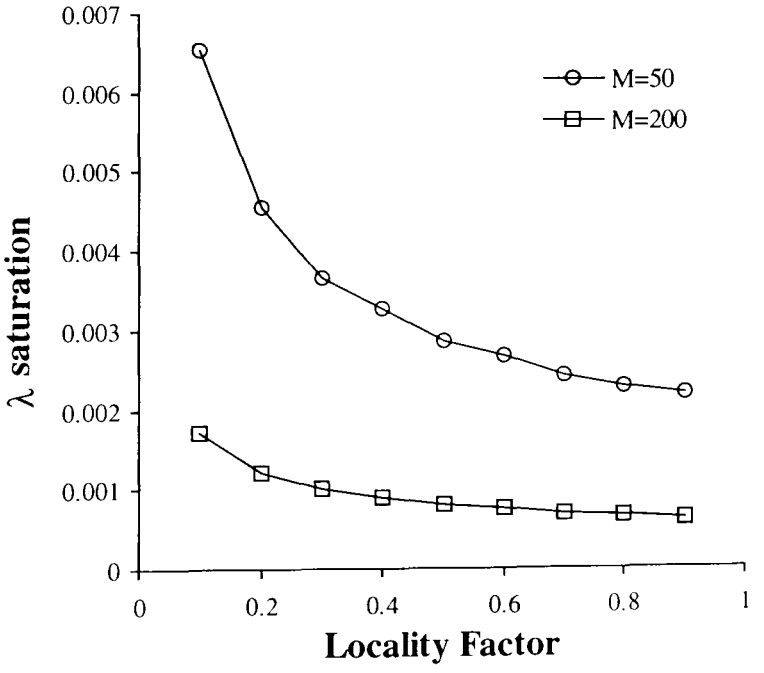


Figure 3.9: The saturation traffic rate versus the traffic locality factor in a unidirectional 10-ary 3-cube with message length M=50 and 200 flits and V=4 virtual channels per physical channel.

becomes smaller when the number of virtual channels becomes larger. This continues until an increase in the number of virtual channels does not change the saturation traffic rate noticeably. Such a limit saturation traffic rate is indeed the physical bandwidth limit of the network. The convergence of the saturation traffic rate to such a limit, when the number of virtual channels increases, can be easily observed in Figure 3.7.

Figure 3.8 shows the mean message latency curves for the unidirectional and bidirectional 10-ary 3-cube when the message length is $M = 50$ and 200 flits and $V = 4$ virtual channels per physical channel. The bidirectional k-ary n-cube has double the bisection width and node pin-out than its unidirectional equivalent. In order to have a fair and realistic comparison, the bisection width or pin-out constraint was held constant in the two networks. So, if we use the unidirectional network as a basis for the comparison (with a channel width equals the flit size), the channel width in the bidirectional network will be half of the flit size, i.e. two channel cycles are required for each flit transmission over a channel in the bidirectional network. The figure reveals that the bidirectional k-ary n-cube outperforms its unidirectional counterpart under the constant bisection width and pin-out constraints. This is because the former network has a lower message distance and lower message traffic rate on its channels which compensate for the lower channel bandwidth.

Figure 3.9 depicts the saturation rates in the unidirectional 10-ary 3-cube when the message length is M=50 and 200 flits and V=4 virtual channels per physical channel as a function of the locality factor f. The results show that the network saturates sooner as the locality factor increases although the performance degradation is faster for longer messages. This is because the larger locality factor f imposes higher traffic rates over network channels. Longer messages increase their service times at a network channel, resulting in longer blocking times. The influence of longer messages on network performance is more noticeable when f increases, as the network suffers from both higher traffic rates on its channels and longer channel service times imposed by longer messages.

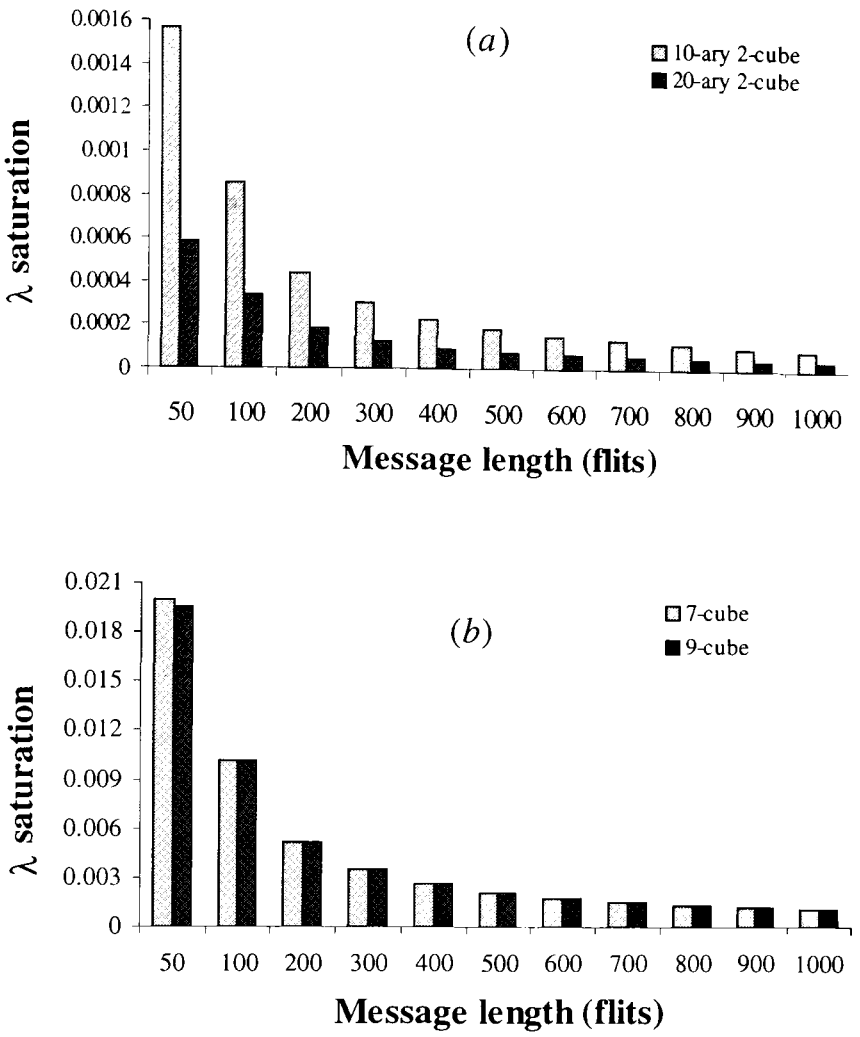


Figure 3.10: The saturation traffic rate versus message length M in (a) 10-ary 2-cube and 20-ary 2-cube, and in (b) 7-dimensional and 9-dimensional hypercube, with V=4 virtual channels per physical channel.

Finally, Figure 3.10 investigates the impact of message length on network performance (saturation rate) in the two well-known instances of k-ary n-cubes, namely the 2-dimensional torus and the hypercube. Figure 3.10(a) depicts the saturation traffic versus message length M in the unidirectional 10-ary 2-cube and 20-ary 2-cube, respectively, when V=4 virtual channels per physical channel. The results show that performance degrades when the network size increases from 100 nodes to 400 nodes. Figure 3.10(b) shows the same quantities for the 7- and 9-dimensional hypercubes where an increase

factor of four in network size has not had a noticeable change in the network saturation traffic. The results reveal an important characteristic of the torus and hypercube, and that is the torus (low dimensional k-ary n-cube) is not truly scalable as its performance depends strictly on its size, while the hypercube (high-dimensional k-ary n-cube) is much more scalable as the total network bandwidth, as a function of the total number of channels, increases with network size.

3.6 Conclusions

In this chapter an analytical model to compute the mean message latency in k-ary n-cubes with fully adaptive wormhole routing was proposed. Simulation experiments have revealed that the latency results predicted by the analytical model are in good agreement with those obtained through simulation. The proposed model achieves a good degree of accuracy under different operating conditions because it computes the exact expression for the probability of message blocking at a given router. It uses M/G/1 queuing theory when calculating the message blocking time at each channel. Furthermore, it achieves this high degree of accuracy while maintaining simplicity, making it a practical evaluation tool that can be used to gain insight into the performance behavior of fully adaptive routing in wormhole-routed k-ary n-cubes. The model was also extended for bidirectional k-ary n-cubes and for traffic patterns that exhibit communication locality.

The analysis using the proposed model shows that the number of virtual channels and the traffic locality have a great impact on network performance. A preliminary comparison between unidirectional and bidirectional k-ary n-cubes, under both constant bisection bandwidth and pin-out constraints, reveals that bidirectional k-ary n-cube outperform their unidirectional counterparts. It is also shown that higher-dimensional k-ary n-cube networks such as hypercubes, are more scalable than their lower-dimensional counterparts such as tori, because, in the former, total network bandwidth scales better with network

size.

Future work in this area may extend the proposed analytical model for other common multicomputer network topologies such as n-dimensional meshes, which are variations of k-ary n-cubes without wrap-around connections. Developing a model for meshes is more complicated than for k-ary n-cubes because traffic rates and service times have to be computed at each network channel; these differ from one channel to the next due to the inherent asymmetry of the topologies.

Existing studies [29, 67, 158, 162, 174] have so far relied on software simulation to examine the performance of adaptive routing under other traffic patterns, such as those generated by the presence of hotspots in the network and matrix transpose communication. In the next chapters, we propose new analytical models that deal with important non-uniform traffic patterns such as hotspots and permutation traffic patterns.

Chapter 4

An Analytical Model of Adaptive Wormhole Routing in the Presence of Hotspot Traffic

Several analytical models of fully adaptive routing have recently been proposed for wormhole-routed k-ary n-cubes under the uniform traffic pattern. Recent studies [59, 74, 158] have revealed that the performance advantages of adaptive routing over deterministic routing are more noticeable when traffic is non-uniform due, for example, to the presence of hotspots in the network [149]. Hotspots arise when a number of nodes direct a fraction of their generated messages to a single destination node. For instance, global synchronisation where each node in the system sends a synchronisation message to a distinguished node is a typical situation that can produce hotspots [88, 149, 158]. The reduction operation [59], where a node receives messages from all other nodes, is another example that may generate hotspots. In distributed shared-memory (DSM) multicomputers, the traffic generated by cache coherency protocol, mainly composed of *invalidation* and *acknowledgement* messages, is also likely to create hotspots [88].

To the best of our knowledge, no study has been reported in the literature for modelling hotspots in wormhole-routed k-ary n-cubes, and consequently most studies have resorted to simulation to evaluate the performance benefits of adaptive routing in these networks. This chapter proposes the first analytical model for computing the mean message latency in k-ary n-cubes with fully adaptive routing in the presence of hotspot traffic. The model is developed for Duato's fully adaptive routing algorithm [63], but the modelling approach can equally use the other routing algorithms described in [115, 174].

4.1 The analytical model

This study assumes that there is only a single hotspot node in the network. The main reason behind this restriction is to keep the notation used for describing the model at a manageable level. Our modeling approach, however, can be extended to deal with the case of multiple hotspots with some effort.

4.1.1 Notation and assumptions

The list of symbols used to describe the model is shown in Table 4.1. Moreover, the model uses the following assumptions that are widely employed in the literature [3-6, 12, 32-34, 42, 43, 48, 49, 58, 81, 84-86, 91, 120, 142, 143].

a) The traffic model proposed by Pfister and Norton [149] is used to generate hotspot traffic. In this model, each generated message has a finite probability h of being directed to the hotspot node, and probability $(1-h)$ of being directed to the other network nodes. Let us refer to these two types of messages as hotspot and regular messages respectively.

b) Nodes generate traffic independently of each other, following a Poisson process with a mean rate of λ_g messages/cycle consisting of regular and hotspot portions

Table 4.1: Notation used in the model for hotspot traffic.

Notation	Description
B_r	mean blocking time seen by a regular message at a given channel
$B_{h_{m,j}}$	mean blocking time seen by a j-hop hotspot message at its m-*th* hop channel
$\overline{d}$	average number of hops that a regular message makes across the network.
h	fraction of hotspot messages generated by each node
J	set of channels that are j hops away from the hotspot node
$\|J\|$	number of elements in the set J
k	network radix
$\overline{k}$	average number of hops that a regular message makes in a dimension
n	network dimension
n_i	number of nodes located i hops away from a given node
N	network size ($N=k^n$)
Q_{v_j}	intermediate variable used for calculating P_{v_j}
P_{h_j}	probability that a message crosses a channel located j hop away from the hotspot
P_{a_j}	probability that all adaptive virtual channels at a physical channel located j hops away from hotspot node are busy
$P_{a\&d_j}$	probability that all adaptive and deterministic virtual channels at a channel located j hops away from hotspot are busy
P_{v_j}	probability that v virtual channels at a channel located j hops away from hotspot node are busy
$P_{t,l}$	probability of termination for a message that makes, on average, l in each dimension
S_j	network latency of a j-hop hotspot message
$\overline{S_{h_j}}$	latency seen by a hotspot message to cross from a channel located j hops away from the hotspot node to its destination (i.e., the hotspot node)
$\overline{S}$	mean network latency
S_{h_j}	network latency of a j-hop hotspot message
$\overline{S_r} / \overline{S_h}$	mean network latency of a regular/hotspot message
S_{s_j}	message latency for a source node located j hops away from hotspot.
$\overline{V}$	average multiplexing degree of the virtual channels at a given physical channel.
w_j	mean waiting time seen by a message at a channel j hops away from the hotspot node
W_{s_j}	mean waiting time at a source node located j hops away from the hotspot node
$\overline{W_s}$	mean waiting time in a given source node
$\Re_{v_j}$	state in the Markov chain denoting that v virtual channels are busy at a channel located j hops away from the hotspot
ξ_j	ratio of the number of nodes which are j hops away from the hotspot node to the number of nodes that are i, $j \le i \le d_{\max}$, hops away from hotspot.
λ_g	message generation rate at a node

Table 4.1: (continued).

Notation	Description
λ_{h_j}	traffic rate of hotspot messages on a channel that is j hops away from the hotspot node
λ_j	total traffic rate on a channel that is j hops away from the hotspot node
λ_r	traffic rate of regular messages on a channel
σ_j	probability of generating a j-hop message
$\varphi_{j,l}$	probability of blocking for a message that makes, on average, l hops per dimension
$\pi_{\beta,l}$	probability that β dimensions have been crossed by a message that makes, on average, l per dimension

of $h\lambda_g$ and $(1-h)\lambda_g$ respectively.

c) Message length is fixed and equal to M flits, each of which is transmitted in one cycle between two adjacent nodes.

d) The local queue at the injection channel in the source node has infinite capacity. Moreover, messages are transferred to the local PE as soon as they arrive at their destinations through the ejection channel.

e) V virtual channels are used per physical channel, as shown in the router structure illustrated in Figure 2.2(a), which are divided into two classes VC_1 and VC_2. According to the adaptive routing algorithm, described in Section 2.2.2, Class VC_2 contains $(V-2)$ virtual channels that are crossed adaptively. On the other hand, class VC_1 contains two virtual channels that are crossed deterministically. Let the virtual channels belonging to class VC_2 and VC_1 be called the *adaptive* and *deterministic* virtual channels, respectively. When there is more than one adaptive virtual channel available a message chooses one at random. To simplify the model derivation no distinction is made between the deterministic and

adaptive virtual channels when computing virtual channel occupancy probabilities [142].

4.1.2 The outline of the model

The mean message latency is composed of the mean network latency, $\overline{S}$, that is the time to cross the network, and the mean waiting time seen by a message in the source node, $\overline{W_s}$. However, to capture the effects of virtual channel multiplexing, the mean message latency has to be scaled by a factor, $\overline{V}$, representing the average degree of virtual channels multiplexing, that takes place at a given physical channel. Therefore, the mean message latency can be written as

$$Latency = (\overline{S} + \overline{W_s})\overline{V} . \tag{4.1}$$

The regular and hotspot messages see different network latencies as they cross different numbers of channels to reach their destinations. If $\overline{S_r}$ and $\overline{S_h}$ denote the mean network latency for regular and hotspot messages respectively, the mean network latency taking into account both types of messages is given by

$$\overline{S} = (1-h)\overline{S_r} + h\overline{S_h} . \tag{4.2}$$

Let us now determine the quantities $\overline{S_r}$, $\overline{S_h}$, $\overline{S}$, $\overline{W_s}$ and $\overline{V}$.

4.1.2.1 Calculation of the mean network latencies $\overline{S_r}$ and $\overline{S_h}$

The average number of hops that a regular message makes across one dimension, $\overline{k}$, and across the network, $\overline{d}$, are given by [6]

$$\overline{k} = \frac{k-1}{2}, \tag{4.3}$$

$$\overline{d} = n\overline{k}. \tag{4.4}$$

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

$$\lambda_r = \frac{(1-h)\lambda_g \overline{d}}{n}. \tag{4.5}$$

The network latency for a message consists of two parts: one is the delay due to the actual message transmission time, and the other is due to the blocking time in the network. As we shall see below, the mean blocking time experienced by a regular message at a given channel is the same across all the channels along its path. Since a regular message makes, on average, $\overline{d}$ hops to reach its destination the mean network latency, $\overline{S_r}$, of a regular message can therefore be written as

$$\overline{S_r} = M + \overline{d} + \overline{d}B_r, \tag{4.6}$$

where M is the message length and B_r is the mean blocking time seen by a regular message at a channel.

The hotspot traffic is not uniformly distributed across the network channels as channels located nearer to the hotspot node receive higher traffic rates than those further away. Consider a channel that is j hops away from the hotspot node, and let P_{h_j} be the probability that a hotspot message uses this channel to reach its destination, which is the hotspot node. Given that each of the N nodes generates, on average, $h\lambda_g$ hotspot messages

in a cycle, the rate of hotspot traffic received by the channel, λ_{h_j}, is simply given by

$$\lambda_{h_j} = hN\lambda_g P_{h_j}. \tag{4.7}$$

The number of nodes located j hops away from a given node (here the hotspot node) is given by Theorem 2.2 as

$$n_i = \vec{A}_n^k(i) = \sum_{l=0}^{n} (-1)^l \binom{n}{l} \binom{i-lk+n-1}{n-1}. \tag{4.8}$$

To calculate P_{h_j}, we firstly have to derive an expression for the number of channels located j hops away from a given node in the unidirectional k-ary n-cube.

THEOREM 4.1. The number of channels that are j hops away from a given node in the unidirectional k-ary n-cube is

$$C_j = \sum_{l=0}^{n-1} \sum_{t=0}^{n-l} (-1)^t (n-l) \binom{n}{l} \binom{n-l}{t} \binom{j-t(k-1)-1}{n-l-1}. \tag{4.9}$$

PROOF. Let the given node be the one with address $X = (x_1, x_2, \cdots, x_n)$. It is easy to see that all the n output channels of the nodes $Y_j = (y_1, y_2, \cdots, y_n)$ located j hops away from the given node X should be counted except those nodes whose addresses include at least one digit y_i, $1 \le i \le n$, which is equal to the corresponding digit in the given node address pattern, i.e. $x_i = y_i$, $1 \le i \le n$. If $x_i = y_i$ the output channel at dimension i of node Y_j should not be considered when counting. Node may have 0, 1, 2, ..., or n-1 such uncountable channels depending to its address pattern. The number of nodes having no such a channel is simply $\Gamma_1^{k-1}(j,n)$ (given by Proposition 2.1), since $\Gamma_1^{k-1}(j,n)$ gives the number of nodes located j hops away from the given node, each at least at distance 1 from the given node in every dimension. The number of nodes having one channel to be

excluded for counting is $\binom{n}{1}\Gamma_1^{k-1}(j,n-1)$, which is the number of nodes located j hops away from the given node whose distance from the given node in exactly one dimension is zero. Generally, the number of nodes which have l channels to be omitted when counting the number of channels j hops away from the given node is $\binom{n}{l}\Gamma_1^{k-1}(j,n-l)$. Summing up all the channels to be counted results in the total number of channels which are j hops away from the given node, C_j, as

$$C_j = \sum_{l=0}^{n-1}(n-l)\binom{n}{l}\Gamma_1^{k-1}(j,n-l) = \sum_{l=0}^{n-1}\sum_{t=0}^{n-l}(-1)^t(n-l)\binom{n}{l}\binom{n-l}{t}\binom{j-t(k-1)-1}{n-l-1}.$$

The probability, P_{h_j}, that a message has used, during its network journey, a particular channel located j hops away from the hotspot node, can be derived as follows. Consider all the channels located j hops away from the hotspot node. Theorem 4.1 gives the number of such channels to be C_j. Recalling that a k-ary n-cube has N nodes, the number of source nodes for which any of these channels (channels located j hops away from the hotspot node) can act as an intermediate channel to reach the hotspot node is $N-\sum_{l=0}^{j-1}n_l$. Therefore, P_{h_j} can be written as

$$P_{h_j} = \frac{N-\sum_{l=0}^{j-1}n_l}{C_j N} = \frac{\sum_{l=j}^{d_{\max}}n_l}{C_j N}, \tag{4.10}$$

where $d_{\max}$ is the maximum number of hops that a message may take to reach its destination (also called network diameter) and is given by

$$d_{\max} = n(k-1). \tag{4.11}$$

Unlike its regular counterpart, a hotspot message encounters different blocking times at different channels due to the non-uniform traffic rates on network channels caused by the

hotspot traffic. A hotspot message may visit j=1, 2, ..., or $d_{\max}$ channels to reach the hotspot node. All these cases have to be taken into account when computing the mean network latency for hotspot messages. The network latency seen by a j-hop hotspot message is given by

$$S_{h_j} = M + j + \sum_{m=1}^{j} B_{h_{m,j}}, \tag{4.12}$$

where $B_{h_{m,j}}$, $1 \le j \le d_{\max}$, is the blocking time of a j-hop hotspot message at its m-th hop channel. Let θ_j be the probability that a randomly chosen node is j-hops away from a given node as destination (here the hotspot node). The number of nodes that are j hops away from a given node can be obtained using Equation 4.8. Dividing this over the total number of nodes (excluding the given node) yields θ_j as

$$\theta_j = \frac{n_j}{N-1}. \tag{4.13}$$

Hence, the average network latency seen by a hotspot message, $\overline{S_h}$, can be expressed by

$$\overline{S_h} = \sum_{j=1}^{d_{\max}} \theta_j S_{h_j}. \tag{4.14}$$

4.1.2.2 Calculation of the blocking times B_r and $B_{h_{m,j}}$

A regular or hotspot message is blocked at a given physical channel when all the adaptive virtual channels of the remaining dimensions to be visited and also the deterministic virtual channels of the lowest dimension still to be visited are busy. The mean blocking time depends on the probability of blocking at a physical channel and on the mean waiting time to access a virtual channel. The probability of blocking depends on the number of output channels, and thus on the virtual channels that a message can use at its next hop.

When blocking occurs, the mean waiting time depends on the location of the current channel relative to the hotspot node as the traffic rate varies from one channel to the next.

A regular message makes, on average, $\bar{d}$ hops to cross the network. Suppose that the message has reached the m^{th}-hop $(1 \le m \le \bar{d})$ channel along its path. This channel can be between 1 and $d_{\max}$ hops away from the hotspot node. Let $\varphi_{j,l}$ denote the probability of blocking for a message which makes, on average, l hops per dimension—in case of a regular message $l = \bar{k}$ —when the current channel is j hops, $1 \le j \le d_{\max}$, away from the hotspot node. Moreover, let w_j denote the mean waiting time when blocking occurs at the channel. Since there are C_j channels (given by Theorem 4.1) that are j hops away from the hotspot node, out of the total number of channels in the network nN, the mean blocking time, B_r , for a regular message can be written as

$$B_r = \sum_{j=1}^{d_{\max}} \frac{C_j}{nN} \varphi_{j,\bar{k}} w_j \,. \tag{4.15}$$

The probability of blocking $\varphi_{j,\bar{k}}$ is computed as follows. Since a regular message makes, on average, $\bar{k}$ hops per dimension, the probability of termination, $P_{t,\bar{k}}$, which is the probability that a message has crossed all the channels of a given dimension when it reaches a given router, is therefore given by [48]

$$P_{t,\bar{k}} = \frac{1}{\bar{k}} \,. \tag{4.16}$$

Hence, the probability that a message still has to visit, say, β out of the n dimensions, $\pi_{\beta,\bar{k}}$, can be written as

$$\pi_{\beta,\bar{k}} = \binom{n}{\beta} \left(1 - P_{t,\bar{k}}\right)^{\beta} P_{t,\bar{k}}^{n-\beta} \tag{4.17}$$

With β dimensions still to be visited, a regular message can select any one of the $(V-2)$ adaptive virtual channels belonging to the $(\beta-1)$ dimensions still to be visited adaptively; it can also use one of the two deterministic channels and any one of the $(V-2)$ adaptive virtual channels at the lowest dimension. Using Equation 4.17 we can write $\varphi_{j,\bar{k}}$ as

$$\varphi_{j,\bar{k}} = \sum_{\beta=1}^{n} \pi_{\beta,\bar{k}} P_{a_j}^{\beta-1} P_{a\&d_j}, \tag{4.18}$$

with P_{a_j} being the probability that all adaptive virtual channels of a physical channel located j hops away from the hotspot node are busy and $P_{a\&d_j}$ being the probability that all adaptive and deterministic virtual channels of that channel are busy.

Let P_{v_j}, $(0 \le v \le V)$, represent the probability that v virtual channels are busy at a physical channel located j hops away from the hotspot node. Using the same scheme for computing Equations 3.14 and 3.15 in Chapter 3, we can write

$$P_{a_j} = P_{V_j} + \frac{2P_{(V-1)_j}}{\binom{V}{V-1}} + \frac{P_{(V-2)_j}}{\binom{V}{V-2}}, \tag{4.19}$$

$$P_{a\&d_j} = P_{V_j} + \frac{2P_{(V-1)_j}}{\binom{V}{V-1}}. \tag{4.20}$$

To determine the mean waiting time, w_j, to acquire a virtual channel, a physical channel is treated as an M/G/1 queue with a mean waiting time of [104]

$$w_j = \frac{\rho_j S_j (1 + C_{S_j}^2)}{2(1-\rho_j)}, \tag{4.21}$$

$$\rho_j = \lambda_j S_j, \tag{4.22}$$

$$C_{S_j}^2 = \frac{\sigma_{S_j}^2}{S_j^2}, \tag{4.23}$$

where λ_j is the traffic rate on the channel located j hops away from the source node, S_j is its service time, and $\sigma_{S_j}^2$ is the variance of the service time distribution. The rate of messages arriving at the channel is composed of regular messages and hotspot messages, and is equal to

$$\lambda_j = \lambda_r + \lambda_{h_j}. \tag{4.24}$$

Let us assume for a moment that there is no hotspot traffic present in the network. Since adaptive routing distributes regular traffic evenly across the network channels, the mean service time for regular messages is the same across all channels and is equal to the mean network latency, $\overline{S_r}$ [36, 142]. The presence of hotspot traffic, however, causes the service time to vary from one channel to another due to the non-uniformity of traffic rates on the channels. When a message reaches a channel that is j hops away from the hotspot node, the mean service time considering both regular and hotspot messages can be written as

$$S_j = \frac{\lambda_r}{\lambda_j}\overline{S_r} + \frac{\lambda_{h_j}}{\lambda_j}\overline{S_{h_j}}, \tag{4.25}$$

where $\overline{S_{h_j}}$ is the mean latency seen by a hotspot message to cross from a channel located j hops away from the hotspot node to the hotspot node itself. The expression of $\overline{S_{h_j}}$ is given by

$$\overline{S_{h_j}} = M + \sum_{i=j}^{d_{\max}} \left(\xi_{i,j} \sum_{l=1}^{j-1} B_{h_{l+i-j,i}} \right), \tag{4.26}$$

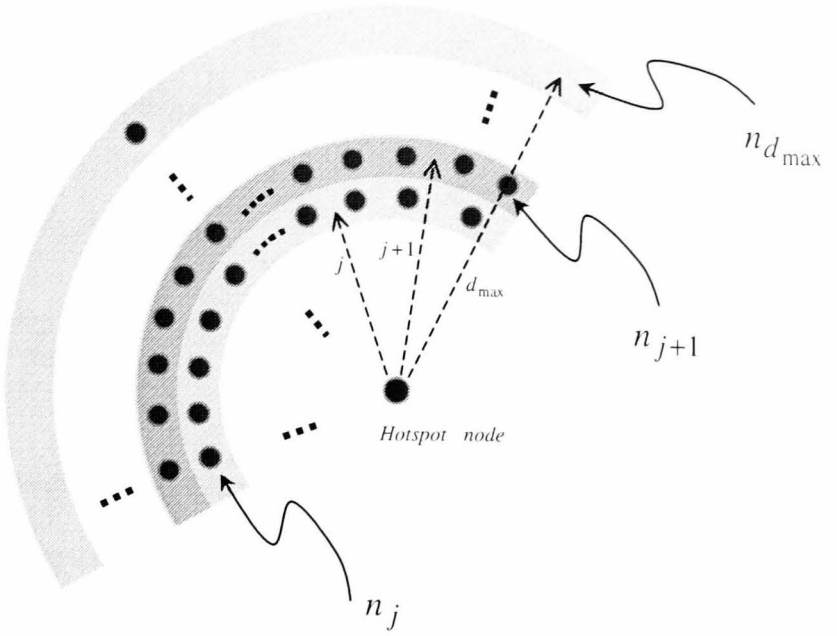


Figure 4.1: Number of nodes located j=1,2,..., d_{max} hops away from the hotspot node, n_j, in the k-ary n-cube.

with $\xi_{i,j}$ being the ratio of the number of nodes which are i, $j \le i \le d_{\max}$, hops away from the hotspot node to the number of nodes that are j hops or farther away from the hotspot node as shown in Figure 4.1. The number of nodes located j hops away from the hotspot (or number of nodes in band j as shown in Figure 4.1) is given by Equation 4.8. Summing up the number of nodes in bands j, j+1, ..., and d_{max} will give the number of nodes located j hops or further away from hotspot. Hence, we can write $\xi_{i,j}$ as

$$\xi_{i,j} = \frac{n_i}{\sum_{m=j}^{d_{\max}} n_m}. \tag{4.27}$$

Since the minimum service time at a channel is equal to the message length, M, following a suggestion proposed in [58], the variance of the service time distribution can be approximated as

$$\sigma_{S_j}^2 = (S_j - M)^2 . \tag{4.28}$$

As a result, the mean waiting time, given by Equation 4.21, becomes

$$w_j = \frac{\lambda_j S_j^2 (1 + \frac{(S_j - M)^2}{S_j^2})}{2(1 - \lambda_j S_j)} . \tag{4.29}$$

All the possible number of hops made by a hotspot message have to be considered when computing its mean blocking time. A hotspot message may make between 1 and d_{max} hops to reach its destination, the hotspot node. The possible number of hops made along a given dimension is between 0 and $k-1$. Equation 4.8 can be used to determine the number of ways to distribute the hops made by the hotspot message among the n dimensions. However, instead of considering all the possible combinations when distributing the hops among the n dimensions, we consider the average case where the message makes an equal number of hops per dimension. This greatly simplifies the computation of the mean blocking time especially when n and k are large due to the large number of combinations that has to be considered. Section 4.3 will reveal that our suggested approximation does not sacrifice the model's accuracy. Using this approximation, the above calculation of the mean blocking time for regular messages can be easily adapted for hotspot messages. When the message reaches the m^{th}-hop channel, it is (j-m+1) hops away from the hotspot node. Therefore, the mean blocking time can be written as

$$B_{h_{m,j}} = \varphi_{j-m+1,j/n} w_{j-m+1} . \tag{4.30}$$

The new expressions for the probability of blocking, $\varphi_{j-m+1,j/n}$, and mean waiting time, w_{j-m+1}, for hotspot messages can be obtained by simply substituting (j-m+1) for j and

j/n [1] for $\bar{k}$ in Equations 4.15-4.18.

Examining the above equations reveals that there are several inter-dependencies between the different variables of the model. For instance, Equation 4.25 and 4.26 reveal that S_j is a function of $B_{h_{m,j}}$ while equations 4.29 and 4.30 show that $B_{h_{m,j}}$ is a function of S_j. Given that closed-form solutions to such inter-dependencies are very difficult to determine, the different variables of the model are computed using iterative techniques for solving equations [104].

4.1.2.3 Calculation of the mean waiting time at the source $\overline{W_s}$

A regular message originating from a source node that is j hops away from the hotspot node sees a network latency of $\overline{S_r}$, whereas a hotspot message sees a latency of S_{h_j} to reach the hotspot node. Therefore, the mean network latency for a message that originates at a source node that is located j hops away from the hotspot node, S_{s_j}, taking into account both regular and hotspot messages with their appropriate weights, is simply

$$S_{s_j} = (1-h)\overline{S_r} + hS_{h_j}. \tag{4.31}$$

Modelling the local queue in the source node that is located j hops away from hotspot node as an M/G/1 queue, with the mean arrival rate λ_g/V (recall that a message in the source node can enter the network through any of the V virtual channels) and service time S_{s_j} with an approximated variance $(S_{s_j} - M)^2$, yields a mean waiting time of

$$W_{s_j} = \frac{\frac{\lambda}{V}S_{s_j}^2\left(1+\frac{(S_{s_j}-M)^2}{S_{s_j}^2}\right)}{2(1-\frac{\lambda}{V}S_{s_j})}. \tag{4.32}$$

[1] Min(1, *j*/*n*) should be used instead of *j*/*n* when *j*<*n*.

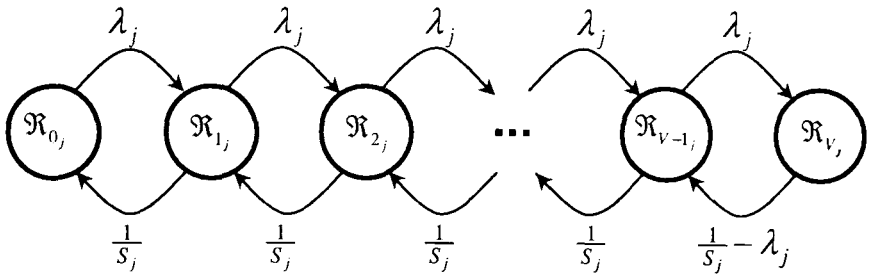


Figure 4.2: The Markov chain used for computing the probability of virtual channel occupancy of a physical channel located j hops away from hotspot.

Averaging over all possible values of j, $1 \le j \le d_{\max}$, gives the mean waiting time in a source node as

$$\overline{W_s} = \sum_{j=1}^{d_{\max}} \theta_j W_{s_j} . \tag{4.33}$$

4.1.2.4 Calculation of average virtual channels multiplexing degree $\overline{V}$

The probability, P_{v_j}, that v virtual channels are busy at a physical channel that is j hops away from the hotspot node, can be determined using a Markovian model shown in Figure 4.2. In this figure, state $\Re_{v_j}$ corresponds to v virtual channels being busy. The transition rate out of state $\Re_{v_j}$ to $\Re_{(v+1)_j}$ is λ_j, while the rate out of $\Re_{v_j}$ to $\Re_{(v-1)_j}$ is $1/S_j$. The transition rate out of the last state, $\Re_{V_j}$, is reduced by λ_j to account for the arrival of messages while a channel is in this state. In the steady state, the model yields the following probabilities

$$Q_{v_j} = \begin{cases} 1, & v = 0 \\ Q_{v-1_j} \lambda_j S_j, & 0 < v < V \\ Q_{v-1_j} \dfrac{\lambda_j}{\frac{1}{S_j} - \lambda_j}, & v = V \end{cases}, \tag{4.34}$$

and

$$P_{v_j} = \begin{cases} 1/\sum_{i=0}^{V} Q_{i_j}, & v = 0 \\ P_{v-1_j}\lambda_j S_j, & 0 < v < V \\ P_{v-1_j}\dfrac{\lambda_j}{\frac{1}{S_j} - \lambda_j}, & v = V \end{cases} . \tag{4.35}$$

In virtual channel flow control, multiple virtual channels share the bandwidth of a physical channel in a time-multiplexed manner. After averaging over all the possible values of j, $1 \le j \le n$, the average degree of multiplexing of virtual channels, that takes place at a given physical channel, is given by [49]

$$\overline{V}_j = \frac{\sum_{v=1}^{V} v^2 P_{v_j}}{\sum_{v=1}^{V} v P_{v_j}}, \tag{4.36}$$

$$\overline{V} = \sum_{j=1}^{d_{\max}} \theta_j \overline{V}_j . \tag{4.37}$$

4.1.3 The hypercube case

When the network is a hypercube (k=2) some equations in the above model should change as follows. The network diameter, the number of nodes located i hops away from a given node, n_i, in the n-dimensional hypercube, and the probability that a node is i hops away from the given node, θ_i, are given respectively by [3]

$$d_{\max} = n , \tag{4.38}$$

$$n_i = \binom{n}{i}, \tag{4.39}$$

$$\theta_i = \frac{n_i}{N-1} = \frac{\binom{n}{i}}{2^n - 1}. \tag{4.40}$$

The average latency for the regular messages, $\overline{S}_r$, can be calculated as

$$\overline{S}_r = \sum_{i=1}^{n} \theta_i S_{r_i}, \tag{4.41}$$

where S_{r_i} is the latency for an i-hop regular message which can itself be computed as

$$S_{r_i} = M + i + \sum_{l=1}^{i} B_{r_{i,l}}. \tag{4.42}$$

$B_{r_{i,l}}$ is the mean blocking time seen by an i-hop regular message at its l-th hop channel, $1 \le l \le i$. Let $P_{i,j,l}$ denote the probability of blocking for an i-hop regular message when at l-th hop is j hops away from the hotspot node. Then, the probability $B_{r_{i,l}}$ is given by

$$B_{r_{i,l}} = \sum_{j=1}^{n} \frac{C_j}{nN} P_{i,j,l} w_j. \tag{4.43}$$

The number of channels located j hops away from the hotspot node in an n-dimensional hypercube is given by [1]

$$C_j = (n - j + 1)\binom{n}{j-1}. \tag{4.44}$$

The probability $P_{i,j,l}$ can be derived as follows. At the l^{th}-hop channel, an i-hop regular message can use $(i-l+1)(V-1)$, $1 \le i \le n$, $1 \le l \le i$, adaptive virtual channels at the remaining $(i-l+1)$ dimensions to be visited adaptively. It can also use one deterministic channel at the lowest dimension to be visited according to deterministic routing. A

message is blocked when all the possible virtual channels it can use are busy. So, the probability that blocking occurs can be written as

$$P_{i,j,l} = P_{a_j}^{i-l} P_{a\&d_j} . \tag{4.45}$$

Recalling that the virtual channel requirement for Duato's adaptive routing in hypercube is slightly different from that in the general k-ary n-cube, the probability that all adaptive virtual channels are busy, P_{a_j}, and the probability that all adaptive and also deterministic virtual channels are busy, $P_{a\&d_j}$, are given by [36]

$$P_{a_j} = P_{V_j} + \frac{P_{(V-1)_j}}{\binom{V}{V-1}}, \tag{4.46}$$

$$P_{a\&d_j} = P_{V_j} . \tag{4.47}$$

As with a regular message, a hotspot message is blocked at a given channel when all the adaptive virtual channels of the remaining dimensions to be visited, and also the deterministic virtual channels of the lowest dimension still to be visited, are busy. Therefore, the above calculation of the mean blocking time for regular messages can be easily adapted for hotspot messages. Consider a hotspot message that has to cross j channels to reach its destination, the hotspot node. When the message reaches the m-th hop channel, it is (j-m+1) hops away from the hotspot node. Therefore, the mean blocking time can be written as

$$B_{h_{m,j}} = P_{j,m,j-m+1} w_{j-m+1} . \tag{4.48}$$

The new expressions for the probability of blocking, $P_{j,m,j-m+1}$, and mean waiting time, w_{j-m+1}, can be obtained by simply substituting j by (j-m+1) in the above equations.

4.2 Model validation

The above model has been validated through a discrete-event simulator that performs a time-step simulation of the network operations at the flit level. In each simulation experiment, a total number of 100000 messages is delivered. Statistics gathering was inhibited for the first 10000 messages to avoid distortions due to the initial start up conditions. Extensive validation experiments have been performed for several combinations of network sizes, message lengths, and virtual channels, and the general conclusions have been found to be consistent across all the cases considered. However, for the sake of specific illustration we provide results for the following cases:

- Examined networks are 8-ary 2-cube, 8-ary 3-cube and 8-dimensional hypercube.
- Message length M=32 and 64 flits.
- Number of virtual channels V=3 and 5 for the 8-ary 2-cube and 8-ary 3-cube and V=2 and 4 for the hypercube.
- Fraction of hotspot traffic h = 0.07, 0.21, 0.35 and 0.49. The hotspot node is assumed to be the node (4,4) in 8-ary 2-cube, the node (4,4,4) in 8-ary 3-cube and the node with linear address N-1=255, in the 8-dimensional hypercube. Due to the symmetry of k-ary n-cube networks, any node can obviously be the hotspot node without any change in simulation results.

Figures 4.3-4.6 show the mean latency curves predicted by the model against those obtained through simulation experiments. In all the figures, the horizontal axis represents the traffic rate (λ_g) while the vertical axis shows the mean message latency in crossing from source to destination. The figures indicate that the analytical model predicts the mean message latency with a reasonable degree of accuracy when the network is in the steady

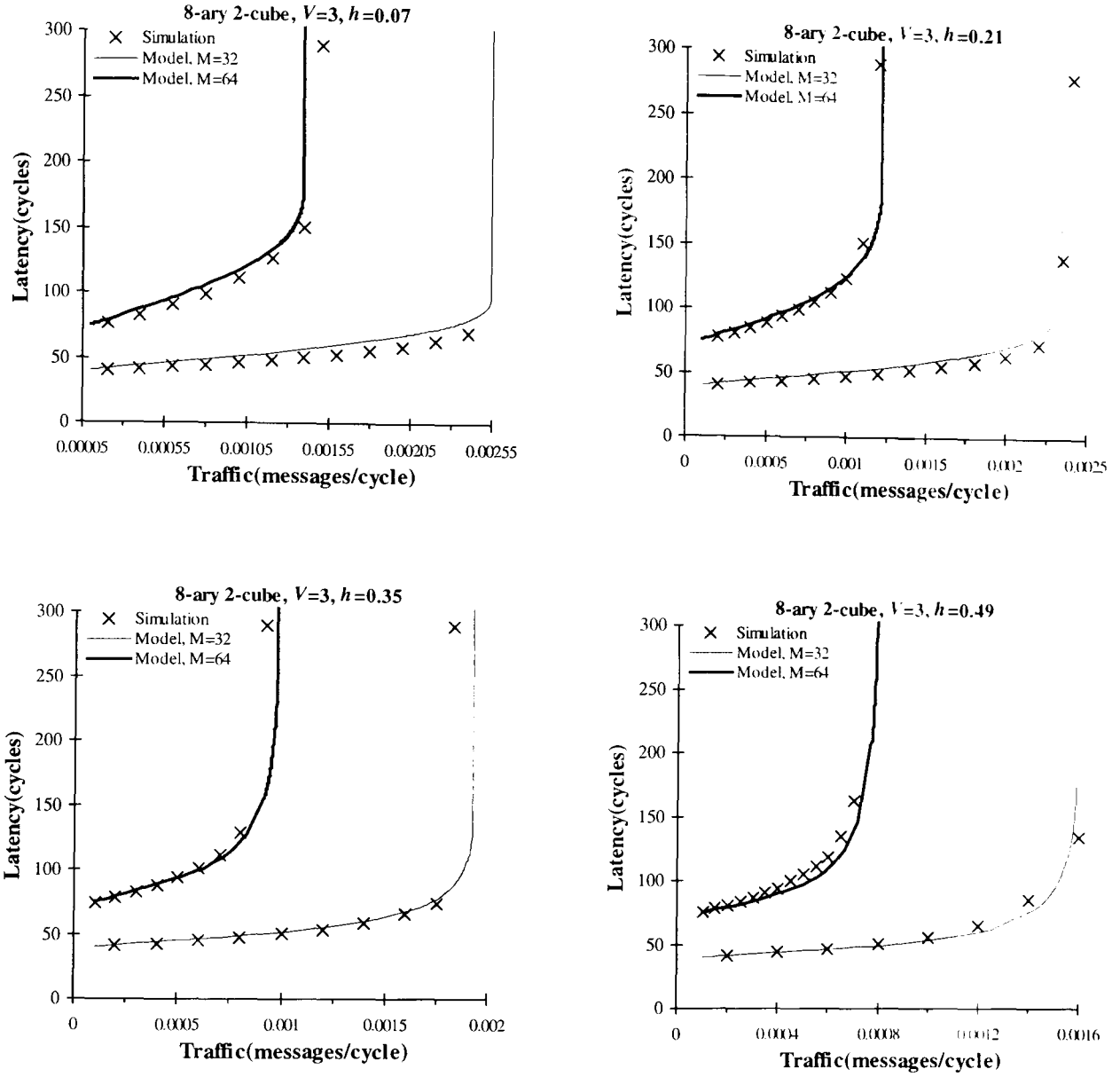
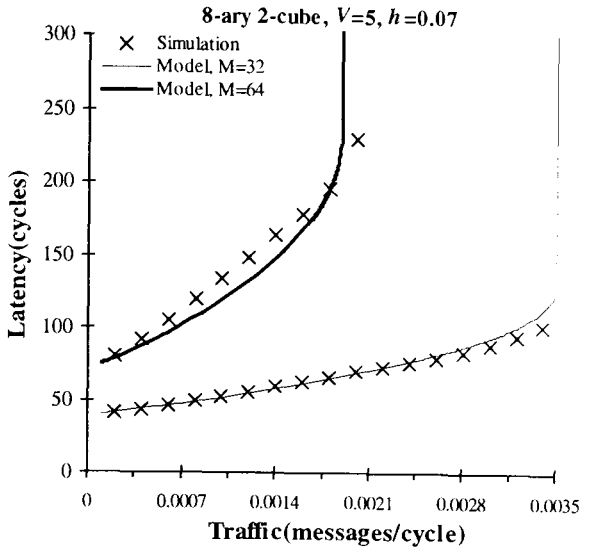
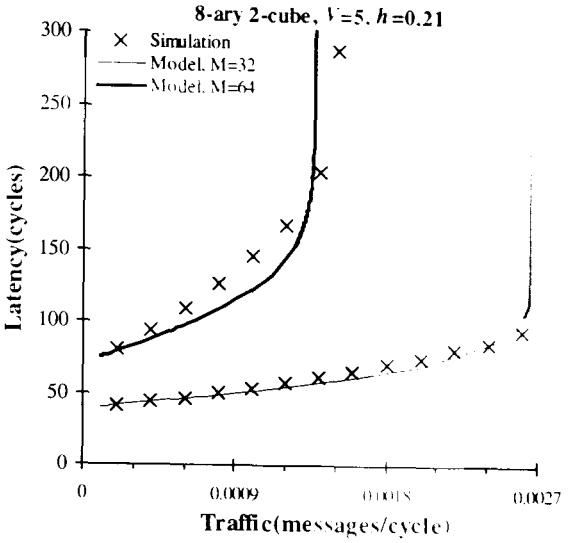
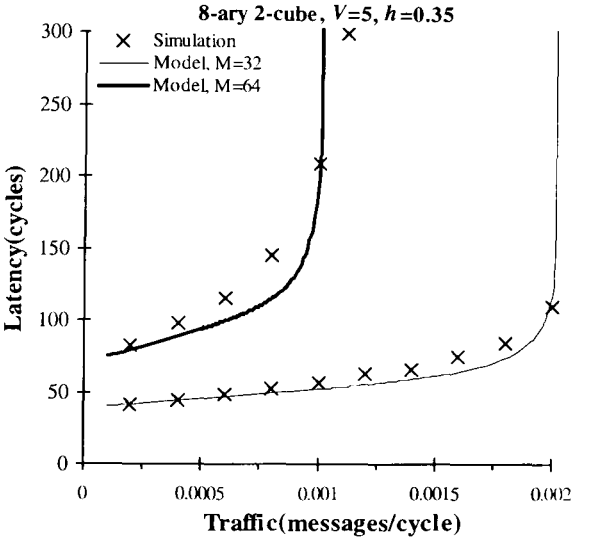


Figure 4.3: The average message latency predicted by the model against simulation results for an 8-ary 2-cube with message length M=32 and 64 flits, hotspot traffic portions h=0.07, 0.21, 0.35 and 0.49, and V=3 virtual channels per physical channel.

state region, that is when it has not reached saturation point. However, there are discrepancies in the results provided by the model and simulation when the network is under heavy traffic and approaches the saturation point. This is due to the approximations that have been made in the analysis to ease the model development. For instance, Equation







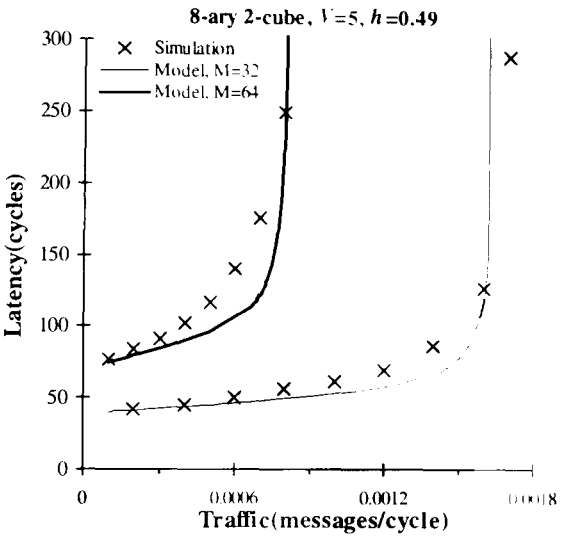


Figure 4.4: The average message latency predicted by the model against simulation results for an 8-ary 2-cube with message length M=32 and 64 flits, hotspot traffic portions h=0.07, 0.21, 0.35 and 0.49, and V=5 virtual channels per physical channel.

4.28 is a crude approximation for computing the variance of the service time received by a message at a given output channel, especially in heavy traffic regions, but it greatly simplifies the calculation of the mean message waiting time. The minimum service time, M, for a channel assumed in 4.28 is much less than the real minimum service time when

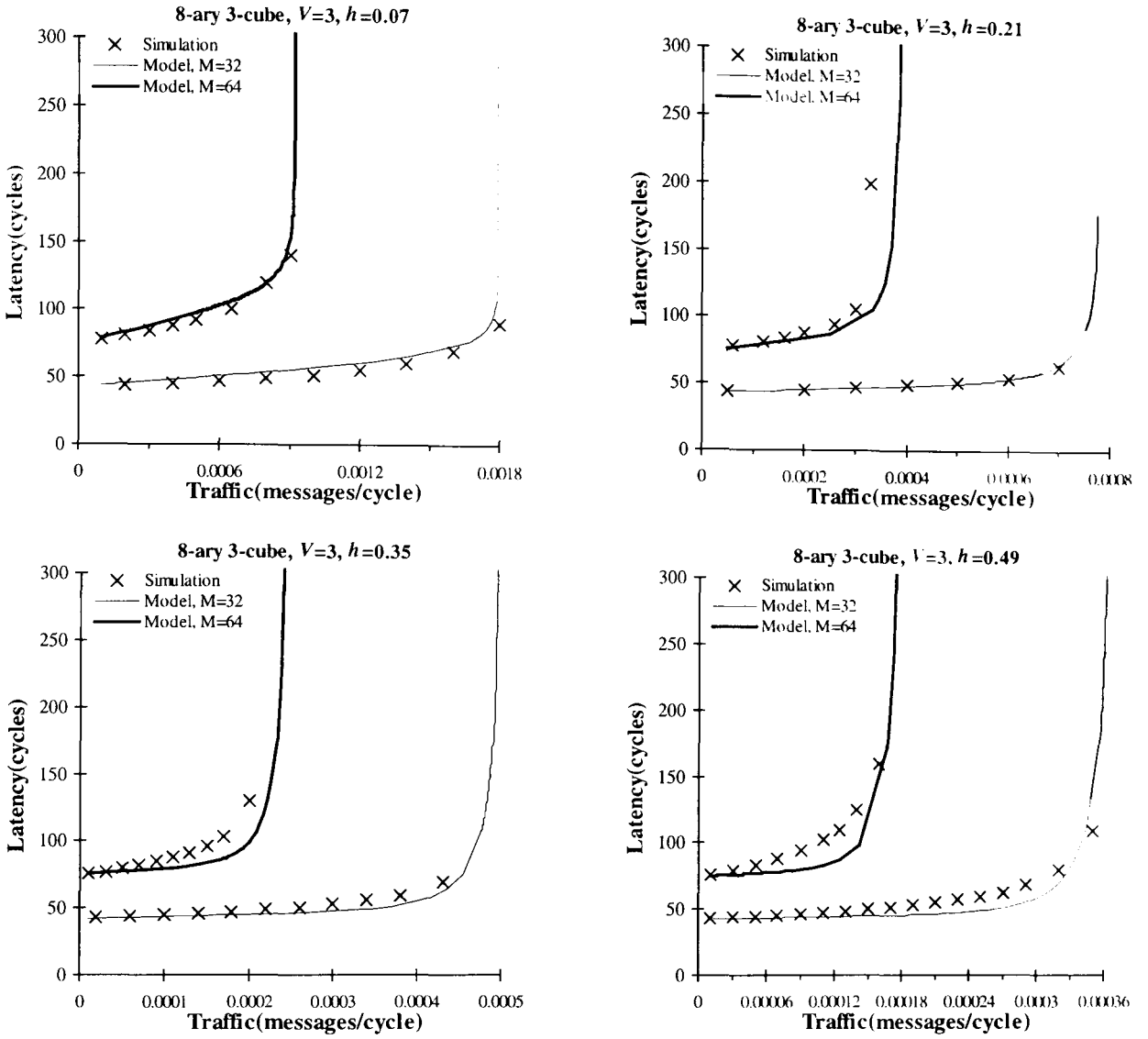


Figure 4.5: The average message latency predicted by the model against simulation results for an 8-ary 3-cube with message length M=32 and 64 flits, hotspot traffic portions h=0.07, 0.21, 0.35 and 0.49, and V=3 virtual channels per physical channel.

the traffic is dense, although it is an appropriate approximation for light traffic loads. Nevertheless, we can conclude that the model produces latency results with a good degree of accuracy in the steady state regions and its simplicity makes it a good practical evaluation tool that can be used to gain insight into the performance behavior of fully adaptive routing in the k-ary n-cube in the presence of hotspot traffic.

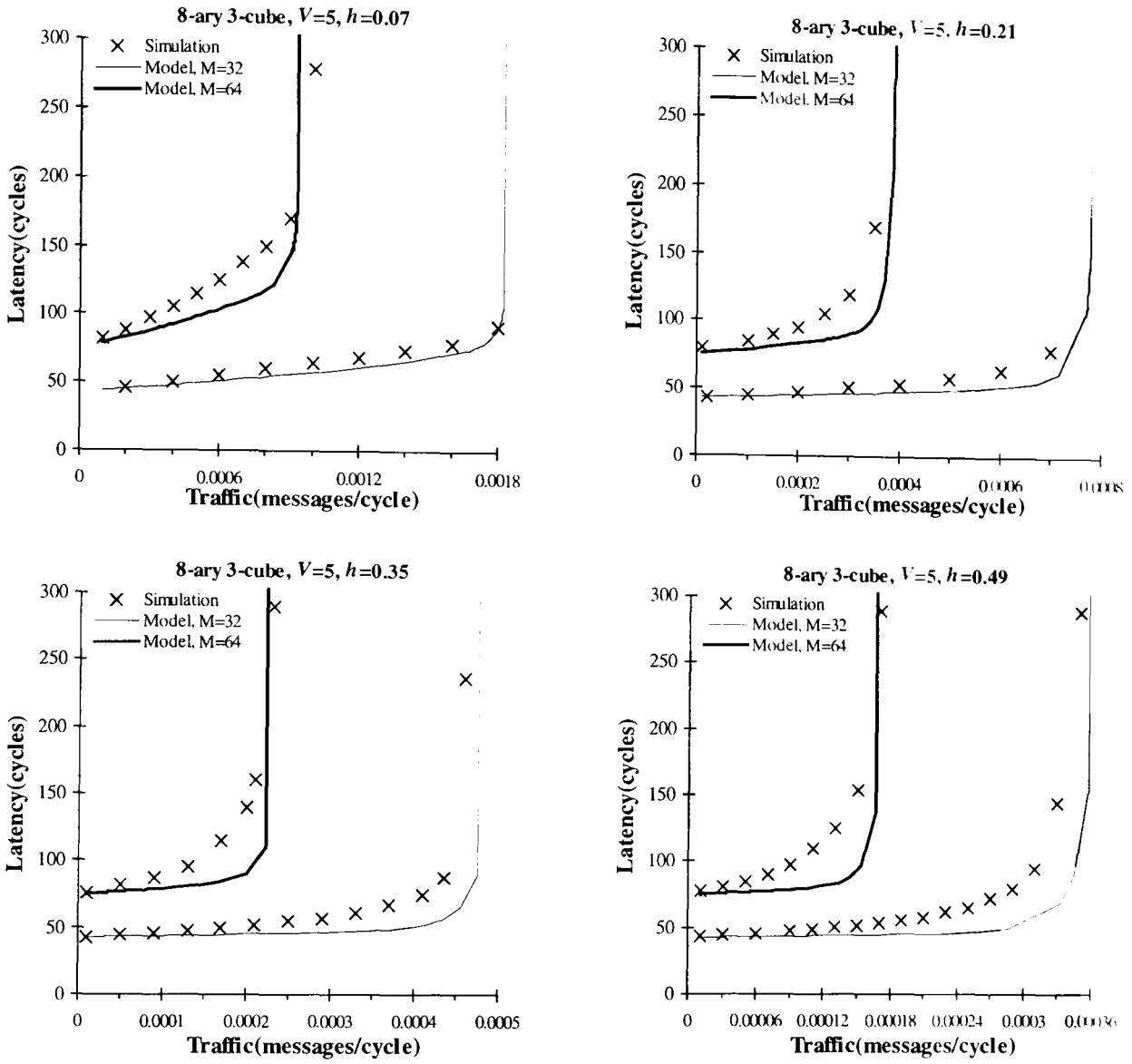


Figure 4.6: The average message latency predicted by the model against simulation results for an 8-ary 3-cube with message length M=32 and 64 flits, hotspot traffic portions h=0.07, 0.21, 0.35 and 0.49, and V=5 virtual channels per physical channel.

As can be seen in the figures, changing the number of virtual channels from V=3 in Figure 4.3 to V=5 in Figure 4.4 has improved the mean message latency. However, the effect of the number of virtual channels is more noticeable when the hotspot traffic portion is relatively small. Thus, for a small percentage of hotspot traffic, the saturation point for the 8-ary 2-cube for V=3 and 5 are quite different. However, when the hotspot traffic rate is

high, more virtual channels cannot effectively improve the latency because most of messages are targeted at the hotspot node and are waiting for many other messages which are also being sent to the same node. When the network is an 8-ary 3-cube, even a hotspot traffic factor as small as h=0.07 defines a sufficiently large hotspot traffic portion to place the network in a condition where adding two virtual channels cannot help to improve message latency (see Figures 4.5 and 4.6). In an 8-ary 3-cube network (512 nodes), this corresponds to a total rate of $511*0.07=35.77\lambda_g$ messages/cycle sent to the hotspot node. This is higher than the same parameter in a 8-ary 2-cube (with 64 nodes) network with a hotspot traffic factor h=0.49, corresponding to $63*0.49=30.87\lambda_g$ hotspot messages/cycle.

In order to assess the accuracy of the model in predicting the latency for the different types of messages in the network, Figures 4.7 and 4.8 provides detailed results for the overall message latency (regular and hotspot messages combined together), hotspot message latency (hotspot messages only), and regular message latency (uniform messages only) predicted by the analytical model, plotted against those obtained by the simulator when V=2. Figure 4.9 shows the results for the overall message latency only with V =4. (We have opted to show results for the overall message latency only in Figures 4.9 for brevity).

The accuracy of the model against the hotspot fraction is illustrated in Figure 4.10, where the overall message latency is plotted against the fraction of the hotspot traffic where V = 2 and $\lambda_g = 0.001$.

4.3 Modelling bidirectional *k*-ary *n*-cubes

In this section, the above model is modified for bidirectional k-ary n-cubes. To this end, the following changes should be made to the model's equations. The average number of hops that a regular messages makes at each dimension $\bar{k}$, and the network diameter $d_{\max}$, are given by [6]

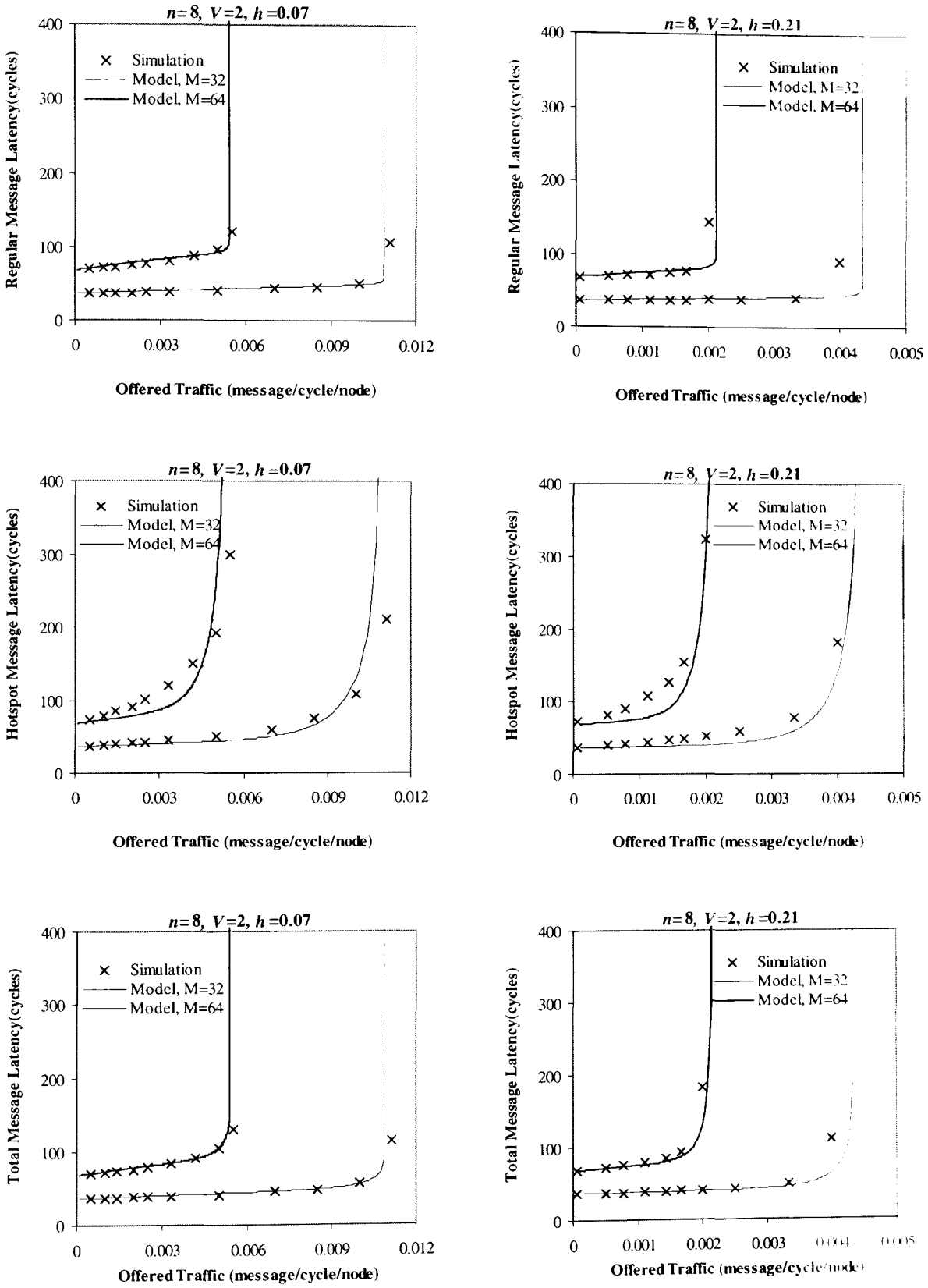


Figure 4.7: The average message latency predicted by the model against simulation results in an 8-dimensional hypercube with V=2 virtual channels per physical channel, M=32 and 64 flits, and hotspot traffic portions h= 0.07, 0.21. Bottom row shows the overall message latency while the top and middle rows show the latency of uniform and hotspot messages respectively.

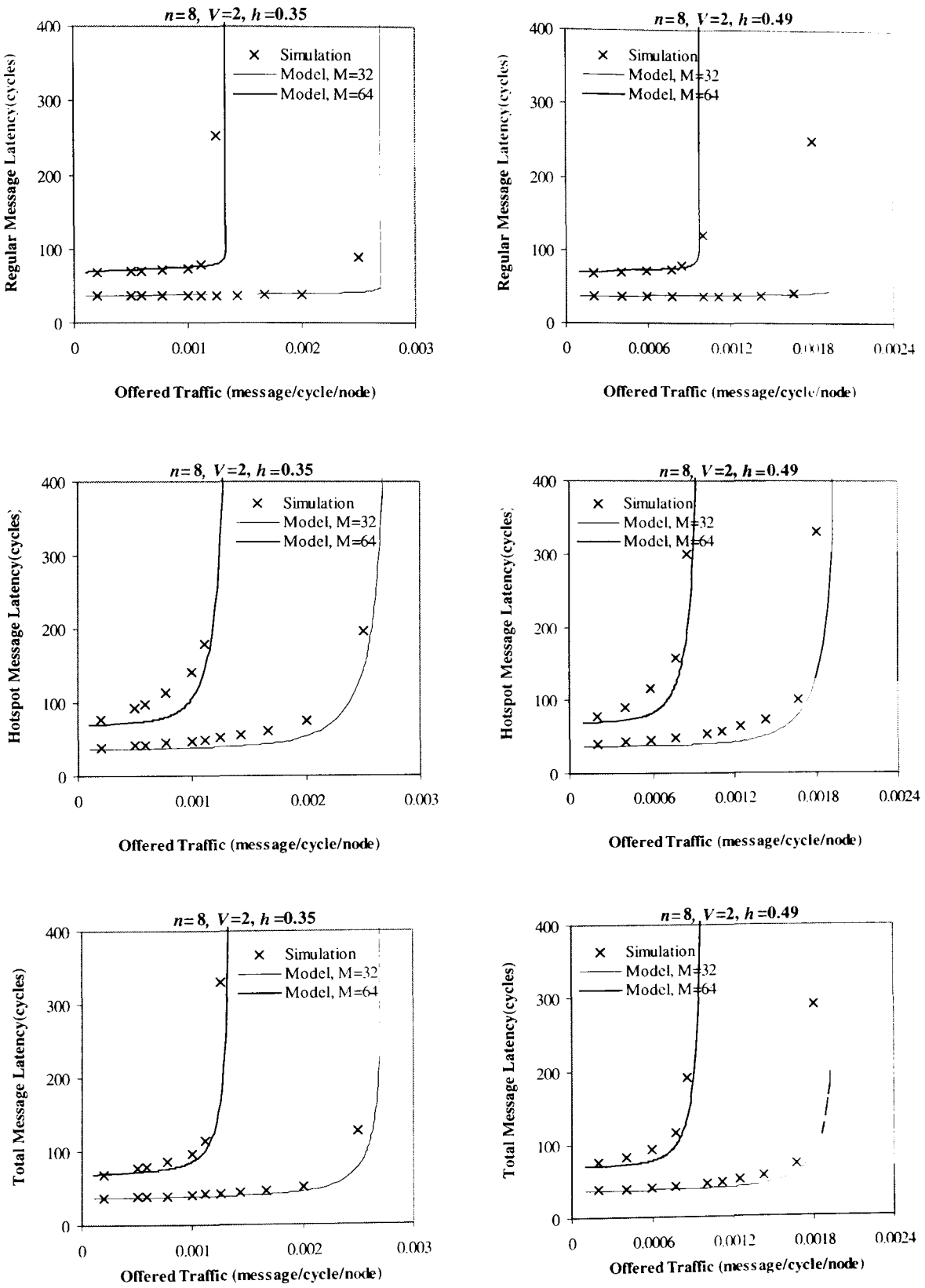


Figure 4.8: The average message latency predicted by the model against simulation results in an 8-dimensional hypercube with V=2 virtual channels per physical channel, M=32 and 64 flits, and hotspot traffic portions h= 0.35, 0.49. Bottom row shows the overall message latency while the top and middle rows show the latency of uniform and hotspot messages respectively.

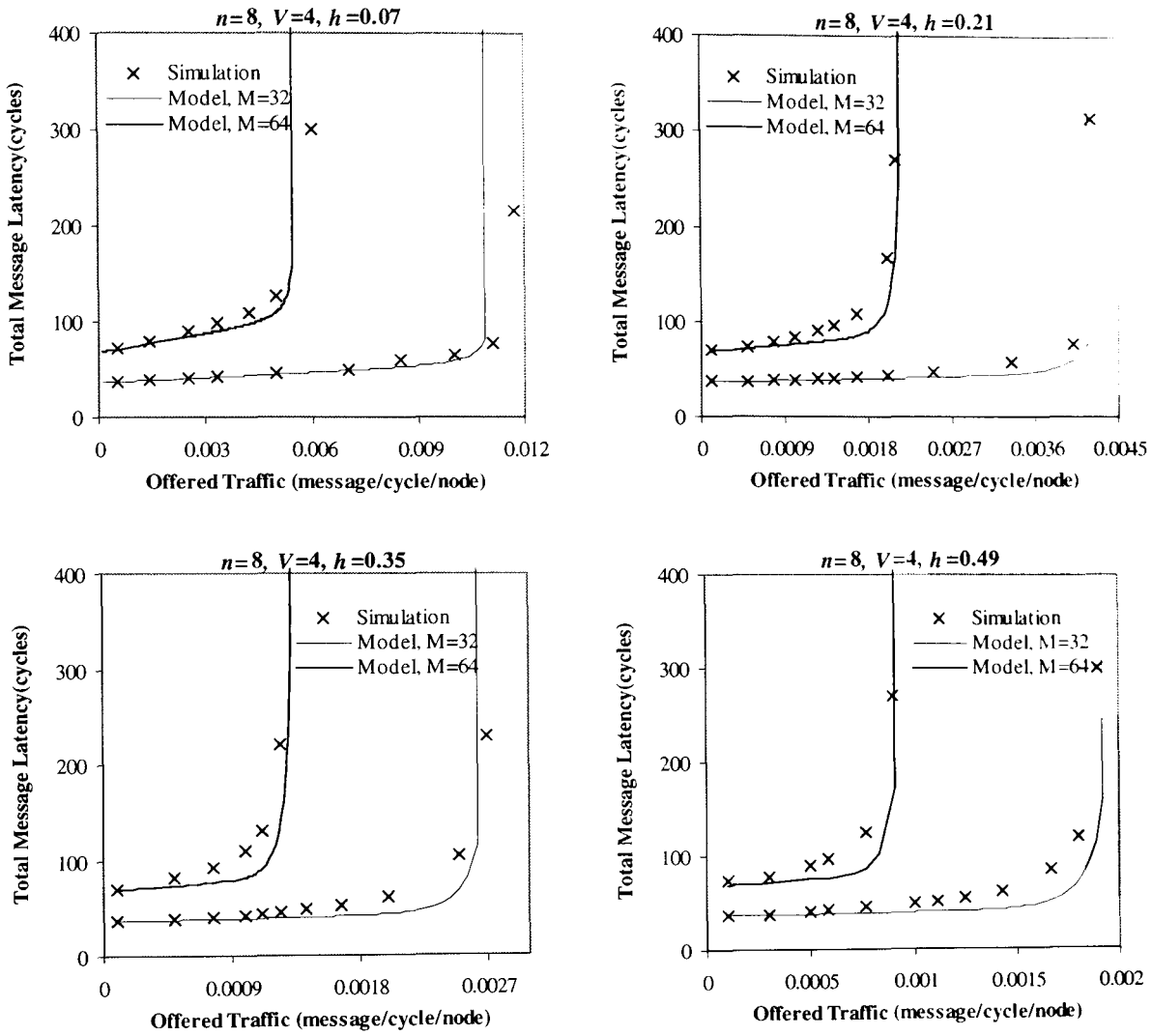


Figure 4.9: The overall message latency predicted by the model against simulation results in the 8-dimensional hypercube with V=4 virtual channels per physical channel, message length M=32 and 64 flits, and hotspot traffic portions h=0.07, 0.21, 0.35, 0.49.

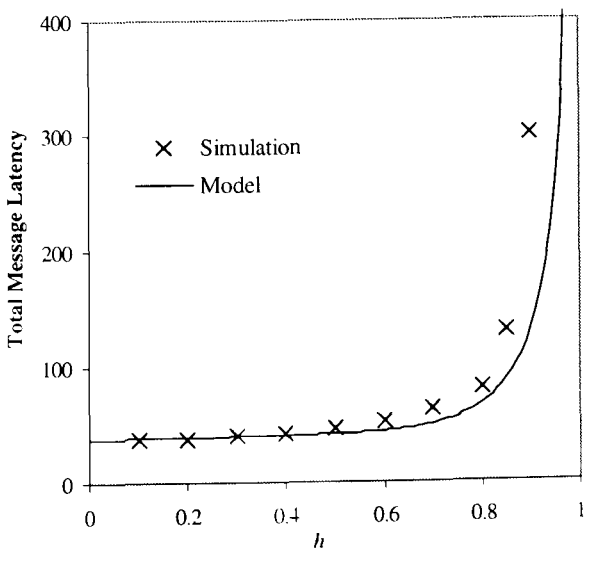


Figure 4.10: The overall message latency versus the portion of the hotspot traffic, h, in the 8-dimensional hypercube with V=2, M=32, and λ_g =0.001.

$$\bar{k} = \begin{cases} \dfrac{k}{4}, & \text{if } k \text{ is even} \\ \dfrac{1}{4}(k - \dfrac{1}{k}), & \text{otherwise} \end{cases}, \tag{4.49}$$

$$d_{\max} = n\left\lfloor \frac{k}{2} \right\rfloor. \tag{4.50}$$

The number of nodes which are i hops away from a given node, n_i, in the bidirectional k-ary n-cube is the surface area of radius i (given by Theorem 2.6), i.e.

$$n_i = \begin{cases} \displaystyle\sum_{m=0}^{n}\sum_{l=0}^{m}(-1)^l 2^m \binom{n}{m}\binom{m}{l}\binom{i-\frac{kl}{2}-2l-1}{m-1}, & k \text{ is odd} \\ \displaystyle\sum_{t=0}^{n}\sum_{m=0}^{n-t}\sum_{l=0}^{m}(-1)^l 2^m \binom{n}{t}\binom{n-t}{m}\binom{m}{l}\binom{i-\frac{k(l+t)}{2}-2l-1}{m-1}, & \text{otherwise} \end{cases}. \tag{4.51}$$

To calculate the number of channels located j hops away from a given node, in the bidirectional k-ary n-cube, we need to know the number of nodes located j hops away from a given node with at least one hop distance from the given node in each dimension.

THEOREM 4.2. The number of nodes located i hops away from a given node in the bidirectional k-ary n-cube with at least one hop distance from the given node in each dimension, is given by

$$\Phi_{\geq 1}^{k,n}(i) = \begin{cases} \displaystyle\sum_{m=0}^{n}\sum_{l=0}^{m}(-1)^l 2^n \binom{n}{m}\binom{m}{l}\binom{i-n-\frac{l(k-1)}{2}+l-1}{m-1}, & k \text{ is odd} \\ \displaystyle\sum_{t=0}^{n}\sum_{m=0}^{n-t}\sum_{l=0}^{m}(-1)^l 2^{n-t} \binom{n}{t}\binom{n-t}{m}\binom{m}{l}\binom{i-\frac{tk}{2}+t-n-\frac{lk}{2}+2l-1}{m-1}, & \text{otherwise} \end{cases}. \tag{4.52}$$

PROOF. We should count the number of ways that i like objects can be distributed over two groups, each of n cells, say $G = \{C_1, C_2, \cdots, C_n\}$ and $G' = \{C'_1, C'_2, \cdots, C'_n\}$, such

that each cell contains not less than one and not more than (k-1)/2 cells and no two corresponding cells, C_i and C'_i for $i=1,2...,n$, can be assigned objects at the same time. The problem can be thought of as that of finding the number of different destinations that an i-hop message can chose from a given source node in $\vec{Q}_n^k$, using a minimal routing algorithm. We should consider two cases: odd k, and even k.

When k is odd, it is apparent that a message can take at most (k-1)/2 hops in each dimension since the network is bidirectional. Let us consider the case that an i-hop message makes some hops over m, ($m=0,1,2,...,n$), fixed dimensions (each in one direction) so that the message has made at least two hops in each dimension. The message makes one hop at any other n-m dimensions. This can be realized in $\Gamma_2^{\frac{k-1}{2}}(i-n+m,m)$ ways. Each of n dimensions could be one of these m dimensions resulting in $\binom{n}{m}\Gamma_2^{\frac{k-1}{2}}(i-n+m,m)$ possible combinations that m dimensions are passed (each in one direction). Recalling that each of two directions in one dimension can be chosen yields the total number of ways to pass m dimension with at least two hop in any of m dimensions as $2^m\binom{n}{m}\Gamma_2^{\frac{k-1}{2}}(i-n+m,m)$. The hops made at each of the other n-m dimensions (one per dimension) can be made in positive or negative directions, with 2^{n-m} combinations, making a total number of $2^{n-m}2^m\binom{n}{m}\Gamma_2^{\frac{k-1}{2}}(i-n+m,m)$ ways to pass m dimensions with at least two hops at each dimension and one hop per each n-m remaining dimensions. Summing up all the combinations for $m=0,1,...,n$ gives the total number of nodes at distance i from a given node in $\vec{Q}_n^k$ (with odd k), such that the distance between these nodes from the given node at each dimension be at least one, to be

$$\begin{aligned}\Phi_{\geq 1}^{k_{odd},n}(i) &= \sum_{m=0}^{n} 2^{n-m}2^m\binom{n}{m}\Gamma_2^{\frac{k-1}{2}}(i-n+m,m) \\ &= \sum_{m=0}^{n}\sum_{l=0}^{m}(-1)^l 2^n\binom{n}{m}\binom{m}{l}\binom{i-n-\frac{l(k-1)}{2}+l-1}{m-1}\end{aligned} \tag{4.53}$$

When the radix k is even, we should count the number of ways that i like objects can be distributed over two groups, each of n cells, say $G=\{C_1,C_2,\cdots,C_n\}$ and $G'=\{C'_1,C'_2,\cdots,C'_n\}$, such that each cell in G contains not less than one and not more than $k/2$-1 objects and each cell in G' contains not less than one and not more than $k/2$ objects and no two correspondent cells, C_i and C'_i, for all $i=0,1,\ldots,n$, can be assigned objects at the same time. Suppose that t, $t=0,1,\ldots,n$, cells in G' have received $k/2$ objects, in $\binom{n}{t}$ ways. The remaining objects may be distributed over the remaining n-t dimensions using the equation given above for odd k, since each dimension in G and G' now receives at most $k/2$-1 and at least one objects. Therefore, we can write $\Phi_{\geq 1}^{k_{even},n}(i)=\sum_{t=0}^{n}\binom{n}{t}\Phi_{\geq 1}^{k-1_{odd},n-t}(i-tk/2)$. Substituting $\Phi_{\geq 1}^{k-1_{odd},n-t}(i-tk/2)$ using the above equation (derived for odd k) results in

$$\Phi_{\geq 1}^{k_{even},n}(i)=\sum_{t=0}^{n}\sum_{m=0}^{n-t}\sum_{l=0}^{m}(-1)^l 2^{n-t}\binom{n}{t}\binom{n-t}{m}\binom{m}{l}\binom{i-\frac{tk}{2}+t-n-\frac{lk}{2}+2l-1}{m-1}. \qquad (4.54)$$

Hence, from Equation 4.53 and Equation 4.54, we have

$$\Phi_{\geq 1}^{k,n}(i)=\begin{cases}\sum_{m=0}^{n}\sum_{l=0}^{m}(-1)^l 2^n\binom{n}{m}\binom{m}{l}\binom{i-n-\frac{l(k-1)}{2}+l-1}{m-1}, & k \text{ is odd}\\ \sum_{t=0}^{n}\sum_{m=0}^{n-t}\sum_{l=0}^{m}(-1)^l 2^{n-t}\binom{n}{t}\binom{n-t}{m}\binom{m}{l}\binom{i-\frac{tk}{2}+t-n-\frac{lk}{2}+2l-1}{m-1}, & \text{otherwise}\end{cases}.$$

Let us now calculate the number of channels located j hops away from a given node, in the bidirectional k-ary n-cube.

THEOREM 4.3. The number of channels that are j hops away from a given node in a bidirectional k-ary n-cube is given by

$$\overleftrightarrow{C}_j =$$

$$\begin{cases} \sum_{l=0}^{n-1}\sum_{m=0}^{n-l}\sum_{z=0}^{m}(-1)^z 2^{n-l}(n-l)\binom{n}{l}\binom{n-l}{m}\binom{m}{z}\binom{i-n-l-\frac{z(k-1)}{2}+z-1}{m-1}, & k \text{ is odd} \\ \sum_{l=0}^{n-1}\sum_{t=0}^{n-l}\sum_{m=0}^{n-l-t}\sum_{z=0}^{m}(-1)^z 2^{n-l-t}(n-l)\binom{n}{l}\binom{n-l}{t}\binom{n-l-t}{m}\binom{m}{z}\binom{i-\frac{tk}{2}+t-n-l-\frac{zk}{2}+2z-1}{m-1}, & \text{otherwise} \end{cases}$$

(4.55)

PROOF. A similar scheme used to prove Theorem 4.1 can be applied here. A channel located j hops away from a given node, say A, is an output channel of a node also j hops away from A. However, some of these channels might be ignored when counting. The number of nodes having no such a channel is simply $\Phi_{\geq 1}^{k,n}(j)$ (given by Equation 4.52), since $\Phi_{\geq 1}^{k,n}(j)$ gives the number of nodes located j hops away from A, each at least at distance one from A in each dimension. The number of nodes having one channel to be excluded for counting is $\binom{n}{1}\Phi_{\geq 1}^{k,n-1}(j)$, which is the number of nodes located j hops away from A whose distance from A in exactly one dimension is zero. Generally, the number of nodes which have l channels to be omitted when counting the number of channels located j hops away from A is $\binom{n}{l}\Phi_{\geq 1}^{k,n-l}(j)$. Summing up all these cases results in the total number of channels which are j hops away from A, $\overleftrightarrow{C}_j$, as

$$\overleftrightarrow{C}_j = \sum_{l=0}^{n-1}(n-l)\binom{n}{l}\Phi_{\geq 1}^{k,n-l}(j) =$$

$$\begin{cases} \sum_{l=0}^{n-1}\sum_{m=0}^{n-l}\sum_{z=0}^{m}(-1)^z 2^{n-l}(n-l)\binom{n}{l}\binom{n-l}{m}\binom{m}{z}\binom{i-n-l-\frac{z(k-1)}{2}+z-1}{m-1}, & k \text{ is odd} \\ \sum_{l=0}^{n-1}\sum_{t=0}^{n-l}\sum_{m=0}^{n-l-t}\sum_{z=0}^{m}(-1)^z 2^{n-l-t}(n-l)\binom{n}{l}\binom{n-l}{t}\binom{n-l-t}{m}\binom{m}{z}\binom{i-\frac{tk}{2}+t-n-l-\frac{zk}{2}+2z-1}{m-1}, & \text{otherwise} \end{cases}$$

.□

The mean blocking time, B_r, for a regular message can now be written as

$$B_r = \sum_{j=1}^{d_{\max}} \frac{\vec{C}_j}{2nN} \varphi_{j,\bar{k}} w_j \,. \tag{4.56}$$

4.4 Performance analysis

In this section we use the model proposed above to conduct a performance analysis of k-ary n-cubes under traffic workloads containing a hotspot. For the sake of an example, the 10-ary 3-cube is used, but the conclusions reached are found to be similar when other network configurations are considered.

Figure 4.11 reveals the effect of the number of virtual channels on the performance by plotting the offered traffic when the network is saturated (saturation traffic) versus the number of virtual channels in the unidirectional 10-ary 3-cube with message length M=50 flits and number of virtual channels V=4 for hotspot fractions h=0.01 and 0.2. It is assumed that the network enters the saturation region when $\rho_j \geq 1$ (given by Equation 4.22); the corresponding λ_g for which the condition $\rho_j \geq 1$ is satisfied is the saturation traffic. As can be seen in the figure, adding virtual channels when the hotspot fraction is low (h=0.01) increases the performance noticeably. However, a performance improvement is not noticeable at all when the hotspot fraction is relatively high (h=0.2). In this situation, more virtual channels cannot effectively improve the latency because messages sent to the hotspot node are forced to wait for other messages trying to reach the same destination.

Figure 4.12 shows the average network latency, $\overline{S}$, as a function of offered traffic by each node for a unidirectional 10-ary 3-cube with message length M=50 flits and number of virtual channels V=4 for hotspot fractions h=0.01 and 0.2. The contributions of hotspot and regular traffic to total mean message latency, $\overline{S}$, are shown separately in dark and light grey, respectively. The figure reveals that when the hotspot fraction is low, the component causing network saturation is the part contributed by regular messages.

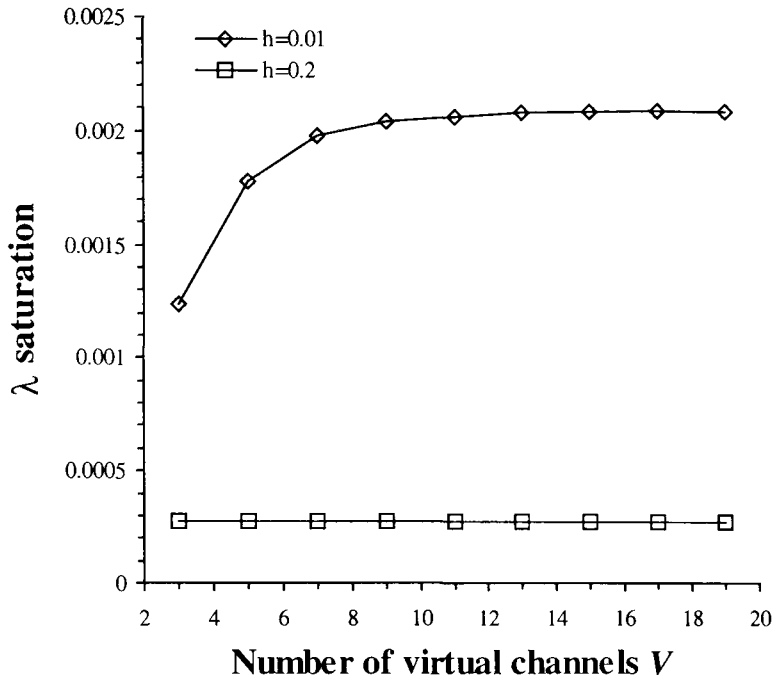


Figure 4.11: The saturate traffic rate versus number of virtual channels per physical channel in a unidirectional 10-ary 3-cube with message length M=50 flits and hotspot traffic portions h=0.01 and 0.2.

However, when hotspot fraction is high, the part due to hotspot messages dominates the other part and causes saturation.

Figure 4.13 shows mean message latency curves for message length M = 50 flits in unidirectional and bidirectional 10-ary 3-cubes with V = 4 virtual channels per physical channel and hotspot fractions h=0.01 and 0.2. The bidirectional k-ary n-cube has double the bisection width and node pin-out of its unidirectional equivalent. In order to have a fair and realistic comparison, the bisection width or pin-out constraint was held constant in the two networks. So, if we use the unidirectional network as a basis for the comparison (with a channel width of the flit size), the channel width in the bidirectional network will be half of the flit size, i.e. for each flit communication over a channel in the bidirectional network, two channel cycles are required. The figure reveals that the bidirectional k-ary n-cube

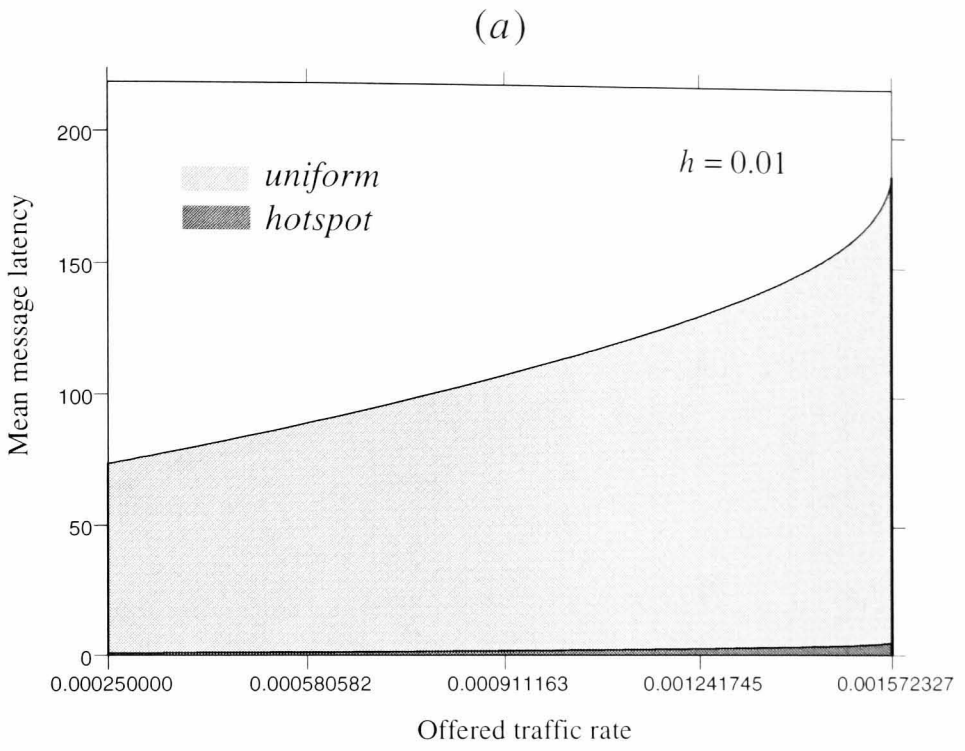


Figure 4.12: The mean message latency composed of two parts (hotspot and uniform messages) versus traffic generation rate in a unidirectional 10-ary 3-cube with message length M=50 flits V=4 virtual channels per physical channel for (a) hotspot traffic portion h=0.01, and (b) hotspot traffic portion h=0.2.

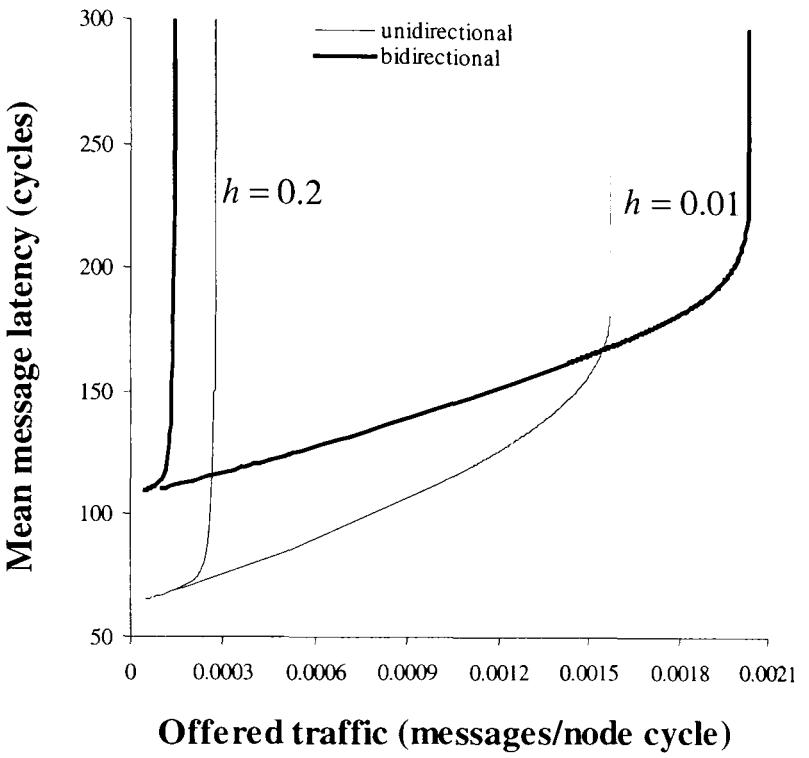


Figure 4.13: The mean message latency versus traffic generation rate in the unidirectional and bidirectional 10-ary 3-cube with message length M=50 flits and V=4 virtual channels per physical channel for hotspot traffic portions h=0.01 and 0.2, under pin-out and bisection bandwidth constraints.

outperforms its unidirectional counterpart under constant bisection width and pin-out constraints when the hotspot fraction is low. This is because the former network has a lower message distance and lower message traffic rate on its channels, compensating for its lower channel bandwidth. Note that when the hotspot fraction is low, the traffic is almost equally distributed over network channels. However, when the hotspot fraction is relatively high, the unidirectional network outperforms the bidirectional counterpart. This is because the large fraction of hotspot traffic causes a large portion of traffic to be placed on the channels around the hotspot. Even the lower diameter of a bidirectional network

cannot compensate for the long service time of the channels around a hotspot whose bandwidths are half of those in an equivalent unidirectional network.

4.5 Conclusions

Several analytical models of fully adaptive routing have recently been proposed for wormhole-routed k-ary n-cubes under the uniform traffic pattern. However, the "uniform traffic" assumption is not always justifiable in practice as there are many parallel applications that exhibit non-uniform traffic flows, which can produce, for example, hotspots in the network [88]. This chapter presented the first analytical model to compute the mean message latency in the presence of hotspot traffic in wormhole-routed k-ary n-cubes with fully adaptive routing. The model is based on assumptions widely used in similar studies. Simulation experiments have revealed that the proposed model produces latency results that are in a good agreement with those produced through simulation experiments.

A preliminary performance analysis has revealed that increasing the number of virtual channels can improve network performance when the hotspot traffic portion, h, is low. However, when h is relatively large (defining a high percentage of hotspot traffic), adding virtual channels cannot improve performance noticeably. It was also shown that when h is small, the dominating factor causing network saturation is the uniform message component while for large h, the dominating factor is the average latency for hotspot messages. Comparing unidirectional k-ary n-cubes against their bidirectional counterparts, under both constant bisection bandwidth and pin-out constraints, shows that bidirectional k-ary n-cubes provide better performance when the hotspot traffic rate is low. However, the opposite is true when hotspot traffic is relatively high.

In the next chapter, we will develop models to deal with two important non-uniform traffic

patterns, namely matrix-transpose and digit-reversal. These patterns are exhibited in many applications including signal processing and matrix-computation.

Chapter 5

Modelling of k-Ary n-Cubes for Other Important Non-Uniform Traffic Patterns

Many real-world parallel applications in science and engineering exhibit non-uniformity in the traffic patterns [41, 57, 90, 168] they create. For instance, the computation of multi-dimensional FFTs, finite element algorithms, matrix problems, and divide and conquer strategies all generate regular communication patterns [79], which may be non-uniform and put uneven bandwidth requirement on network channels. Permutations such as matrix-transpose, digit-reversal, shuffle, exchange, butterfly and vector-reversal are examples of regular communication patterns that generate typical non-uniform traffic in the network (see [59, 79] for more details on these permutations).

In this chapter, analytical models of fully adaptive routing in k-ary n-cubes in the presence of non-uniform traffic generated by two important permutation patterns, namely matrix-transpose and digit-reversal, are proposed. Results obtained through simulation experiments confirm that the proposed models predict message latency with an acceptable degree of accuracy under different working conditions.

The rest of the chapter is organised as follows. Section 5.1 describes the analytical model

for matrix-transpose permutations. In Section 5.2 the model is described for digit-reversal traffic. Both models are validated in Section 5.3 through simulation experiments, while Section 5.4 discusses the changes required for bidirectional k-ary n-cubes. Section 5.5 uses the proposed models for performance analysis of k-ary n-cubes in the presence of permutation traffic patterns. Section 5.6 concludes the chapter.

5.1 The analytical model for matrix-transpose traffic

The notation used in the model is shown in Table 5.1. Moreover, the model uses assumptions which are commonly employed in the literature [3-6, 12, 32-34, 42, 43, 48, 49, 58, 81, 84-86, 99, 120, 142, 143] as follows.

a) To cover a wider traffic range and to allow for a more generally applicable model we assume that there are two types of traffic in the network: "matrix-transpose" and "uniform". In the traffic pattern generated according to the matrix-transpose permutation [79], a message originating from source node $X = x_1 x_2 \cdots x_n$ is sent to the node

$$\mathbb{M}(X) = \begin{cases} x_{l+1}x_{l+2}\cdots x_{2l}x_1x_2\cdots x_l & \text{if } n = 2l \\ x_{l+1}x_{l+2}\cdots x_{2l+1}x_1x_2\cdots x_l & \text{if } n = 2l+1 \end{cases}. \tag{5.1}$$

In the uniform traffic pattern, a message is destined to any other nodes in the network with equal probability. Let us refer to these two types of messages as matrix-transpose and uniform messages, respectively. When a message is generated it has a probability m of being a matrix-transpose message and probability $1-m$ of being uniform. When m=0 the traffic pattern is purely uniform and when m=1 only matrix-transpose traffic is defined. A similar traffic model has already been used by the authors in [149] to generate other non-uniform traffic

Table 5.1: Notation used in the model for matrix-transpose permutation traffic

Notation	Description
d_m	average number of hops made by a matrix-transpose message across the network
d_u	average number of hops made by a uniform message across the network
$\overline{d}$	average number of hops made by a message across the network
k	network radix
k_u	average number of hops made by a uniform message in each dimension
k_m	average number of hops made by a matrix-transpose message in each dimension
L	message length
m	probability that a source generates a matrix-transpose message
$\mathcal{M}(X)$	matrix-transpose permutation function
n	network dimension
$n_{m_i,even}$	number of different source and destination pairs whose addresses differ in i digits and the destination address is the matrix-transpose permutation of the source address when n is even
$n_{m_i,odd}$	number of different source and destination pairs whose addresses differ in i digits and the destination address is the matrix-transpose permutation of the source address when n is odd
n_{m_i}	number of different source and destination pairs whose addresses differ in i digits and the destination address is the matrix-transpose permutation of the source address
N	network size ($N=k^n$)
P_a	probability that all adaptive virtual channels at a physical channel are busy
$P_{a\&d}$	probability that all adaptive and deterministic virtual channels at a physical channel are busy
$P_{block_{m_i,j}}$	probability of blocking when a matrix-transpose message, whose source and destination addresses differ in i digits, is at its j-th hop channel
$P_{block_{m_l,i,j}}$	probability of blocking when a matrix-transpose message, whose source and destination addresses differ in i digits, has already passed l dimensions and is at its j-th hop channel
$P_{block_{u_j}}$	probability of blocking when a d_u-hop uniform message is at its j-th hop channel
$P_{block_{u_l,j}}$	probability of blocking when a d_u-hop uniform message has already passed l dimensions at its j-th hop channel
P_{m_i}	probability that the source and destination addresses for a matrix-transpose message differ in exactly i digits
$P_{pass_{m_l,i,j}}$	probability that l dimensions are passed by a matrix-transpose message, whose source and destination addresses differ in i digits, when it is at its j-th hop channel
$P_{pass_{u_l,j}}$	probability that l dimensions are passed when a d_u-hop uniform message is at its j-th hop channel
P_{u_i}	probability that a uniform message makes i hops to reach its destination
P_v	probability that v virtual channels at a physical channel are busy
Q_v	an intermediate variable for calculating P_v
$\overline{S}$	mean network latency for a message
S_m	mean network latency for a matrix-transpose message
S_{m_i}	network latency for a matrix-transpose message whose source and destination addresses differ in i digits
S_u	mean network latency for a uniform message

Table 5.1: continued

Notation	Description
V	number of virtual channels used per physical channel
$\overline{V}$	average degree of multiplexing of virtual channels at a physical channel
w_c	mean waiting time to acquire a virtual channel when a message is blocked
$\overline{W}_s$	mean waiting time at a source node
x	address of a source node $x = x_1 x_2 \cdots x_n$
ζ_m	fraction of the matrix-transpose traffic in the network
ζ_u	fraction of the uniform traffic in the network
λ_m	generation rate of matrix-transpose messages at a source node
λ_c	message rate on a channel
λ_g	generation rate of messages at a source node
λ_u	generation rate of uniform messages at a source node
σ_S^2	variance of the service time distribution at a channel

patterns containing hotspots.

b) Nodes generate traffic independently of each other which follows a Poisson process with a mean rate of λ_g messages/cycle. Hence, the mean generation rate of the uniform traffic is $(1-m)\lambda_g$ and that of the matrix-transpose traffic is $m\lambda_g$.

c) Message length is fixed and equal to M flits, each of which is transmitted in one cycle between two adjacent nodes.

d) The local queue in the source node has infinite capacity. Moreover, messages are transferred to the local PE through the ejection channel as soon as they arrive at their destinations.

e) V virtual channels are used per physical channel, divided in two groups VC_1 and VC_2 as discussed in Chapter 2. Group VC_1 contains 2 virtual channels, crossed deterministically (e.g. in an increasing order of dimensions) and group VC_2 contains $(V-2)$ virtual channels crossed adaptively. When there is more than one

adaptive virtual channel available a message chooses one at random. To simplify the model derivation, no distinction is made between the deterministic and adaptive virtual channels when computing the different virtual channel occupancy probabilities [142].

5.1.1 The outline of the model

The mean message latency is composed of the mean network latency, $\overline{S}$, that is the time to cross the network, and the mean waiting time seen by a message in the source node, $\overline{W_s}$. However, to capture the effects of virtual channel multiplexing, the mean message latency has to be scaled by a factor, $\overline{V}$, representing the average degree of virtual channel multiplexing, that takes place at a physical channel. Therefore, the mean message latency can be written as

$$Latency = (\overline{S} + \overline{W_s})\overline{V} . \tag{5.2}$$

Given that a uniform message can make between 1 and $d_{max}=n(k-1)$ hops (i.e., the network diameter), the average number of hops that a uniform message makes across the network, d_u, is given by

$$d_u = \sum_{i=1}^{d_{\max}} i P_{u_i} , \tag{5.3}$$

where P_{u_i} is the probability that a uniform message makes i hops to reach its destination. The average number of hops that a uniform message makes in each dimension can therefore be expressed as

$$k_u = \frac{d_u}{n} . \tag{5.4}$$

The probability P_{u_i} can be calculated using Theorem 2.4 as

$$P_{u_i} = \frac{\vec{A}_n^k(i)}{N-1} = \frac{1}{N-1}\sum_{l=0}^{n}(-1)^l \binom{n}{l}\binom{i-lk+n-1}{n-1}. \qquad (5.5)$$

Before computing the average number of hops that a matrix-transpose message makes across the network, let us first calculate the probability, P_{m_i}, that the source and destination addresses for a newly-generated matrix-transpose message differ in exactly i digits. Examining the address patterns generated by matrix-transpose permutations reveals that this probability has to be calculated in different ways for odd and even values of n. Let $X = x_1 x_2 \cdots x_n$ and $\mathbb{M}(X) = x'_1 x'_2 \cdots x'_n$ be, respectively, the source and destination addresses for a matrix-transpose message. When n is even, every digit difference between the first $n/2$ digits of the source and destination addresses, $x_1 x_2 \cdots x_{n/2}$ and $x'_1 x'_2 \cdots x'_{n/2}$, results in a same digit difference in the remaining $n/2$ address digits, $x_{n/2} x_{n/2+1} \cdots x_n$ and $x'_{n/2} x'_{n/2+1} \cdots x'_n$. Therefore, P_{m_i} is zero when i is odd. Let us determine the number of possible cases where the source and destination address patterns of a matrix-transpose message differ in i (i= 0, 2, 4, ..., n) digits. This can be done by simply considering only the first $n/2$ digits in the source and destination addresses, and thus enumerating the number of combinations where $x_1 x_2 \cdots x_{n/2}$ and $x'_1 x'_2 \cdots x'_{n/2}$ are different in exactly j; (j= 0, 1, 2, ..., $n/2$) digits. Digits in the address pattern $x_1 x_2 \cdots x_{n/2}$ with a corresponding digit in the pattern $x'_1 x'_2 \cdots x'_{n/2}$ make up k^2 combinations in k combinations of which those two digits are equal while in the other k^2-k combinations they are different. Therefore, the number of possible combinations that result in the patterns $x_1 x_2 \cdots x_{n/2}$ and $x'_1 x'_2 \cdots x'_{n/2}$ differ in exactly j digits is $\binom{n/2}{j} k^{n/2-j}(k^2-k)^j$ (j= 0, 1, 2, ..., $n/2$). The number of possible combinations where $x_1 x_2 \cdots x_n$ and $x'_1 x'_2 \cdots x'_n$ are different in exactly i digits (i= 0, 2, 4, ..., n) is given by

$$n_{m_i,even} = \binom{\frac{n}{2}}{\frac{i}{2}}\left(k^2-k\right)^{\frac{i}{2}} k^{\frac{n}{2}-\frac{i}{2}}, (i= 0, 2, 4, \ldots, n). \qquad (5.6)$$

Consider the case where n is odd. Examining the address patterns of the source $x_1 x_2 \cdots x_n$ and destination $x'_1 x'_2 \cdots x'_n$ address patterns for a matrix-transpose permutation shows that finding the number of combinations where these address patterns are different in exactly i digits (i= 0, 1, 2, ..., n) is equivalent to the problem of finding the number of i-digit radix-k address patterns where no two adjacent digits are equal and the first and last digits are also different. To compute such a number consider the following result from graph theory.

DEFINITION 5.1. *Chromatic polynomial* [175]: The chromatic polynomial of a graph is a function giving the number of ways that the graph nodes may be colored using a given number of colors such that no two neighboring nodes get the same color.

PROPOSITION 5.1. If G is a ring of length i, then the chromatic polynomial of G is

$$\Omega(i,C) = (C-1)^i + (-1)^i (C-1), \tag{5.7}$$

where C is the number of available colors [175].

Assume that each node in a ring of i nodes corresponds to a digit of an i-digit address pattern where each can be colored with color 0,1,..., or k-1 such that no two adjacent nodes have the same color. Equation 5.7 can be applied to find the number of address patterns meeting the conditions given above, by replacing C with k. Since these i digits can be chosen from n, the number of combinations in which the address patterns of the source and destination nodes of a matrix-transpose message differ in exactly i digits can be expressed as

$$n_{m_i,odd} = \binom{n}{i} \Omega(i,k) = \binom{n}{i} (k-1)^i + \binom{n}{i} (-1)^i (k-1). \tag{5.8}$$

Combining Equations 5.6 and 5.8 gives a general expression for the number of possible combinations where address patterns $x_1 x_2 \cdots x_n$ and $x'_1 x'_2 \cdots x'_n$ differ in exactly i digits (i= 0, 1,..., n), as

$$n_{m_i} = \begin{cases} 0, & \text{if } i \text{ is odd} \\ \binom{\frac{n}{2}}{\frac{i}{2}} \left(k^2 - k\right)^{\frac{i}{2}} k^{\frac{n}{2}-\frac{i}{2}}, & \text{if } n \text{ is even and } i \text{ is even} \\ \binom{n}{i}(k-1)^i + \binom{n}{i}(-1)^i (k-1), & \text{if } n \text{ is odd and } i \text{ is even} \end{cases} . \quad (5.9)$$

Thus, the probability that the source and destination addresses for a matrix-transpose message differ in i digits, P_{m_i}, can be written as

$$P_{m_i} = \frac{n_{m_i}}{N - n_{m_0}} = \begin{cases} 0, & \text{if } i \text{ is odd} \\ \dfrac{\binom{\frac{n}{2}}{\frac{i}{2}} \left(k^2 - k\right)^{\frac{i}{2}} k^{\frac{n}{2}-\frac{i}{2}}}{k^n - k^{\frac{n}{2}}}, & \text{if } n \text{ is even and } i \text{ is even} \\ \dfrac{\binom{n}{i}(k-1)^i + \binom{n}{i}(-1)^i (k-1)}{k^n - k}, & \text{if } n \text{ is odd and } i \text{ is even} \end{cases} . \quad (5.10)$$

Let us assume that the i-th digit (i= 0, 1,..., n) in the source address, x_i, is different from that of the destination address, i.e. x'_i. Considering all possible values that x_i and x'_i may take (i.e. $0 \le x_i, x'_i < k$) the average difference between x_i and x'_i, which is the average number of hops that a matrix-transpose message makes in the i-th dimension, is given by [6]

$$k_m = \frac{k-1}{2} . \quad (5.11)$$

The average number of hops that a matrix-transpose message makes across the network is

$$d_m = \sum_{i=1}^{n} i k_m P_{m_i} \ . \tag{5.12}$$

Examining the traffic generated by the matrix transpose permutation shows that a fraction n_{m_0} / N of the network nodes send uniform messages only and the remaining fraction $(1 - n_{m_0} / N)$ send a combination of uniform (with probability 1-m) and matrix-transpose messages (with probability m). Using Equations 5.3-5.12, the average number of hops, $\bar{d}$, that a message makes in the network is derived as

$$\bar{d} = \frac{n_{m_0}}{N} d_u + \left[1 - \frac{n_{m_0}}{N}\right](m d_m + (1-m) d_u)$$

$$= m\left[1 - \frac{n_{m_0}}{N}\right] d_m + \left[\frac{n_{m_0}}{N} + (1-m)\left(1 - \frac{n_{m_0}}{N}\right)\right] d_u \ , \tag{5.13}$$

where the uniform and matrix-transpose messages contribute with the following weights

$$\zeta_m = m\left[1 - \frac{n_{m_0}}{N}\right], \tag{5.14}$$

$$\zeta_u = \frac{n_{m_0}}{N} + (1-m)\left[1 - \frac{n_{m_0}}{N}\right]. \tag{5.15}$$

As adaptive routing uses any available channel to bring messages closer to their destinations, it distributes the rate of message traffic almost evenly among all network channels. Since a message makes, on average, $\bar{d}$ hops in the network, the total traffic existing in the network at a given time is $N \bar{d} \lambda_g$. Given that a router in the k-ary n-cube has n output channels the rate of messages arriving at each channel, λ_c, can be written as [6]

$$\lambda_c = \frac{N\bar{d}\lambda_g}{nN} = \frac{\bar{d}\lambda_g}{n}. \tag{5.16}$$

The uniform and matrix-transpose messages see different network latencies as they cross different channels to reach their destinations. If S_u and S_m denote the mean network latency for uniform and for matrix-transpose messages, respectively, the mean network latency taking into account both types of messages can be written as

$$\bar{S} = \zeta_m S_m + \zeta_u S_u . \tag{5.17}$$

Averaging over all possible cases for a matrix-transpose message, gives the mean network latency for matrix-transpose messages, S_m, as

$$S_m = \sum_{i=1}^{n} P_{m_i} S_{m_i} , \tag{5.18}$$

where S_{m_i} is the network latency for a matrix-transpose message whose source and destination address patterns differ in i digits. As a uniform message takes, on average, d_u hops to cross the network, the mean network latency for uniform messages, S_u, is given by

$$S_u = M + d_u + \sum_{j=1}^{d_u} P_{block_{u_j}} w_c , \tag{5.19}$$

where $P_{block_{u_j}}$ is the probability of blocking when a uniform message arrives at the j-th hop channel and w_c is the mean waiting time to acquire a virtual channel given that a message is blocked. The term $M + d_u$ in the above equation accounts for the message transmission time, while $P_{block_{u_j}} w_c$ accounts for the delay due to blocking at the j-th hop channel $(1 \le j \le d_u)$ along the message path. Similarly, the network latency for a matrix-transpose message, S_{m_i}, whose source and destination address patterns are different in i digits, is given by

$$S_{m_i} = M + ik_m + \sum_{j=1}^{ik_m} P_{block_{m_{i,j}}} w_c \,, \tag{5.20}$$

where $P_{block_{m_{i,j}}}$ is the probability of blocking when the matrix-transpose message arrives at the j-th hop channel. A message (uniform or matrix-transpose) is blocked at the j-th hop channel when all the adaptive virtual channels of the remaining dimensions to be visited and, in addition, the deterministic virtual channel of the lowest dimension to be visited are busy. To compute the probability of blocking, $P_{block_{u_j}}$, for a uniform message let us consider such a message that makes d_u hops across the network (k_u hops in each of n dimensions) and has arrived at the j-th hop channel along its path. The message may already have passed up to $(j-1)/k_u$ dimensions. If l, $0 \le l \le (j-1)/k_u$, dimensions are passed then there are still (n-l) dimensions to pass. Therefore, the probability of blocking can be expressed as

$$P_{block_{u_{l,j}}} = P_{pass_{u_{l,j}}} P_a^{n-l-1} P_{a\&d} \,. \tag{5.21}$$

In the above equation, $P_{pass_{u_{l,j}}}$ is the probability that l dimensions are passed at the j-th hop channel, P_a is the probability that all adaptive virtual channels of a physical channel are busy and $P_{a\&d}$ is the probability that all adaptive and deterministic virtual channels at a physical channel are busy. Since l may be 0, 1, ..., or $(j-1)/k_u$, the probability of blocking at the j-th hop channel is given by

$$P_{block_{u_j}} = \sum_{l=0}^{(j-1)/k_u} P_{block_{u_{l,j}}} \,. \tag{5.22}$$

The probability that l dimensions are passed at the j-th hop channel, $P_{pass_{u_{l,j}}}$, can be computed as follows. The number of combinations that l particular dimensions are passed is $\Gamma_0^{k_u-1}(j-lk_u, n-l)$. These l dimensions can be chosen from n dimensions in $\binom{n}{l}$ ways resulting in a total of $\binom{n}{l}\Gamma_0^{k_u-1}(j-lk_u, n-l)$ combinations. Dividing this by the total

number of combinations that j hops can be made over n dimensions will give the probability that a uniform message has passed l dimensions at its j-th hop as

$$P_{pass_{u_{l,i}}} = \frac{\binom{n}{l}\Gamma_0^{k_u-1}(j-lk_u, n-l)}{\Gamma_0^{k_u-1}(j,n)}. \tag{5.23}$$

Let us consider a matrix-transpose message passing i dimensions to reach its destination, i.e. a matrix-transpose massage whose source and destination addresses differ in i digits. Such a message makes ik_m hops over i dimensions. Adopting the same approach taken above for calculating $P_{block_{u_j}}$ for uniform messages we can derive $P_{block_{m_{i,j}}}$ as

$$P_{pass_{m_{l,i,j}}} = \frac{\binom{i}{l}\Gamma_0^{k_m-1}(j-lk_m, i-l)}{\Gamma_0^{k_m}(j,i)}, \tag{5.24}$$

$$P_{block_{m_{i,j}}} = \sum_{l=0}^{\frac{j-1}{k_m}} P_{block_{m_{l,i,j}}}, \tag{5.25}$$

$$P_{block_{m_{l,i,j}}} = P_{pass_{m_{l,i,j}}} P_a^{i-l-1} P_{a\&d},\ (0 \le l \le (j-1)/k_m). \tag{5.26}$$

Let P_v, $0 \le v \le V$, denote the probability that v virtual channels are busy at a physical channel. As in equations 3.14 and 3.15, the probabilities P_a and $P_{a\&d}$ can be expressed in terms of P_{v_j} as

$$P_a = P_V + \frac{2P_{V-1}}{\binom{V}{V-1}} + \frac{P_{V-2}}{\binom{V}{V-2}}, \tag{5.27}$$

$$P_{a\&d} = P_V + \frac{2P_{V-1}}{\binom{V}{V-1}}. \tag{5.28}$$

To determine the mean waiting time to acquire a virtual channel when a message is

blocked, w_c, an M/G/1 queue with an arrival rate of λ_c, and service time of $\overline{S}$, is used. The waiting time for such a queue can be expressed as [104]

$$w_c = \frac{\rho \overline{S}\left(1 + C_{\overline{S}}^2\right)}{2(1-\rho)}, \tag{5.29}$$

$$\rho = \lambda_c \overline{S}, \tag{5.30}$$

$$C_{\overline{S}}^2 = \frac{\sigma_{\overline{S}}^2}{\overline{S}^2}, \tag{5.31}$$

where $\sigma_{\overline{S}}^2$ is the variance of the service time distribution. Since the minimum service time is equal to the message length *M*, following a suggestion proposed in [58], the variance of the service time distribution can be approximated as

$$\sigma_{\overline{S}}^2 = (\overline{S} - M)^2. \tag{5.32}$$

As a result, the mean waiting time, w_c, to acquire a virtual channel when a message is blocked, given by Equation 5.29, becomes

$$w_c = \frac{\lambda_c \overline{S}^2 \left[1 + \frac{(\overline{S} - M)^2}{\overline{S}^2}\right]}{2(1-\lambda_c \overline{S})}. \tag{5.33}$$

The probability, P_v, that v adaptive virtual channels are busy at a physical channel, can be determined using a Markov chain as shown in Figure 3.1 with V+1 states: π_0, π_1,..., π_V. State π_v, $(0 \le v \le V)$, corresponds to v virtual channels being busy. The transition rate out of state π_v to state π_{v+1} is the traffic rate λ_c while the rate out of state π_v to state π_{v-1} is $\frac{1}{\overline{S}}$. The transition rate out of state π_V is reduced by λ_c to account for the arrival of

messages while a channel is in this state. The probability P_v can be computed using the steady-state equations as [49]

$$Q_v = \begin{cases} 1, & \text{if } v = 0 \\ Q_{v-1}\lambda_c \overline{S}, & \text{if } 0 < v < V \\ Q_{v-1}\dfrac{\lambda_c}{\frac{1}{\overline{S}} - \lambda_c}, & \text{if } v = V \end{cases}, \tag{5.34}$$

$$P_v = \begin{cases} 1/\sum_{i=0}^{V} Q_i, & \text{if } v = 0 \\ P_{v-1}\lambda_c \overline{S}, & \text{if } 0 < v < V \\ P_{v-1}\dfrac{\lambda_c}{\frac{1}{\overline{S}} - \lambda_c}, & \text{if } v = V \end{cases}. \tag{5.35}$$

In virtual channel flow control, multiple virtual channels share the bandwidth of a physical channel in a time-multiplexed manner. The average degree of multiplexing of virtual channels at a physical channel in the network is given by [49]

$$\overline{V} = \frac{\sum_{i=0}^{V} i^2 P_i}{\sum_{i=0}^{V} i P_i}. \tag{5.36}$$

The calculation of the mean waiting time, $\overline{W_s}$, at the local queue in the source node is realized in the same manner as that used for calculating the mean waiting time at a given network channel. The local queue is treated as an M/G/1 queue with an arrival rate of λ_g / V (recall that a message in the source node can enter the network through any of the V virtual channels), a service time of $\overline{S}$, and thus a mean waiting time of [104]

$$\overline{W_s} = \frac{\frac{\lambda_g}{V}\overline{S}^2\left[1+\frac{(\overline{S}-M)^2}{\overline{S}^2}\right]}{2(1-\frac{\lambda_g}{V}\overline{S})}. \tag{5.37}$$

5.1.2 The hypercube case

For an n-dimensional hypercube network (2-ary n-cube) the above model changes slightly as follows. Equation 5.5 giving the probability that a new-generated uniform message is an i-hop message, is now given by [3]

$$P_{u_i} = \frac{\binom{n}{i}}{N-1}. \tag{5.38}$$

As a digit in the hypercube is a bit, each digit difference between two nodes' address patterns also means a distance of one hop between the two nodes. Hence, the average distance traversed by a matrix-transpose message in the network (given by equation 5.12) is now given by

$$d_m = \sum_{i=1}^{n} iP_{m_i}. \tag{5.39}$$

The network latency for a message consists of the message transmission time and the delay due to blocking in the network. Therefore, we can write S_{m_i} (given by equation 5.20) as

$$S_{m_i} = M + i + \sum_{j=1}^{i} P_{block_{m_i,j}}\overline{S}. \tag{5.40}$$

The probability of blocking when an i-hop matrix-transpose message arrives at the j-th hop channel is now given by

$$B_{block_{m_{i,j}}} = P_a{}^{i-j} P_{a\&d}\,, \tag{5.41}$$

where P_a is the probability that all adaptive virtual channels of a physical channel are busy and $P_{a\&d}$ is the probability that all adaptive and deterministic virtual channels of a physical channel are busy.

Similarly the probability of blocking when a typical uniform message (which is a d_u -hop message) arrives at the j-th hop channel is now given by

$$B_{block_{u_j}} = P_a{}^{d_u - j} P_{a\&d}\,. \tag{5.42}$$

The probabilities P_a and $P_{a\&d}$ can now be computed by [36]

$$P_a = P_V + \frac{P_{V-1}}{\binom{V}{V-1}}, \tag{5.43}$$

$$P_{a\&d} = P_V\,. \tag{5.44}$$

5.2 The analytic model for digit-reversal traffic

The model for digit-reversal traffic pattern uses almost the same notation and assumptions used for the matrix-transpose traffic pattern. However, to define the digit-reversal traffic assumptions *a* and *b* should change as follows.

a) There are two types of traffic in the network: "uniform" and "digit-reversal". In the uniform traffic pattern, a message is sent to any other node in the network with equal probability. In the traffic pattern generated according to the digit-reversal permutation [59, 79], a message generated in the source node $X = x_1 x_2 \cdots x_n$ is sent to the node $D(X) = x_n x_{n-1} \cdots x_1$. Let us refer to these two types of messages as

uniform and digit-reversal messages, respectively. When a message is generated it has a finite probability α of being a digit-reversal message and probability $(1-\alpha)$ of being uniform. When $\alpha = 0$, the traffic pattern is purely uniform while $\alpha = 1$ defines a pure digit-reversal traffic.

b) Nodes generate traffic independently of each other, and which follows a Poisson process with a mean rate of λ_g messages/cycle. Therefore, the message generation rate of the uniform and digit-reversal traffics are respectively $(1-\alpha)\lambda_g$ and $\alpha\lambda_g$.

The mean message latency is the sum of the mean network latency, $\overline{S}$, the time to cross the network, and the mean waiting time seen by a message in the source node, $\overline{W_s}$, both scaled by $\overline{V}$, the average degree of virtual channel multiplexing that takes place at a physical channel, i.e.

$$Latency = (\overline{S} + \overline{W_s})\overline{V}. \tag{5.45}$$

Examining the address patterns generated by digit-reversal permutations reveals that we need to consider even and odd values of n separately when computing the different quantities, $\overline{S}$, $\overline{W_s}$, and $\overline{V}$. This is because when n is even all network channels receive both uniform and digit-reversal traffic. However, when n is odd not all channels receive both types of messages. While the channels associated with the centre dimension (dimension $(n+1)/2$) receive uniform messages only, channels at the other dimensions (1, 2, ..., $(n-1)/2$, $(n+1)/2+1$, ..., n) receive the uniform as well as digit-reversal messages.

5.2.1 Outline of the model when *n* is even

As mentioned above in assumptions we use the digit-reversal permutation function $\Gamma(X)$ (instead of $M(X)$ in the model described above for the matrix-transpose traffic) and thus

use digit-reversal traffic portion parameter α (instead of parameter m used for generating a traffic pattern including matrix-transpose and uniform traffic patterns). When the number of dimensions, n, is even, the analysis is similar to that of matrix-transpose traffic pattern with even n. The model in this case can be obtained by simply changing all indices m in the matrix-transpose model to d.

5.2.2 Outline of the model when *n* is odd

As explained above, when n is odd, channels belonging to dimension $(n+1)/2$ receive uniform messages only. The traffic due to digit-reversal messages falls only on the channels belonging to dimensions 1, 2, ..., $(n-1)/2$, $(n+1)/2+1$, ..., n. Let us refer to dimension $(n+1)/2$ as the "centre-dimension" and the channels belonging to this dimension as the "centre-channels". Similarly, let us refer to other dimensions as "other-dimensions" and their associated channels as the "other-channels". In subsections 5.2.2.1 to 5.2.2.6, required changes in the model are discussed.

5.2.2.1 Calculation of the number of ways that two addresses are different in *i* digits

When n is odd the digit $x_{(n+1)/2}$ in the address X is equal to the digit $x'_{(n+1)/2}$ in $\mathrm{D}(X)$. As a result, the number of combinations, where the address patterns X and $\mathrm{D}(X)$ are different in i digits, is multiplied by k to account for all possible values that digit $x_{(n+1)/2}$ may have. Therefore, the number of possible combinations where $x_1 x_2 \cdots x_n$ and $x'_1 x'_2 \cdots x'_n$ are different in exactly i digits (i= 0, 2, 4,..., n) is given by $\binom{\frac{n-1}{2}}{\frac{i}{2}} \left(k^2 - k\right)^{\frac{i}{2}} k^{\frac{n-1}{2}-\frac{i}{2}+1} = \binom{\frac{n-1}{2}}{\frac{i}{2}} \left(k - 1\right)^{\frac{i}{2}} k^{\frac{n+1}{2}}$. The number of combinations, where the address patterns X and $\mathrm{D}(X)$ are different in i digits for i=0, 1, ..., n, is therefore given by

$$n_{d_i} = \begin{cases} \binom{\frac{n-1}{2}}{\frac{i}{2}} (k-1)^{\frac{i}{2}} k^{\frac{n+1}{2}}, & \text{if } i \text{ is even} \\ 0, & \text{otherwise} \end{cases}. \quad (5.46)$$

5.2.2.2 Calculation of the probability of blocking

Let $P_{a_{other}}$ and $P_{a\&d_{other}}$ define the probability that all adaptive virtual channels at an other-channel are busy and the probability that all adaptive and deterministic virtual channels at an other-channel are busy. Similarly, let $P_{a_{centre}}$ and $P_{a\&d_{centre}}$ define the probability that all adaptive virtual channels at a centre-channel are busy and the probability that all adaptive and deterministic virtual channels at a centre-channel are busy. When l dimensions are passed, the remaining n-l dimensions may make several combinations. The probability that the centre-channel is already passed is $\binom{n-1}{l-1} / \binom{n}{l}$. Therefore, the blocking probability after passing l dimensions is $\left[\binom{n-1}{l-1} / \binom{n}{l}\right] P_{a_{other}}^{n-l-1} P_{a\&d_{other}}$. If the centre-channel has not been passed yet, with a probability of $1 - \binom{n-1}{l-1} / \binom{n}{l}$, two cases may arise. First, the centre-channel is passed as last dimension with a probability of $1/(n-l)$ for which the blocking probability becomes $1/(n-l)\left[1 - \binom{n-1}{l-1} / \binom{n}{l}\right] P_{a_{other}}^{n-l-1} P_{a\&d_{centre}}$. Second, the centre-channel is not be passed as the last dimension, with a probability of $(n-l-1)/(n-l)$; in this case the probability of blocking becomes $(n-l-1)/(n-l)\left[1 - \binom{n-1}{l-1} / \binom{n}{l}\right] P_{a_{other}}^{n-l-2} P_{a_{centre}} P_{a\&d_{other}}$. Putting all these cases together will result in the probability of blocking, for the uniform message, when it has already passed l dimensions, as

$$P_{block_{u_{l,i}}} = P_{pass_{u_{l,i}}} \left[\frac{\binom{n-1}{l-1}}{\binom{n}{l}} P_{a_{other}}^{n-l-1} P_{a\&d_{other}} + \left(1 - \frac{\binom{n-1}{l-1}}{\binom{n}{l}}\right) \left(\frac{1}{(n-l)} P_{a_{other}}^{n-l-1} P_{a\&d_{center}} + \frac{(n-l-1)}{(n-l)} P_{a_{other}}^{n-l-2} P_{a_{center}} P_{a\&d_{other}} \right) \right]. \quad (5.47)$$

5.2.2.3 Calculation of the probabilities of virtual channel occupancy for other/centre-channels

Adapting the same method as for deriving Equations 3.14 and 3.15, we can express the probability that all adaptive virtual channels of a centre-channel are busy, $P_{a_{centre}}$, the probability that all adaptive and deterministic virtual channels of a centre-channel are busy, $P_{a\&d_{centre}}$, the probability that all adaptive virtual channels of an other-channel are busy, $P_{a_{other}}$, and the probability that all adaptive and deterministic virtual channels of an other-channel are busy, $P_{a\&d_{other}}$, all in terms of $P_{V_{centre}}$ and $P_{V_{other}}$ as

$$P_{a_{centre}} = P_{V_{centre}} + \frac{2P_{V-1_{centre}}}{\binom{V}{V-1}} + \frac{P_{V-2_{centre}}}{\binom{V}{V-2}}, \tag{5.48}$$

$$P_{a\&d_{centre}} = P_{V_{centre}} + \frac{2P_{V-1_{centre}}}{\binom{V}{V-1}}, \tag{5.49}$$

$$P_{a_{other}} = P_{V_{other}} + \frac{2P_{V-1_{other}}}{\binom{V}{V-1}} + \frac{P_{V-2_{other}}}{\binom{V}{V-2}}, \tag{5.50}$$

$$P_{a\&d_{other}} = P_{V_{other}} + \frac{2P_{V-1_{other}}}{\binom{V}{V-1}}. \tag{5.51}$$

5.2.2.4 Calculation of the traffic rate on network channels

While all channels receive uniform traffic, only the channels belonging to other-dimensions receive digit-reversal traffic. Therefore, when n is odd the traffic rate arriving at each centre-channel, $\lambda_{c_{centre}}$ and each other-channel, $\lambda_{c_{other}}$, can be expressed as

$$\lambda_{c_{center}} = \frac{\zeta_u d_u \lambda_g}{n}, \tag{5.52}$$

$$\lambda_{c_{other}} = \lambda_{c_{center}} + \frac{\zeta_d d_d \lambda_g}{n-1}. \tag{5.53}$$

5.2.2.5 Calculation of the mean waiting times at a network channel

The mean waiting time for a centre-channel and an other-channel, $w_{t_{centre}}$ and $w_{t_{other}}$, can be expressed as

$$w_{t_{centre}} = \frac{\lambda_{c_{centre}} S_u^2 \left[1 + \frac{(S_u - M)^2}{S_u^2}\right]}{2(1 - \lambda_{c_{centre}} S_u)}, \tag{5.54}$$

and

$$w_{t_{other}} = \frac{\lambda_{c_{other}} S_{t_{other}}^2 \left[1 + \frac{(S_{t_{other}} - M)^2}{S_{t_{other}}^2}\right]}{2(1 - \lambda_{c_{other}} S_{t_{other}})}. \tag{5.55}$$

where S_u (given by Equation 5.19) and $S_{t_{other}}$ (calculated below) are approximated values for service time of a centre-channel and an other-channel, respectively. Therefore, the mean waiting time for a channel taking both types into account would be

$$w_t = \frac{1}{n} w_{t_{centre}} + \left[1 - \frac{1}{n}\right] w_{t_{other}}. \tag{5.56}$$

The mean service time for an other-channel, $S_{t_{other}}$, can be approximated as

$$S_{t_{other}} = \frac{\lambda_{c_{centre}}}{\lambda_{c_{other}}} S_u + \left[1 - \frac{\lambda_{c_{centre}}}{\lambda_{c_{other}}}\right] S_d. \tag{5.57}$$

5.2.2.6 Calculation of the average degree of virtual channels multiplexing

Adapting the approach used to calculate P_v when n is even, we can write the expression of the probability of having v busy virtual channels at a centre-channel, $P_{v_{centre}}$, and at an other-channel, $P_{v_{other}}$, as follows.

$$Q_{v_{centre}} = \begin{cases} 1, & \text{if } v = 0 \\ Q_{v-1_{centre}} \lambda_{c_{centre}} S_u, & \text{if } 0 < v < V \\ Q_{v-1_{centre}} \dfrac{\lambda_{c_{centre}}}{\frac{1}{S_u} - \lambda_{c_{centre}}}, & \text{if } v = V \end{cases}, \tag{5.58}$$

$$P_{v_{centre}} = \begin{cases} 1/\sum_{i=0}^{V} Q_{i_{centre}}, & \text{if } v = 0 \\ P_{v-1_{centre}} \lambda_{c_{centre}} S_u, & \text{if } 0 < v < V \\ P_{v-1_{centre}} \dfrac{\lambda_{c_{centre}}}{\frac{1}{S_u} - \lambda_{c_{centre}}}, & \text{if } v = V \end{cases}, \tag{5.59}$$

$$Q_{v_{other}} = \begin{cases} 1, & \text{if } v = 0 \\ Q_{v-1_{other}} \lambda_{c_{other}} S_{t_{other}}, & \text{if } 0 < v < V \\ Q_{v-1_{other}} \dfrac{\lambda_{c_{other}}}{\frac{1}{S_{t_{other}}} - \lambda_{c_{otehr}}}, & \text{if } v = V \end{cases}, \tag{5.60}$$

and

$$P_{v_{other}} = \begin{cases} 1/\sum_{i=0}^{V} Q_{i_{other}}, & \text{if } v = 0 \\ P_{v-1_{other}} \lambda_{c_{otehr}} S_{t_{other}}, & \text{if } 0 < v < V \\ P_{v-1_{otehr}} \dfrac{\lambda_{c_{other}}}{\frac{1}{S_{t_{other}}} - \lambda_{c_{other}}}, & \text{if } v = V \end{cases}. \tag{5.61}$$

where $S_{t_{other}}$ is the mean service time for an other-channel (given by Equation 5.57). The average degree of multiplexing of virtual channels belonging to a centre-channel and to an other-channel in the network, and the total average multiplexing degree of virtual channels in the network, are given by

$$\overline{V}_{centre} = \frac{\sum_{i=0}^{V} i^2 P_{i_{centre}}}{\sum_{i=0}^{V} i P_{i_{centre}}}, \tag{5.62}$$

$$\overline{V}_{other} = \frac{\sum_{i=0}^{V} i^2 P_{i_{other}}}{\sum_{i=0}^{V} i P_{i_{other}}}, \tag{5.63}$$

$$\overline{V} = \frac{1}{n}\overline{V}_{center} + \left[1 - \frac{1}{n}\right]\overline{V}_{other}. \tag{5.64}$$

5.2.3 The hypercube model

When the network is hypercube (k=2), if the dimensionality of the network n is even similar changes to those made on the matrix-transpose model with even n must be applied. For odd n, we may adopt the proposed model for odd n with similar changes made to the model for hypercube with even n. Only Equation 5.47, giving the probability of blocking, should now change to

$$P_{block_{u_i}} = \frac{\binom{n-1}{i-1}}{\binom{n}{i}} P_{a_{other}}^{n-i-1} P_{a\&d_{other}} + \left(1 - \frac{\binom{n-1}{i-1}}{\binom{n}{i}}\right)\left(\frac{1}{(n-i)} P_{a_{other}}^{n-i-1} P_{a\&d_{center}} + \frac{(n-i-1)}{(n-i)} P_{a_{other}}^{n-i-2} P_{a_{center}} P_{a\&d_{other}}\right). \tag{5.65}$$

5.3 Validation of the models

The above model has been validated through a discrete-event simulator that mimics the behaviour of Duato's fully adaptive routing at the flit level in k-ary n-cubes. In each simulation experiment, a total number of 100K messages is delivered. Statistics gathering was inhibited for the first 10K messages to avoid distortions due to the initial startup conditions. 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, and the mean queueing time at the source node, the time spent at the local queue before entering the first network channel.

Numerous experiments have been performed for several combinations of network sizes, message lengths, digit-reversal traffic fractions, and number of virtual channels to validate the model. However, for the sake of specific illustration, Figures 5.1-5.5 depict latency results predicted by the proposed models plotted against those provided by the simulator for an 8-ary 2-cube, an 8-ary 3-cube, a 7-dimensional hypercube and an 8-dimensional hypercube with M=32 and 64 flits. Moreover, the number of virtual channels per physical channel was set to V=2, 3, 4, or 5 and the fraction of matrix-transpose and digit (bit)-reversal messages was assumed to be m, α = 0.1, 0,2, 0.6, 0.7 or 0.8. We have tried to include a wide range of parameters (for V, m, α) getting different values in different scenarios.

The horizontal axis in each figure shows the traffic generation rate at each node (λ_g) while the vertical axis shows the mean message latency. Figure 5.1 shows the average latency versus message generation traffic in an 8-ary 2-cube for V=3 and 5 virtual channels per physical channel, message length M=32 and 64 flits, and matrix-transpose

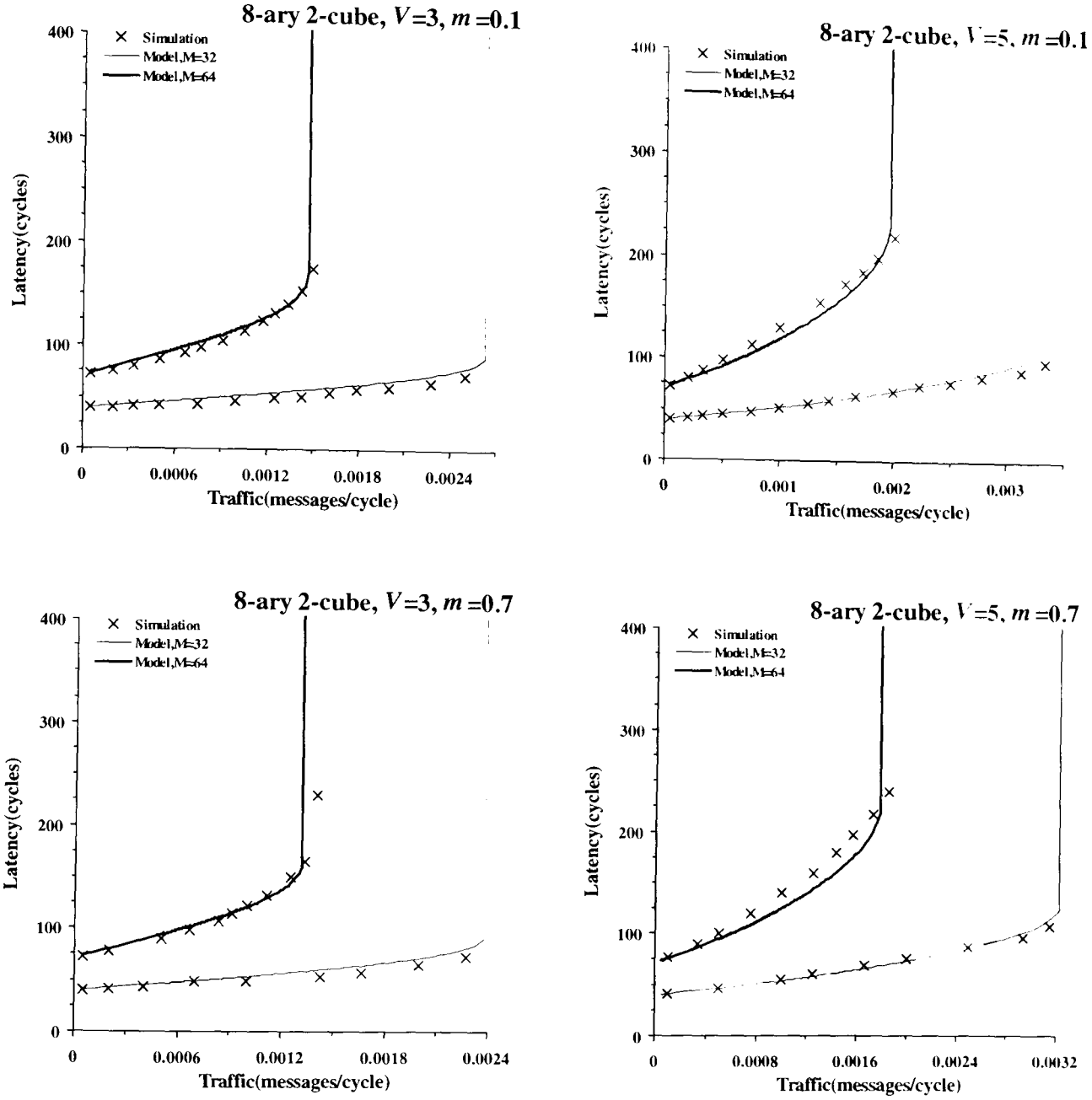


Figure 5.1: The average message latency predicted by the model against simulation results in an 8-ary 2-cube for V=3 and 5 virtual channels per physical channel, and message length M=32 and 64 flits, with matrix-transpose traffic portions m=0.1 and 0.7.

traffic portions m=0.1 and 0.7. Note that since n=2, we have $\mathcal{M}(X)=I\lambda(X)$ and therefore this figure is also valid for digit-reversal traffic pattern.

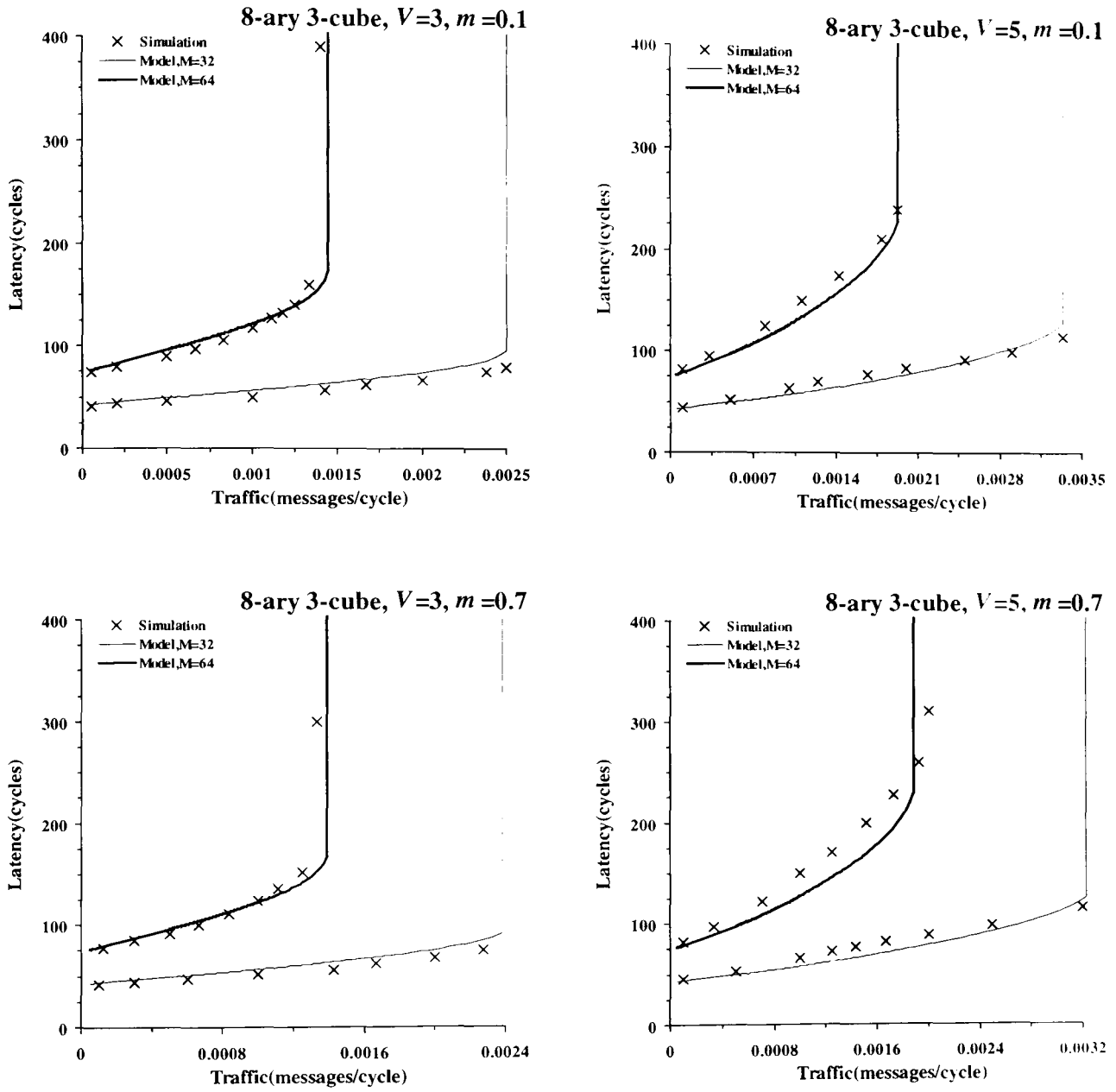


Figure 5.2: The average message latency predicted by the model against simulation results in an 8-ary 3-cube for V=3 and 5 virtual channels per physical channel, and message length M=32 and 64 flits, with matrix-transpose traffic portions m=0.1 and 0.7.

Figure 5.2 illustrates the average latency versus message generation traffic in an 8-ary 3-cube for V=3 with 5 virtual channels per physical channel, message length M=32 and 64 flits, and matrix-transpose traffic portions m=0.1 and 0.7. Figure 5.3 shows the average

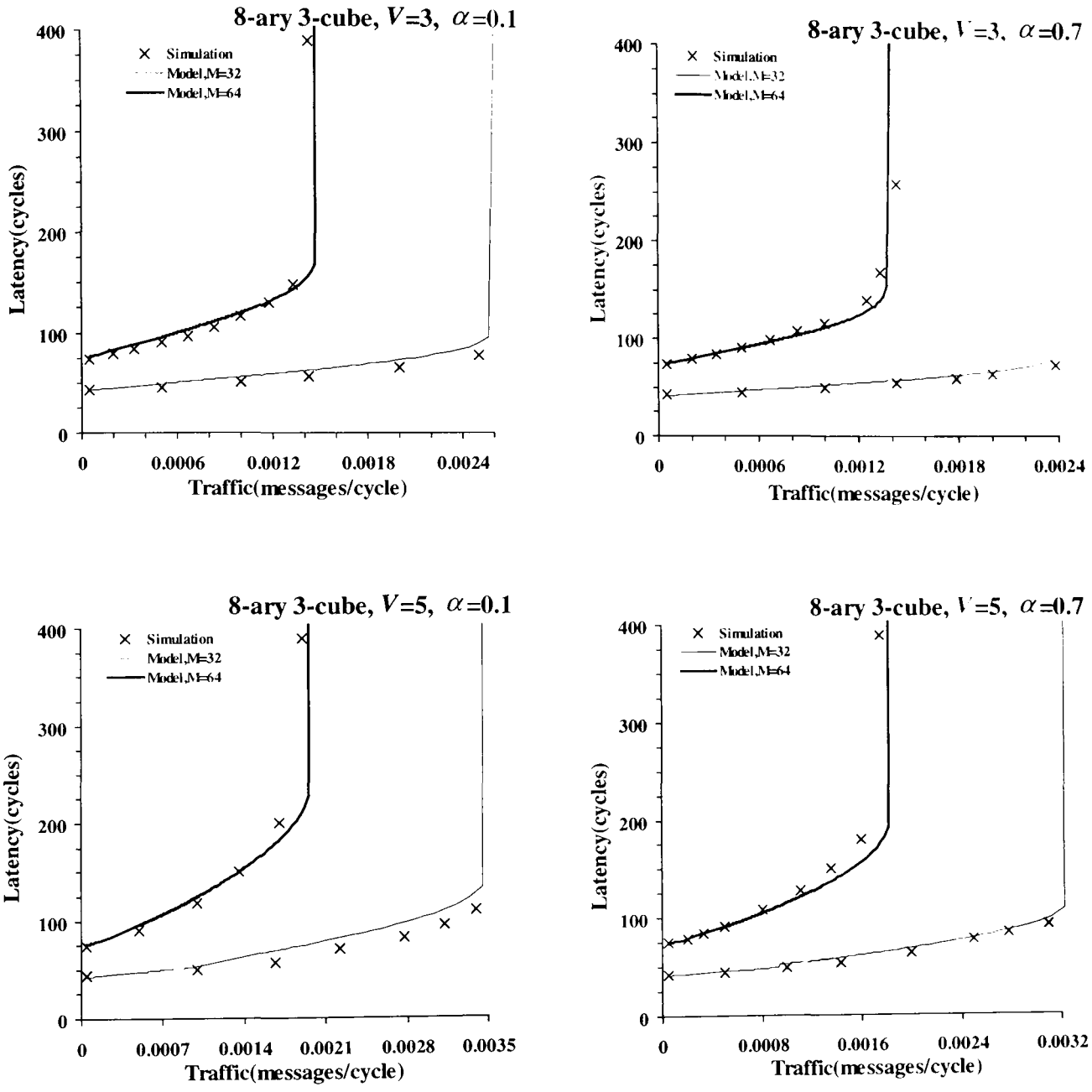


Figure 5.3: The average message latency predicted by the model against simulation results in an 8-ary 3-cube for V=3 and 5 virtual channels per physical channel, and message length M=32 and 64 flits, with digit-reversal traffic portions α=0.1 and 0.7.

latency versus message generation traffic for the same scenario but for digit-reversal traffic pattern with α=0.1 and 0.7.

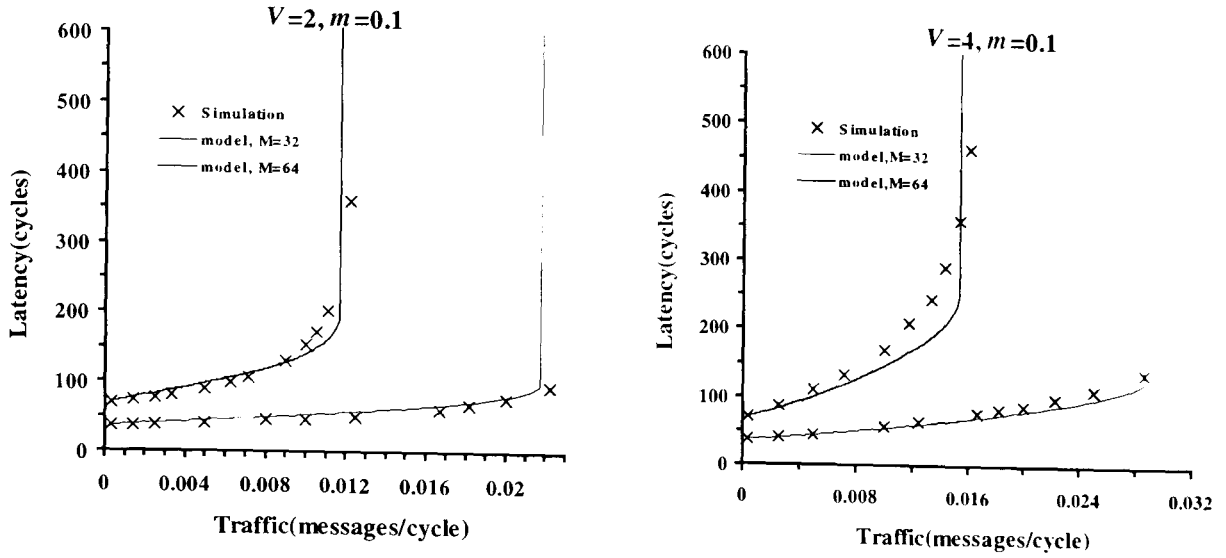


Figure 5.4: The average message latency predicted by the model against simulation results in an 8-dimensional hypercube with V=2 and 4 virtual channels per physical channel, message length M = 32 and 64 flits and matrix-transpose traffic potions m= 0.1 and 0.8.

Figure 5.4 shows mean message latency predicted by the analytical model against simulation results in an 8-dimensional hypercube, for message length M = 32 and 64 flits, number of virtual channels V=2 and 4, and matrix transpose traffic portion m= 0.1 and 0.8.

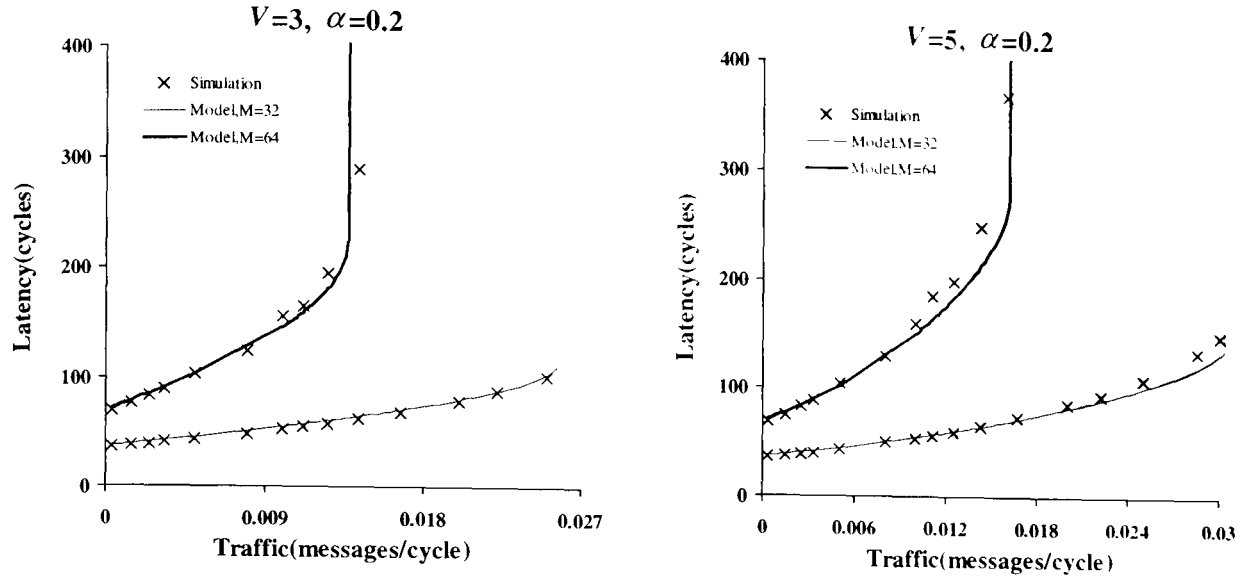


Figure 5.5: The average message latency predicted by the model against simulation results in a 7-dimensional hypercube with V=3 and 5 virtual channels per physical channel, message length M=32 and 64 flits, and bit-reversed traffic portions α=0.2 and 0.6.

Finally, Figure 5.5 shows mean message latency predicted by the analytical model against simulation results in a 7-dimensional hypercube, for message length M = 32 and 64 flits, number of virtual channels V=3 and 5, and bit-reversal traffic portion α= 0.2 and 0.6.

The figures reveal that in all cases, the analytical model predicts the mean message latency with a good degree of accuracy in the steady state regions. However, some discrepancies around the saturation point are apparent. This is due to the approximations made to ease the derivation of the model such as in the estimation of the variance of service time distribution at a channel. Such an approximation greatly simplifies the model as it allows us to avoid computing the exact distribution of message service time at a given channel, which is not a straightforward task due to the interdependencies between service times at successive channels caused by the reliance of wormhole routing on a blocking mechanism for flow control. However, the simplicity of the model makes it a practical evaluation tool that gives insight into the performance behavior of fully adaptive routing in k-ary n-cube interconnection networks.

It is worth noting that latency results for different values of m and α reveal that matrix-transpose or digit-reversal traffic patterns has a little impact on the mean message latency since adaptive routing is able to exploit alternative paths of the k-ary n-cube to route blocked messages, and as a result it can distribute traffic load approximately evenly among the network channels.

5.4 Considering bidirectional networks

When the network is bidirectional some equations in the models described above should change as follows. The network diameter d_{max} is $n\left\lfloor \frac{k}{2} \right\rfloor$. The probability that a uniform message is an i-hop message is given, using Theorem 2.4, to be $\frac{\overleftrightarrow{A}_n^k(i)}{N-1}$. Therefore,

$$P_{u_i} = \frac{1}{N-1}\begin{cases} \sum_{m=0}^{n}\sum_{l=0}^{m}(-1)^l 2^m \binom{n}{m}\binom{m}{l}\binom{i-\frac{kl}{2}-2l-1}{m-1}, & k \text{ is odd} \\ \sum_{t=0}^{n}\sum_{m=0}^{n-t}\sum_{l=0}^{m}(-1)^l 2^m \binom{n}{t}\binom{n-t}{m}\binom{m}{l}\binom{i-\frac{k(l+t)}{2}-2l-1}{m-1} & \text{otherwise} \end{cases} \quad (5.66)$$

The average number of hops that a matrix-transpose or a digit-reversal message takes in each dimension are now

$$k_d = k_m = \begin{cases} \frac{k}{4}, & \text{if } k \text{ is even} \\ \\ \frac{1}{4}(k - \frac{1}{k}), & \text{if } k \text{ is odd} \end{cases} . \qquad (5.67)$$

Since a router in the k-ary n-cube has $2n$ output channels, the rate of messages received by each channel, λ_c, is now given by [6]

$$\lambda_c = \frac{\lambda_g \overline{d}}{2n} . \qquad (5.68)$$

Recall that, when developing the models for unidirectional k-ary n-cubes, we assumed an almost equal traffic on network channels. Validation experiments confirmed that this was an acceptable approximation with adaptive routing in unidirectional k-ary n-cubes. However, with bidirectional networks such an assumption may result in inaccurate predictions especially for high traffic generation rates.

5.5 Analysis

The proposed analytical models are now used to study the performance merits of the k-ary n-cube with adaptive routing and virtual channels under the non-uniform traffic posed by matrix-transpose and digit-reversal permutations. We have repeated the analysis of Chapter 3 (for uniform traffic) and observed the same results in all cases. This was predictable since, in the validation section (Section 5.3), we saw that the effect of matrix-transpose and digit-reversal traffic patterns on network performance is small. However, let us consider two networks, a 10-ary 4-cube and a 10-ary 5-cube, as examples with even and

odd dimensionality (n) and examine the effect of a non-uniform traffic portion (m and α) on network performance. We have used these networks for the sake of the present discussion, but the conclusions reached here have been found to be universally valid.

Figure 5.6 shows the saturation traffic rate against matrix-transpose traffic portion (m) in a unidirectional 10-ary 4-cube (network with even n), with V=4 virtual channels per physical channel, and message length M=50 and 200 flits. The network enters the saturation region when $\rho \geq 1$ (Equation 5.30); the corresponding λ_g for which $\rho \geq 1$ is satisfied, is the saturation traffic rate. As can be seen from the figure, the effect of the matrix-transpose traffic portion is negligible especially for long messages (M=200 flits). The same curves are obtained when considering digit-reversal traffic pattern, since n is even.

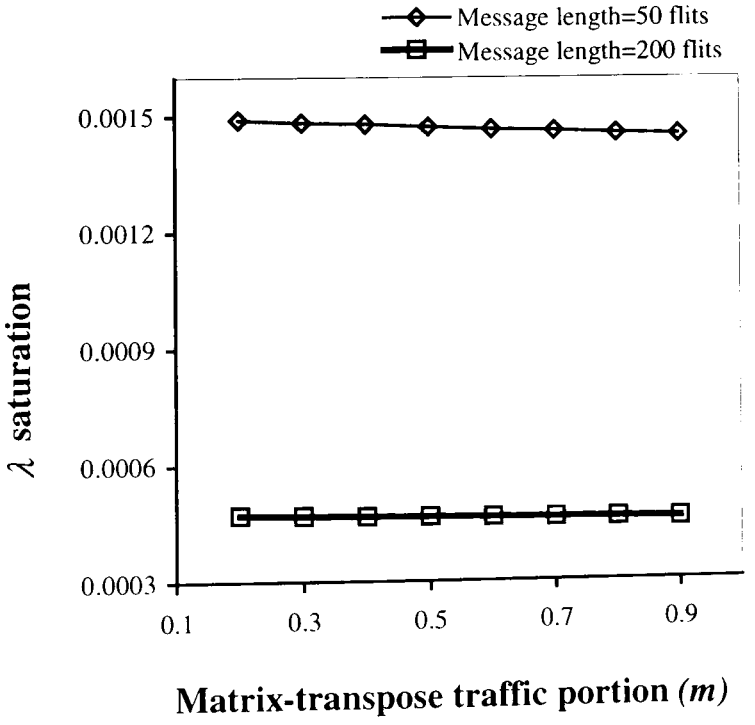
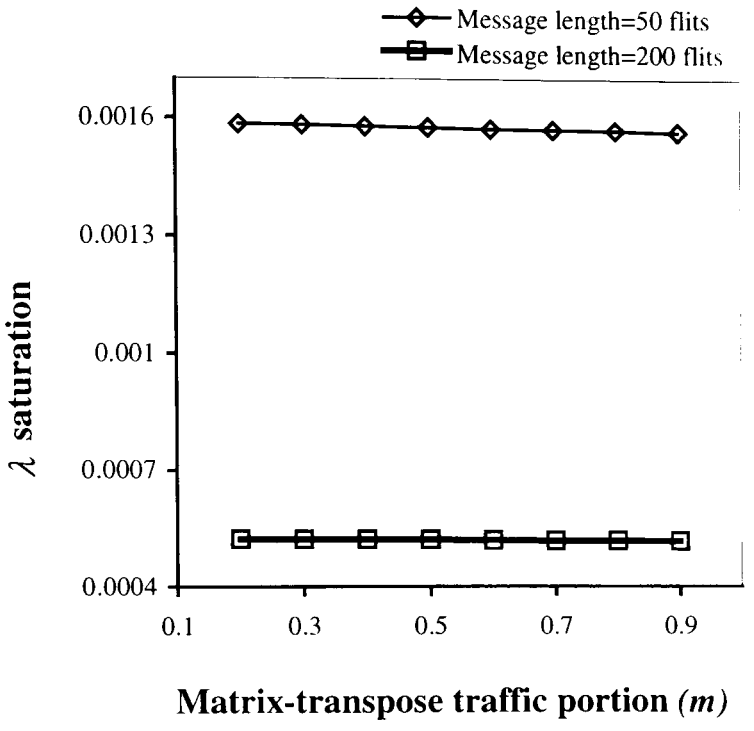
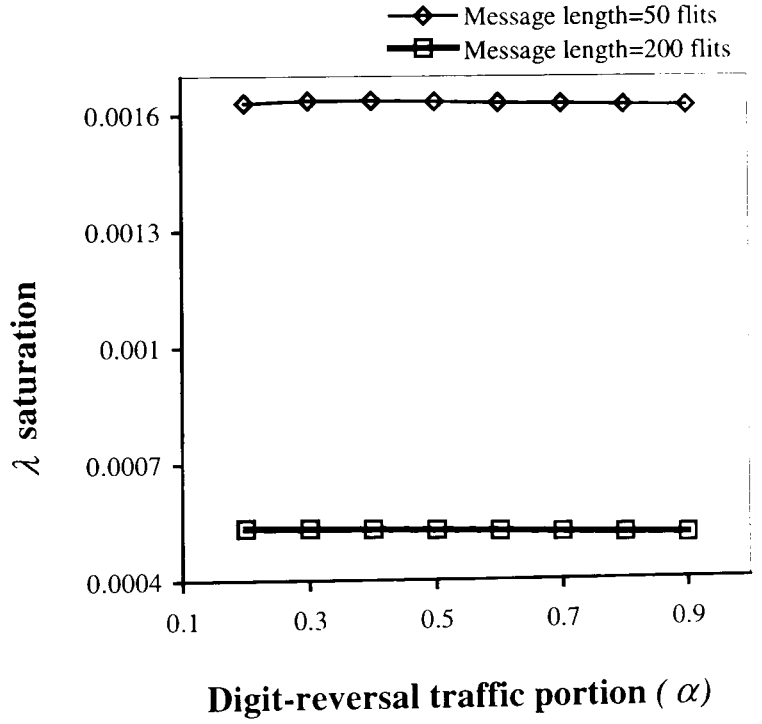


Figure 5.6: The saturation traffic rate versus matrix-transpose traffic portion (m) in a unidirectional 10-ary 4-cube, with V=4 virtual channels per physical channel and message length M=50 and 200 flits. Note that the curves for the digit-reversal traffic pattern are the same since n is even.



(a)



(b)

Figure 5.7: The saturation traffic rate versus permutation traffic portions (m or α) in a unidirectional 10-ary 5-cube, with V=5 virtual channels per physical channel, and message length M=50 and 200 flits; (a) in the presence of matrix-transpose traffic, (b) in the presence of digit-reversal traffic.

A similar trend can be seen in Figure 5.7 for a 10-ary 5-cube (network with odd n) with V=5 virtual channels per physical channel and message lengths of M=50 and 200 flits in the presence of matrix-transpose and digit-reversal traffic patterns.

These figures again confirm that the effect of non-uniformity posed by matrix-transpose and digit-reversal traffic patterns on network performance is almost negligible since adaptive routing enables the blocked messages to advance towards their destination using free alternative paths.

5.6 Conclusions

This chapter has presented analytical models for computing message latency in wormhole-switched k-ary n-cubes with fully adaptive routing in the presence of traffic generated by the matrix-transpose and digit-reversal permutations used in many parallel applications (e.g., matrix problems and signal processing). Simulation experiments have confirmed that the latency results predicted by the analytical models are in good agreement with those obtained through simulation experiments. The results show that matrix-transpose and digit-reversal traffic portions do not have a large impact on overall network performance when fully adaptive routing algorithms are used in unidirectional k-ary n-cubes.

In the next chapter, we shall use the models proposed in Chapters 3, 4 and 5 to compare the performance of k-ary n-cubes under two well-known technological constraints: constant bisection bandwidth and pin-out. To do so, we need a new cost-performance model, the first to consider the costs of both network channels and internal router hardware.

Chapter 6

Performance Comparison of Multi-dimensional *k*-Ary *n*-Cubes

An extensive examination of interconnection networks has been conducted over the last decade, both with a view to studying fundamental graph-theoretic properties and feasibility of implementation in various technologies [4, 6, 10, 20, 35, 44, 62, 85, 157]. The latter consideration is of crucial importance since in practice implementation technology puts bandwidth constraints on network channels, and these are important factors in determining how well the theoretical properties of a particular network topology can be exploited. When systems are implemented on a single VLSI-chip, the *wiring density* of the network determines the overall system cost and performance [48]. For instance, Dally [44] has shown that under the constant wiring density constraint (with constant bisection bandwidth), the 2D torus outperforms the hypercube. This is because the former topology has wider channels, thus higher channel bandwidth, that compensate for its higher diameter.

Other researchers, including Abraham [4] and Agrawal [6], have conducted similar studies to Dally's and arrived at the same conclusion. However, they have also argued that while the wiring density constraint is certainly applicable where an entire network is

implemented on a single VLSI-chip, this is not the case in the currently more realistic situation where a network has to be partitioned over many chips. In such circumstances, they have identified that the most critical bandwidth constraint is imposed by the chip's I/O pins through which any data entering or leaving the chip must travel. Abraham [4] and Agrawal [6] have concluded that it is the hypercube which exhibits better performance under such a pin-out constraint. All these studies [4, 6, 10, 20, 44, 157] have used *deterministic* routing. Moreover, they have all taken account of network channel costs while ignoring those associated with the internal hardware of the routers.

This chapter examines the relative performance merits of the torus and hypercube with fully adaptive wormhole routing in the presence of different traffic patterns, namely uniform, hotspot, matrix-transpose and digit-reversal. To do so, we use the analytical models proposed in Chapters 3, 4 and 5, of fully adaptive routing in wormhole-switched k-ary n-cubes in the presence of uniform, hotspot, matrix-transpose and digit-reversal traffic patterns. The comparison is conducted under constant bisection bandwidth and pin-out constraints and for both pipelined and non-pipelined wire delay models[1] [157].

The rest of the chapter is organised as follows. Section 6.1 briefly gives the assumptions made in comparison. Section 6.2 defines a new cost-performance model while section 6.3 uses the proposed cost-performance model and compares the performance merits of the torus and hypercube under both the constant bisection bandwidth and pin-out constraints, and considering pipelined and non-pipelined wire delay models, in the presence of different traffic patterns. Finally, Section 6.4 concludes this chapter.

6.1 Assumptions

We make the following assumptions, widely used in the literature [3-6, 10, 29-31, 38, 39,

[1] We talk about these wire delay models in Section 6.2.2.

44, 45, 54, 77, 80-82, 95, 114, 135, 136].

a) The uniform and non-uniform (hotspot, matrix-transpose and digit-reversal) traffic patterns are considered.

b) Nodes generate traffic independently of each other, which follows a Poisson process with a mean rate of λ_g messages/cycle. In case of non-uniform traffic, two non-uniform and uniform portions, $x\lambda_g$ and $(1-x)\lambda_g$, are presented where x may be replaced by h for hotspot, m for matrix-transpose and α for digit-reversal traffic patterns.

c) Message length is fixed and equal to M flits.

d) The channel cycle time and switch internal delay are assumed to be respectively, t_c and t_s clock cycles.

e) The local queue at the injection channel in the source node has infinite capacity. Moreover, messages are transferred to the local PE as soon as they arrive at their destinations through the ejection channel.

f) V virtual channels are used per physical channel, divided in two groups VC_1 and VC_2 as discussed in Chapter 2. In a general k-ary n-cube network, group VC_1 contains 2 virtual channels which are crossed deterministically (e.g. in an increasing order of dimensions) and group VC_2 contains $(V-2)$ virtual channels which are crossed adaptively. For the special case, the hypercube (k=2), group VC_1 contains 1 virtual channel and group VC_2 contains V-1 virtual channels.

Note that switch internal delay was not considered when developing the models in Chapters 3, 4 and 5 as the accuracy of the model does not depend on this parameter and is mainly dependent on how well the model can predict the blocking delay in the network. However, as we will see, the pipelined wire delay model depends on the switch internal

delay. To take this into account, we have to rewrite some of the Equations in the proposed models. For example, Equation 3.5 should change to $S_H = (|H|+M)*(t_c+t_s)+\sum_{j=1}^{|H|}B_j$, and the variance of the service time distribution, used to compute Equations 3.20 and 3.21, can be approximated as $\sigma_S^2 = (\overline{S} - M*(t_s+t_c))^2$.

6.2 The proposed cost-performance model

Most practical and experimental machines employ either the 2D torus or 3D torus as the two most famous instances of lower-dimensional *k*-ary *n*-cubes, and the hypercube as the best-known example of higher dimensional networks. In this section, we compare the performance merits of these networks for different implementation constraints and working conditions. To do so, we use the analytical models already introduced in Chapters 3, 4, and 5. We would rather use the unidirectional *k*-ary *n*-cube model since a hypercube is topologically a unidirectional 2-ary *n*-cube whereas the bidirectional 2-ary *n*-cube is a hypercube with redundant inter-node links. However, before discussing the relative performance merits of the torus and the hypercube, this section examines the constraints imposed by implementation technology on channel bandwidth and wiring delays.

6.2.1 Implementation constraints

Due to the limited channel bandwidth imposed by implementation technology, a flit is broken into channel words (or *phits* [55]), each of which is transferred in one cycle. If the *channel width* (i.e. number of wires) is C_w bits, a message of B bits is divided into $M=B/C_w$ phits [6]. In practice, a flit in wormhole routing may be composed of one or more phits.

Dally [44] has used the *bisection width*, i.e. the number of wires that cross the middle of the network, as a rough measure of the network wiring density in a pure VLSI implementation. Let us define $k_{2D-torus}$, $k_{3D-torus}$ and n to be, respectively, the radix of the 2D torus, the radix of the 3D torus and the dimension of the hypercube network, chosen such that the network size in the three topologies is equal to $N = (k_{2D-torus})^2 = (k_{3D-torus})^3 = 2^n$. Let us assume that a network is implemented on the two dimensional physical plane with $\sqrt{N}$ nodes along each dimension. The bisection width of the 2D torus, the 3D torus and the hypercube, $B_{2D-torus}$, $B_{3D-torus}$ and $B_{hypercube}$, with a channel width, $C_{w_{2D-torus}}$, $C_{w_{3D-torus}}$ and $C_{w_{hypercube}}$, can be expressed as [6, 44]

$$B_{2D-torus} = 2\sqrt{N} \times C_{w_{2D-torus}}, \tag{6.1}$$

$$B_{3D-torus} = 2\sqrt[3]{N^2} \times C_{w_{3D-torus}}, \tag{6.2}$$

$$B_{hypercube} = 2\frac{N}{2} \times C_{w_{hypercube}} = N \times C_{w_{hypercube}}. \tag{6.3}$$

If the bisection width is held fixed, the relationship between channel widths in the 2D torus, the 3D torus and the hypercube is given by

$$C_{w_{2D-torus}} = \sqrt[6]{N} \times C_{w_{3D-torus}} = \frac{\sqrt{N}}{2} C_{w_{hypercube}}. \tag{6.4}$$

Similarly, in multiple-chip implementations, where a complete node is fabricated on a chip, *pin*-out, which is the number of I/O pins (i.e. node degree × channel width), is a more suitable metric [1, 3]. The node pin-out for the 2D torus, 3D torus and hypercube, $P_{2D-torus}$, $P_{3D-torus}$ and $P_{hypercube}$, can be written as

$$P_{2D-torus} = 4C_{w_{2D-torus}}, \tag{6.5}$$

$$P_{3D-torus} = 6C_{w_{3D-torus}}, \tag{6.6}$$

$$P_{hypercube} = 2nC_{w_{hypercube}}. \tag{6.7}$$

Assuming a constraint of constant node pin-out, the channel width relationship in the considered networks will be

$$C_{w_{2D-torus}} = \frac{3}{2}C_{w_{3D-torus}} = \frac{n}{2}C_{w_{hypercube}}. \tag{6.8}$$

Equations 6.4 and 6.8 reveal that the torus has wider channels than the hypercube under both the constant bisection width and node pin-out constraints. For typical network sizes, under the constant wiring density constraint the 2D torus has even wider channels than under constant pin-out constraint, relative to the 3D torus and hypercube.

6.2.2 Wire delay model

We take the 2D torus as our base network for the comparison and calculate the desired parameters in the 3D torus and the hypercube in terms of those in the 2D torus base network. When mapped into the 2D plane, the 3D torus and the hypercube end up with longer wires, and therefore with higher wire delays than their 2D torus equivalent because of their larger number of dimensions, which have to be folded into the 2D plane. Note that this has to be taken into account even when a constant pin-out constraint is in effect because any network system has to be implemented ultimately either in a 2D (e.g PCBs) or 3D (cabinets, etc) physical medium. We focus here on a 2D plane (i.e. normalising the implementation parameters for a 2D torus) rather than a 3D space although we could equally use a 3D space implementation constraint [157], simply changing Equations 6.9 and 6.10 below. However, as long as a relative performance assessment is the goal, a 2D plane or 3D space implementation constraint result in similar conclusions.

The wire delay, due to long wires, can be reduced by using wire transmission line characteristics, as suggested by Scott and Goodman [157]; the wire has a storage capacity and can simply be treated as sequence of stages in the pipeline transmission of phits, with no need to wait for a phit to arrive before transmitting the next one. Such pipelined wire delays can be easily modelled by scaling the channel cycle time by factor $\bar{\gamma}_{pipelined}$, given as [157]

$$\bar{\gamma}_{pipelined} \approx \frac{2(\sqrt{N}-1)}{nkR}, \tag{6.9}$$

where R is the ratio of the switch cycle time (t_s) to the channel cycle time, t_c, in the 2D torus. A detailed derivation of $\bar{\gamma}_{pipelined}$ can be found in [157]. When R=1, the wire delay is equivalent to the switch delay in a 2D torus, but becomes higher in the 3D torus and the hypercube, reflecting their longer wires. When normal channels are considered (non-pipeline wire delay model) the channel cycle time is scaled by [157]

$$\bar{\gamma}_{non-pipelined} = \begin{cases} \dfrac{\sqrt{N}}{kR}, & \text{if } k > 2 \\ \dfrac{\sqrt{N}}{4R}, & \text{if } k = 2 \end{cases}. \tag{6.10}$$

6.2.3 Cost of routers

The constant bisection bandwidth and pin-out constraints have already been used [4, 6, 44] to fix the network cost (without taking into account the router's internal hardware cost) when VLSI implementation and multiple-chip implementation are considered. However these constraints do not consider the cost of the hardware used inside a router, and to make a fair comparison we must also take this into account (due to the crossbar switch, address decoder unit, virtual channel buffers and associated logic) fixing it for the hypercube, the

2D torus and the 3D torus. This is also useful when calculating the channel cycle time in the 3D torus and hypercube with respect to the channel cycle time in the 2D torus. To do so, each virtual channel in the 2D torus, the 3D torus and the hypercube is associated with a flit-size buffer (we assume that the flit size is the channel width of the 2D torus). This means that the phit size is equal to the flit size in the 2D torus while in the hypercube and the 3D torus it is different as the channel width in the hypercube and the 3D torus is smaller than that of the 2D torus. With $C_{w_{2D-torus}}$, $C_{w_{3D-torus}}$ and $C_{w_{hypercube}}$ being the phit size (also channel width) in the 2D torus, the 3D torus and the hypercube, respectively, and $t_{c_{2D-torus}}$ being the channel cycle time in the 2D torus, the time required to send a flit across a physical channel (or flit transmission delay), in the 3D torus and the hypercube, can be given by

$$t_{c_{3D-torus}} = \mu_{3D-torus} t_{c_{2D-torus}}, \tag{6.11}$$

$$t_{c_{hypercube}} = \mu_{hypercube} t_{c_{2D-torus}}, \tag{6.12}$$

where $\mu_{3D-torus}$ and $\mu_{hypercube}$ are scaling factors of flit transmission delay in the 3D torus and the hypercube (compared to that in the 2D torus), given by

$$\mu_{3D-torus} = \frac{C_{w_{2D-torus}}}{C_{w_{3D-orus}}} \bar{\gamma}_{3D-torus}, \tag{6.13}$$

$$\mu_{hypercube} = \frac{C_{w_{2D-torus}}}{C_{w_{hypercube}}} \bar{\gamma}_{hypercube}. \tag{6.14}$$

This takes into account the effect of wire delay when mapping the 3D torus and the hypercube into a 2D plane (given by Equations 6.9 and 6.10) and the effect of narrower channel width in the 3D torus and hypercube.

Using pipelined channels, taking the 2D torus as the base network and assuming its channel cycle period is equal to the unit of time (one clock cycle), we can derive the flit

transmission delay factors for the 3D torus and the hypercube (using Equations 6.4 and 6.8), when the constant bisection width constraint is considered, as

$$\mu_{3D-torus} = \sqrt[6]{N} \times \frac{2(\sqrt{N}-1)}{3R\sqrt[3]{N}} = \frac{2(\sqrt{N}-1)}{3R\sqrt[6]{N}}, \tag{6.15}$$

$$\mu_{hypercube} = \frac{\sqrt{N}}{2} \times \frac{2(\sqrt{N}-1)}{2nR} = \frac{N-\sqrt{N}}{2nR}. \tag{6.16}$$

When the pin-out constraint is applied these equations are found to be

$$\mu_{3D-torus} = \frac{3}{2} \times \frac{2(\sqrt{N}-1)}{3R\sqrt[3]{N}} = \frac{\sqrt{N}-1}{R\sqrt[3]{N}}, \tag{6.17}$$

$$\mu_{hypercube} = \frac{n}{2} \times \frac{2(\sqrt{N}-1)}{2nR} = \frac{\sqrt{N}-1}{2R}. \tag{6.18}$$

Similarly, for normal (non-pipelined) channels when the constant bisection width constraint is considered we can write

$$\mu_{3D-torus} = \sqrt[6]{N} \times \frac{\sqrt{N}}{kR} = \frac{\sqrt[3]{N}}{R}, \tag{6.19}$$

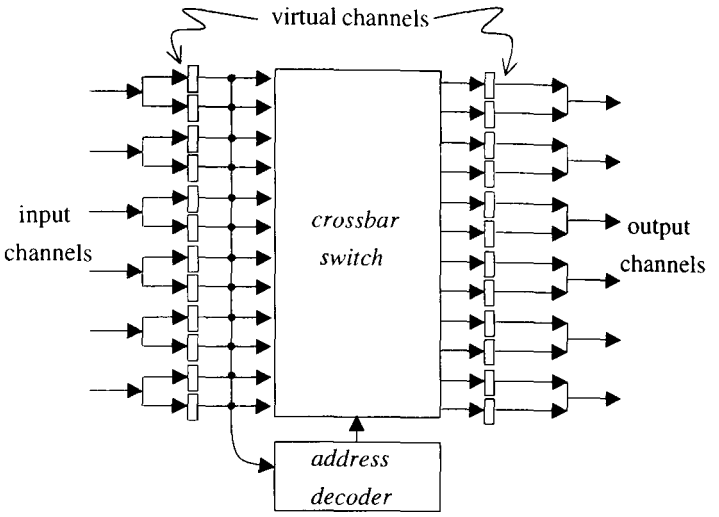
$$\mu_{hypercube} = \frac{\sqrt{N}}{2} \times \frac{\sqrt{N}}{4R} = \frac{N}{8R}, \tag{6.20}$$

and when the pin-out constraint is applied we have

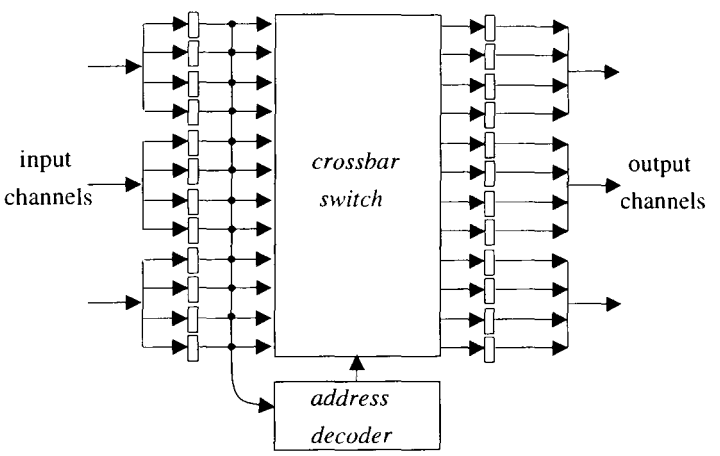
$$\mu_{3D-torus} = \frac{3}{2} \times \frac{\sqrt{N}}{kR} = \frac{3\sqrt[6]{N}}{2R}, \tag{6.21}$$

$$\mu_{hypercube} = \frac{n}{2} \times \frac{\sqrt{N}}{4R} = \frac{n\sqrt{N}}{8R}. \tag{6.22}$$

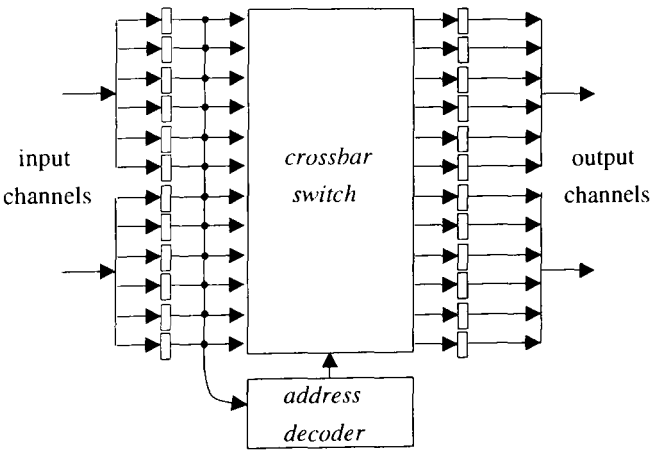
The above equations help us to calculate the normalised channel cycle time in the 3D torus and the hypercube with respect to that of the base network (the 2D torus) but it makes the



(a)



(b)



(c)

Figure 6.1: The result of normalizing the cost of routers internal hardware for the 64-node network with three different topologies, (a) 6-dimensional hypercube, (b) 4x4x4 3D torus, and (c) 8x8 2D torus.

total buffer used in the 3D torus and the hypercube larger than that in the base network. For instance, with a 64-node network size, the 2D base network (8x8 torus) has 2 flit buffers for the two input channels, the 3D network (4x4x4 torus) has 3 flit buffers for the three input channels and the hypecube (6-dimensional) has 6 flit buffers for the 6 input channels at each node.

To make the total buffer size equal for each router, for the three different networks, we increase the number of virtual channels (associated to each physical channel) in the 2D and 3D torus to make the total node buffer size equal to that in the equivalent hypercube. The total buffer used in the router is simply (*number of physical channel*)×(*number of virtual channels per physical channel*). Therefore, the total buffer space used inside the router in the 2D torus, the 3D torus and the hypercube is respectively given by

$$Buffer_{hypercube} = nV_{hypercube}, \tag{6.23}$$

$$Buffer_{2D-torus} = 2V_{2D-torus}, \tag{6.24}$$

$$Buffer_{3D-torus} = 3V_{3D-torus}, \tag{6.25}$$

where $V_{hypercube}$, $V_{2D-torus}$, and $V_{3D-torus}$ denote respectively the number of virtual channels per physical channel in the hypercube, 2D torus and 3D torus. Recalling that the minimum number of virtual channels (per physical channel) in the hypercube $V_{hypercube}$, according to the Duato's theory, is 2 and using Equations 6.23, 6.24 and 6.25, we have

$$V_{hypercube} = 2, \tag{6.26}$$

$$V_{2D-torus} = n, \tag{6.27}$$

$$V_{3D-torus} = \frac{2n}{3}, \tag{6.28}$$

assuming equal buffer requirements at different node types (for the three different topologies). Note that $V_{2D-torus}$ and $V_{3D-torus}$ must be at least 3 to be able to exploit Duato's adaptive routing. For the above 64-node example network, we have $V_{hypercube} = 2$, $V_{2D-torus} = 6$, and $V_{3D-torus} = 4$ each imposing a total of 12 flit buffers (corresponding to 12 virtual channels) at each node as illustrated in Figure 6.1.

Without the above normalisation, the crossbar switch and address decoder delay may be different in each network for the following reasons. The router's internal switch (bridging the input channels to the output channels) in the hypercube would be larger than that in the torus due to its larger number of input and output channels. As a consequence, the switching delay in the hypercube would be higher due to the additional complexity, according to Chien's model [35]. However, comparable switch and address decoder delays in the networks can be obtained if the routers have comparable switch sizes and equal number of virtual channels. Using such a intra-node cost model for normalising the total number of virtual channels at different node types, we firstly make the cost of hardware used inside a router (both with buffer and crossbar switch size) in the 2D torus, the 3D torus and the hypercube network equal; secondly, we make the crossbar switch and address decoder delays in these networks comparable since these delays are some functions of the number of virtual channels in the router [35].

6.3 Comparison results and discussion

In this section, the performance of the three networks in question is examined for both the constant bisection width and the constant pin-out constraints. For illustration, the following various network sizes are examined:

- A small size of N=64 nodes; configured as an 8x8 torus, a 4x4x4 torus, and a 6-dimensional hypercube.

Table 6.1: Calculated flit transmission delay factors* (μ) and number of virtual channels per physical channel (V) in the three topologies (2D torus, 3D torus, hypercube) for different network sizes and implementation constraints with R=1 and for both pipelined and non-pipelined channels.

Network size / Topology	$N = 64$					$N = 512$					$N = 4096$				
	V	μ				V	μ				V	μ			
		Bisection width		Pin - out			Bisection width		Pin - out			Bisection width		Pin - out	
		pipelined	non - pipelined	pipelined	non - pipelined		pipelined	non - pipelined	pipelined	non - pipelined		pipelined	non - pipelined	pipelined	non - pipelined
2D Torus	6	1	1	1	1	9	1	1	1	1	12	1	1	1	1
3D Torus	4	3	4	2	3	6	6	8	3	5	8	11	16	4	6
Hypercube	2	5	8	4	6	2	28	64	11	26	2	168	512	32	96

*All calculated values are rounded up to the nearest integer number.

- A medium size of N=512 nodes; configured as a 23x23 torus[1], an 8x8x8 torus, and a 9-dimensional hypercube.
- A moderately large size of N=4096 nodes; configured as a 64x64 torus, a 16x16x16 torus and a 12-dimensional hypercube.

The flit transmission dealys in the 3D torus and the hypercube are normalised to that of the 2D torus using Equations 6.11 and 6.12 under the constant wiring density and pin-out constraints. Let us set R=1, implying that the switching time (t_S) is equal to the channel cycle time (t_C) in the 2D torus. Assuming that a physical channel in the hypercube has V=2 virtual channels, one deterministic and one adaptive, and using Equations 6.26 and 6.27 the number of virtual channels per physical channel in the 2D torus and the 3D torus are calculated in order to have an equal router cost for the three considered networks. Table 1 illustrates the flit transmission delay factor (μ) and the number of virtual channels per physical channel (V) calculated for three network sizes (N=64, 512 and 4096) under both

[1] Approximate root is used for N=512.

the constant bisection width and pin-out constraints using both pipelined and non-pipelined wire delay models, for the three network topologies (2D torus, 3D torus and hypercube). In what follows, all message lengths are quoted in terms of flits in the 2D torus.

6.3.1 The results for uniform traffic load

Figure 6.2 (a) depicts latency results in the 2D torus, 3D torus and hypercube under the constant bisection width constraint, using pipelined wire delay model, and for message length M=64 flits in the 64, 512, and 4096- node systems. The figure reveals that the 2D torus is able to exploit its wider channels to provide a lower latency than the 3D torus and hypercube under light to moderate traffic.

For small network sizes, however, as traffic increases its performance degrades as message blocking rises, soon offsetting any advantage of having wider channels compared to the hypercube. This is mainly due to small diameter of the hypercube and its rich connectivity providing more alternative routes and thus more adaptivity. Moreover for small networks the relative flit transmission delay in the hypercube is not large compared to the 2D torus base network.

With moderate and large networks, the relatively slower and thinner channels in the hypercube and 3D torus plus the effect of blocking prevent the hypercube from exploiting its main topological advantages (lower diameter and more alternate routes, giving more adaptivity), resulting in performance degradation compared to the 2D torus. This is most noticeable in large networks (e.g. 4096 node networks in the figure). In all cases, when the traffic is low, since there are no message blocking effects, the lower flit transmission delay in the 2D torus (compared to the 3D torus and the hypercube), results in a better performance in the 2D torus. The same conclusion was obtained by Dally [44] and

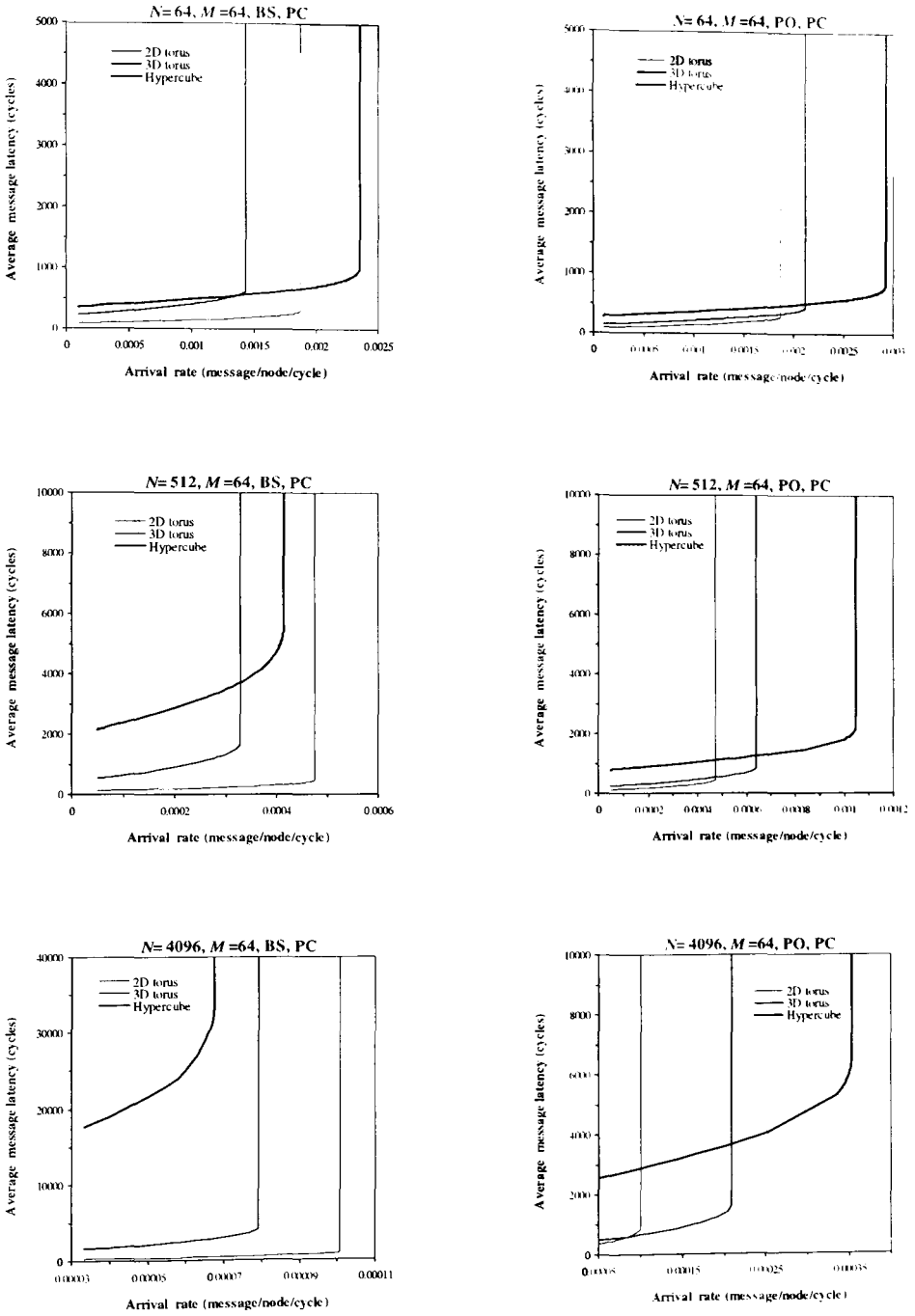
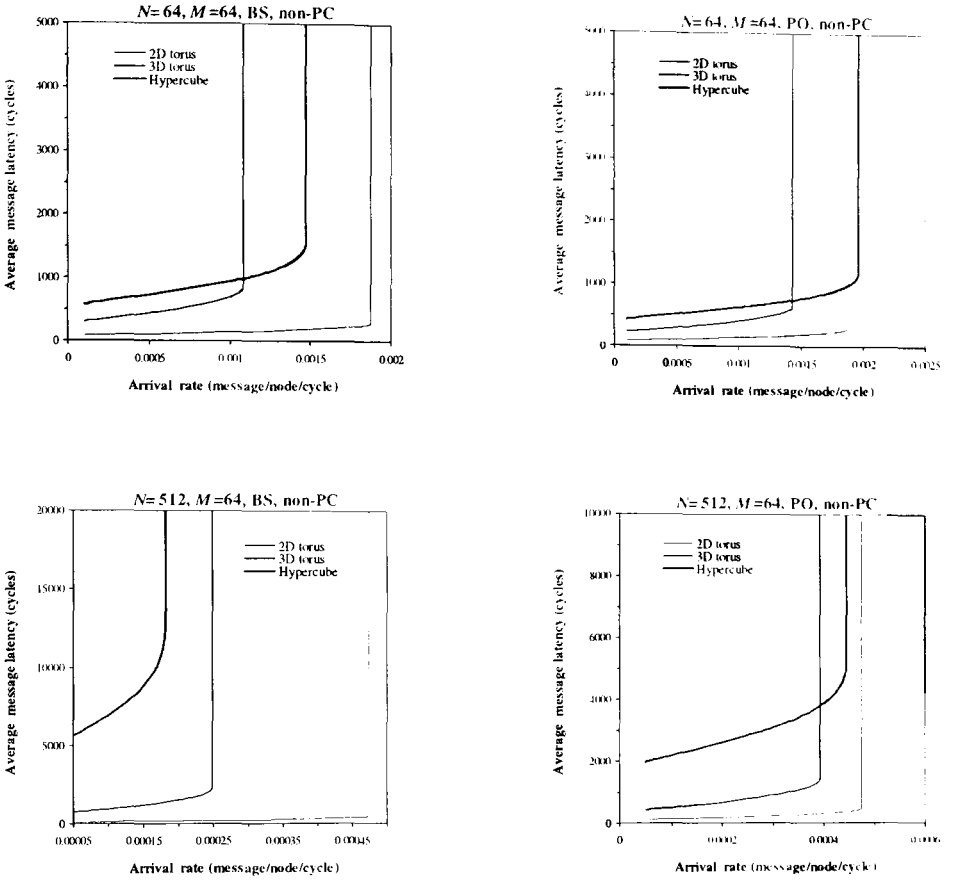
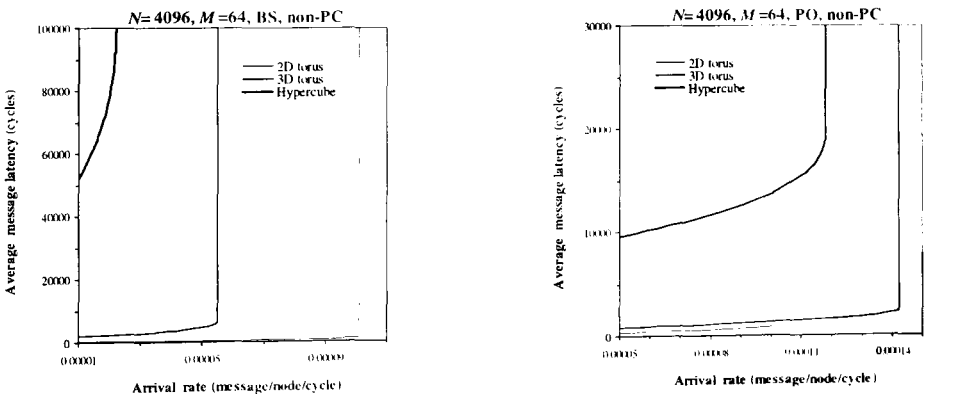


Figure 6.2: The average message latency for different network sizes (N=64, 512, 4096) and topologies (the 2D torus, 3D torus and hypercube), for message length M=64 flits, and pipelined wire delay model when (a) constant bisection width constraint, and (b) constant pin-out constraint, are applied.





(a) **(b)**

Figure 6.3: The average message latency for different network sizes (N=64, 512, 4096) and topologies (the 2D torus, 3D torus and hypercube), for message length M=64 flits, and non-pipelined wire delay model when (a) constant bisection width constraint, and (b) constant pin-out constraint, are applied.

Agarwal [6] for deterministic routing and uniform traffic, i.e. lower dimensional k-ary n-cubes outperform their higher dimensional counterparts under a bisection bandwidth constraint.

Let us now consider the effect of constant pin-out constraint with pipelined wire delay model. As can be seen in Figure 6.2(b), the pin-out constraint favours the higher dimensional k-ary n-cubes (compared to the bisection width constraint) thus advantaging the hypercube. This is the case for all network sizes. Abraham [4] and Agarwal [6] have also compared the performance merits of the torus and the hypercube under a constant pin-out constraint with deterministic routing and uniform traffic. They concluded that higher dimensional k-ary n-cubes have superior performance over their lower dimensional counterparts under this constraint.

Figure 6.3 shows the same quantities as Figure 6.2 but for a non-pipelined wire delay model. When the constant bisection bandwidth constraint is considered, the conclusion arrived at is almost the same as for the pipelined wire delay model (shown in Figure 6.3(a)), i.e. lower dimensional k-ary n-cubes exhibit superior performance compared to their higher dimensional counterparts, especially for moderately large networks. However, when a constant pin-out constraint is considered, the results are very interesting. For small moderate, and large networks the topology exhibiting the best relative performance is respectively the hypercube, 2D torus and 3D torus. This differs from Abraham's [4] and Agarwal's [6] conclusions because we have considered the effects of virtual channels and the cost of hardware used inside the routers.

6.3.2 The results for hotspot traffic

Figure 6.4 depicts latency results in the 2D torus, 3D torus and hypercube under the

constant bisection width constraint, using a pipelined wire delay model, for message length M=64 flits and hotspot traffic fraction h=0.05, 0.2 and 0.5 in the 64, 512, and 4096-node systems. The figure reveals that the 2D torus is able to exploit its wider channels to provide a lower latency than the 3D torus and hypercube under light to moderate traffic when the hotspot portion is small. However, as traffic increases its performance degrades as message blocking rises, soon offsetting any advantage of having wider channels, even when h is small. This is mainly due to the fact that the small diameter of the hypercube and its rich connectivity provide more alternative routes and thus more adaptivity. Moreover for small networks the relative flit transmission delay in the hypercube is not large (compared to the 2D torus base network). When hotspot traffic fraction increases, the 2D torus dominates the hypercube because, in such a scenario, the main component contributing to the mean message latency is due to hotspot messages (see Chapter 4, Section 4.2). These messages are often blocked in the network by other hotspot messages that have already acquired the channels leading to the hotspot node. The effects of blocking plus the relatively higher flit transmission delay of the hypercube, in this case, do not enable the hypercube to exploit its main topological advantages (lower diameter and more alternate routes giving more adaptivity), resulting in a performance degradation compared to the 2D torus.

With moderate and relatively large networks even small hotspot fractions cause the same conditions that occur in small networks with larger h. In addition, relativey slower and thinner channels in the hypercube and 3D torus favour the 2D torus further. This is most noticeable in large networks (e.g. 4096 node networks in the figure). In all cases, when the traffic is low, since there is no message blocking effect, the lower flit transmission delay in the 2D torus (compared to the 3D torus and the hypercube), results in a better performance in the 2D torus. The same conclusion was derived by Dally [44] and Agarwal [6] for deterministic routing and uniform traffic.

Figure 6.5 shows latency results in the 2D torus, 3D torus and hypercube under the constant pin-out constraint, using pipelined wire delay model, for message length M=64 flits in 64, 512, and 4096- node systems. As can be seen in the figure, the pin-out constraint favours the higher dimensional k-ary n-cubes (compared to the bisection width constraint) thus advantaging the hypercube slightly. However, similar to the results obtained for the bisection bandwidth constraint, the same conclusion can be made for small networks with a large hotspot fraction and for medium and large networks with a relatively small hotspot fraction. Note that the saturation traffic rates in the three networks are closer compared to those shown in Figure 6.4 under the constant bisection width constraint. Abraham [4] and Agarwal [6] have compared the performance merits of the torus and the hypercube under the constant pin-out constraint with deterministic routing, uniform traffic and a non-pipelined wire delay model, showing that the hypercube has superior performance over the torus. Their conclusion is different from ours because we have considered the virtual channels effects and also the cost of hardware used inside the routers in the presence of hotspot traffic.

Figures 6.6 and 6.7 show the same results shown in Figures 6.4 and 6.5 for a non-pipelined wire delay model. Since the non-pipelined wire delay model favours the 2D torus more than when the pipelined wire delay model is considered, a better performance is achieved for the 2D torus. The conclusion in the case of the non-pipelined wire delay model is similar.

Since it is clear that the 2D torus gives better performance than the other two for medium and large network sizes, let us now focus on the smaller network (64-nodes). Figure 6.8 illustrates the mean message latency against message length in the 64-node 2D torus, 3D torus and hypercube networks for both the constant bisection width and pin-out constraints with a pipelined wire delay model. The traffic generation rate at each node, λ_g, is fixed at 0.0001 and the hotspot fraction is assumed to be $h = 0.2$. As can be seen in the figure,

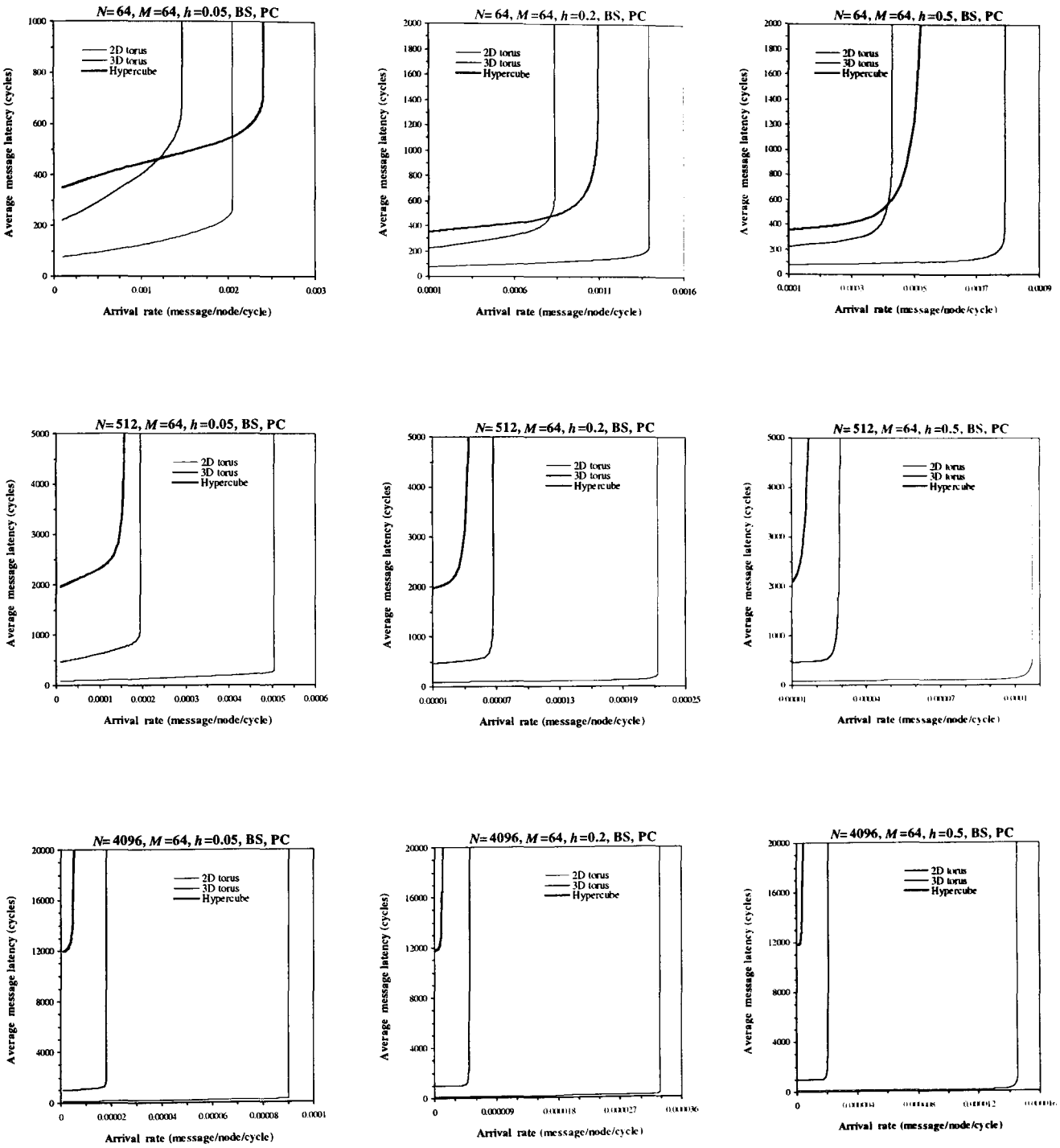


Figure 6.4: The average message latency for different network sizes (N=64, 512, 4096) and topologies (the 2D torus, the 3D torus and the hypercube), with hotspot traffic portions h=0.05, 0.2, and 0.5, when the constant bisection width constraint and pipelined wire delay model are applied. The message length is M=64 flits.

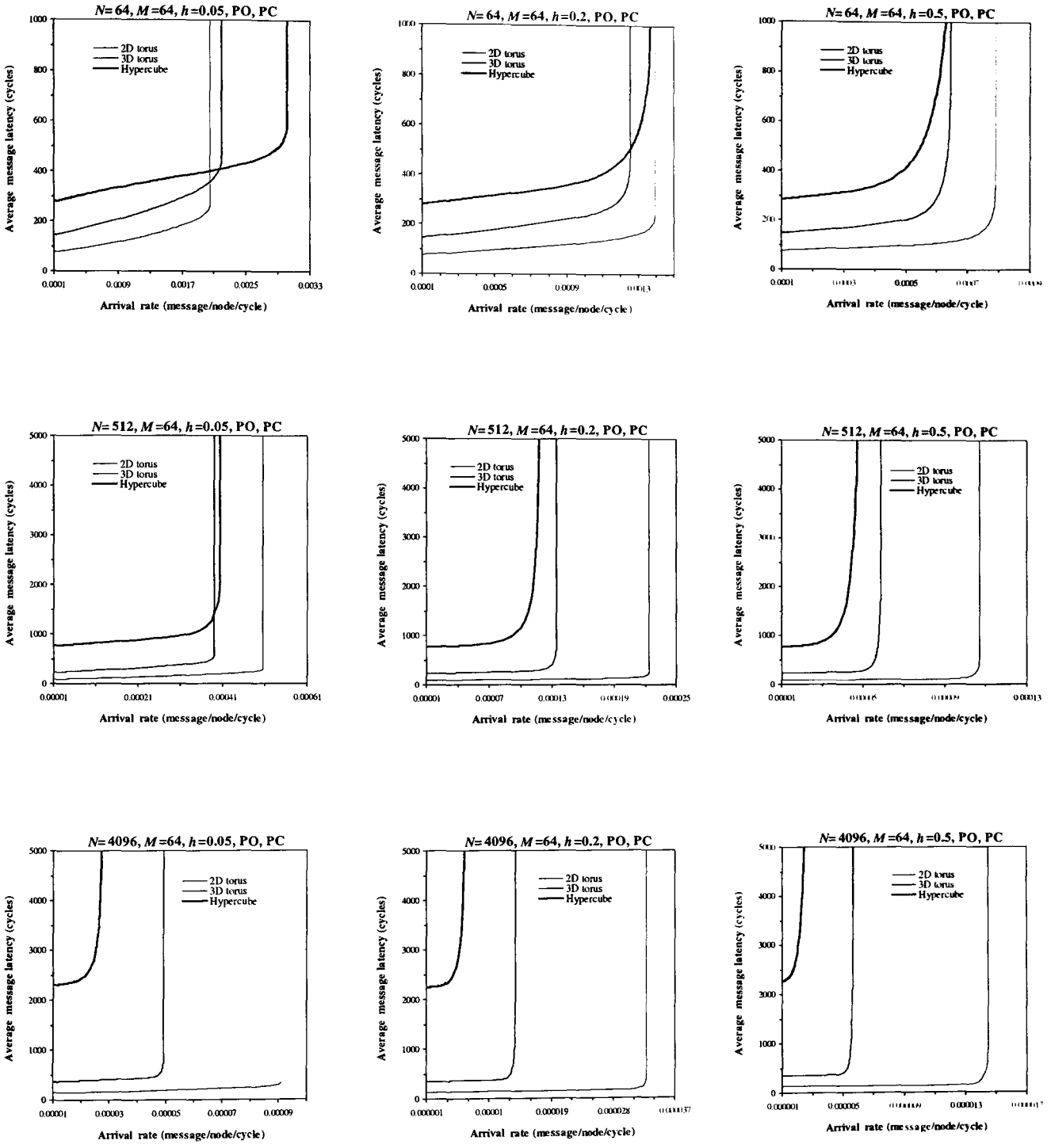


Figure 6.5: The average message latency for different network sizes (N=64, 512, 4096) and topologies (the 2D torus, the 3D torus and the hypercube), with hotspot traffic portions h=0.05, 0.2, and 0.5, when the constant pin-out constraint and pipelined wire delay model are applied. The message length is M=64 flits.

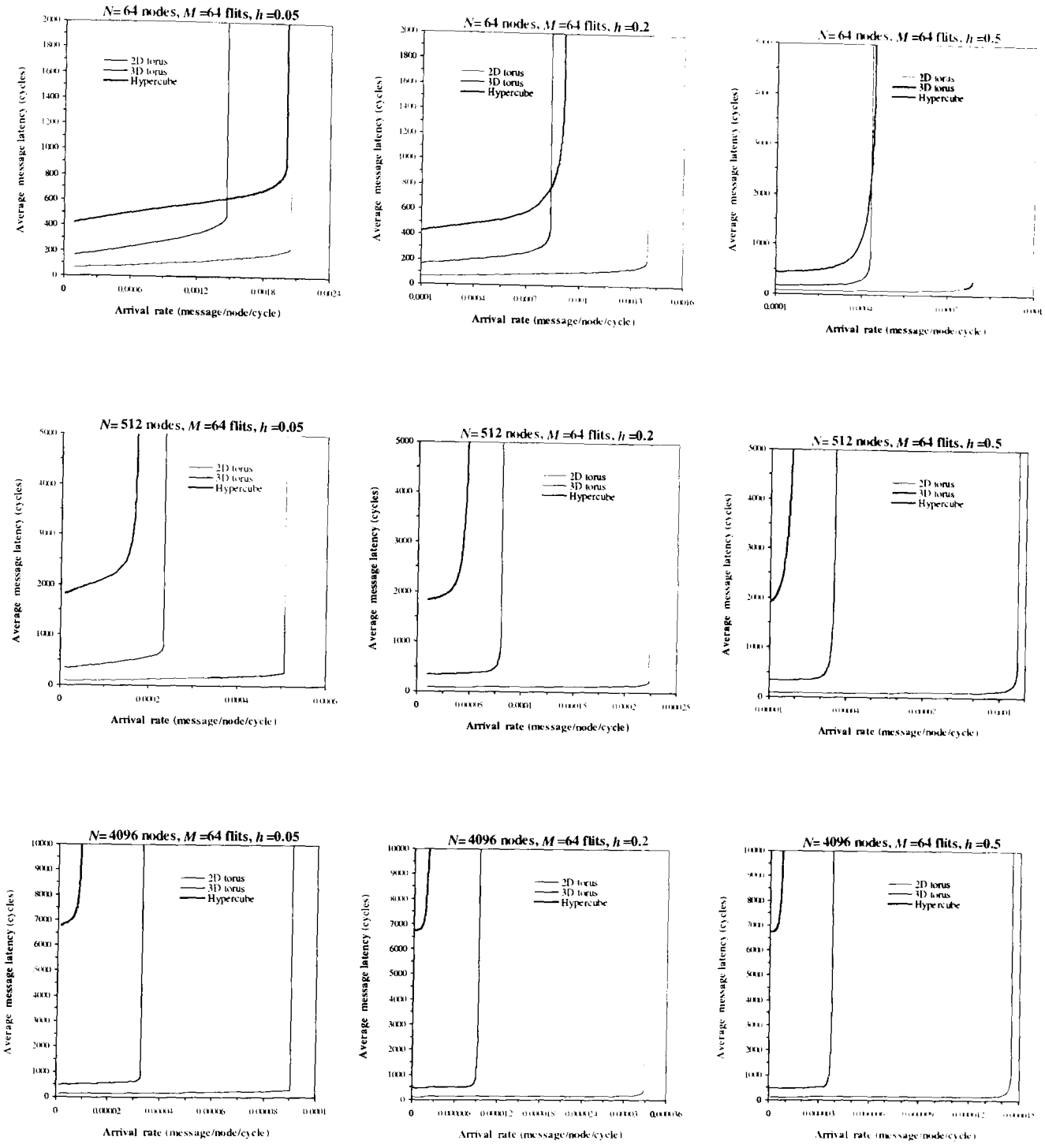


Figure 6.6: The average message latency for different network sizes (N=64, 512, 4096) and topologies (the 2D torus, the 3D torus and the hypercube), with hotspot traffic portions h=0.05, 0.2, and 0.5, when the constant bisection width constraint and non-pipelined wire delay model are applied. The message length is M=64 flits.

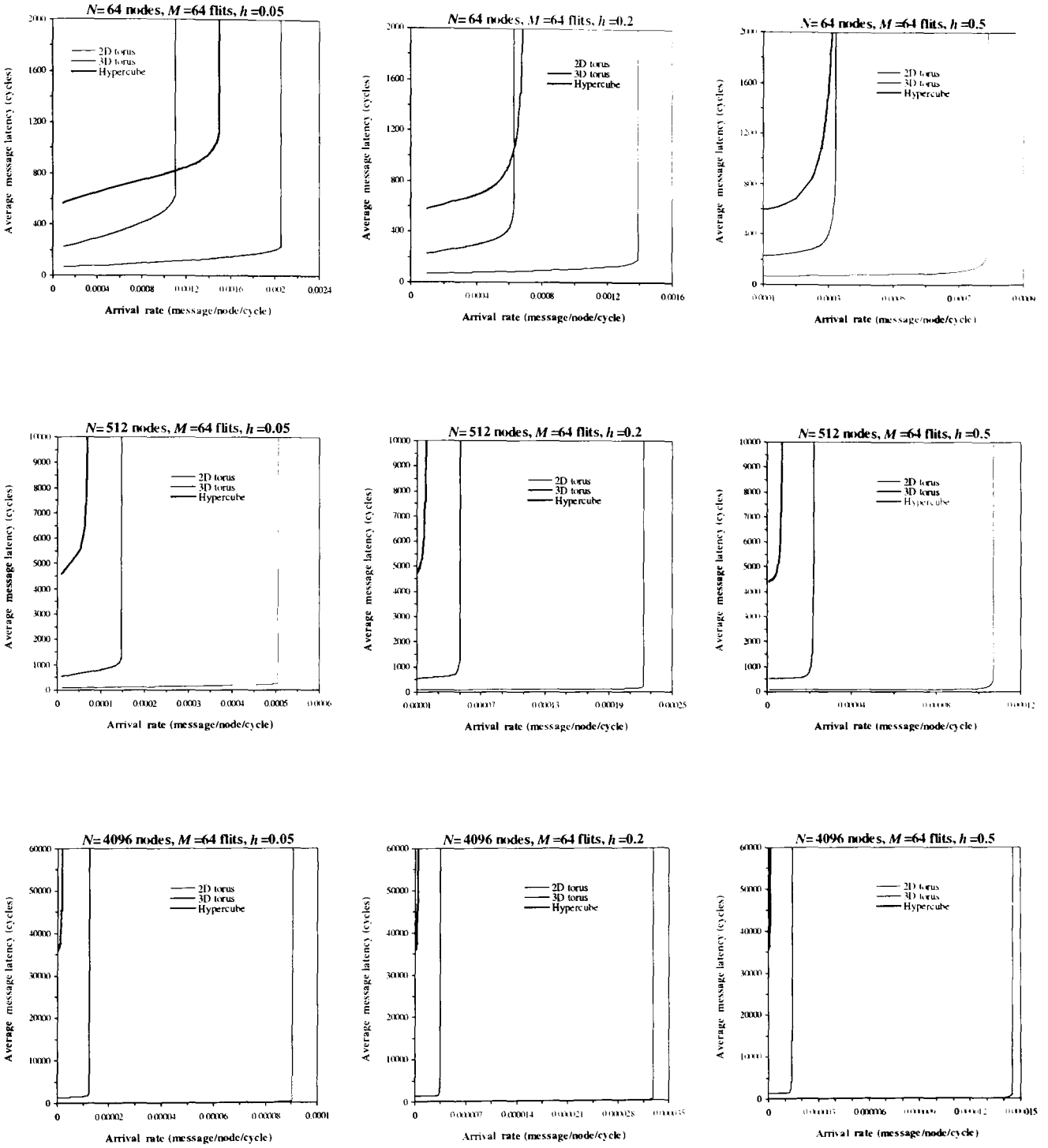


Figure 6.7: The average message latency for different network sizes (N=64, 512, 4096) and topologies (the 2D torus, the 3D torus and the hypercube), with hotspot traffic portions h=0.05, 0.2, and 0.5, when the constant pin-out constraint and non-pipelined wire delay model are applied. The message length is M=64 flits.

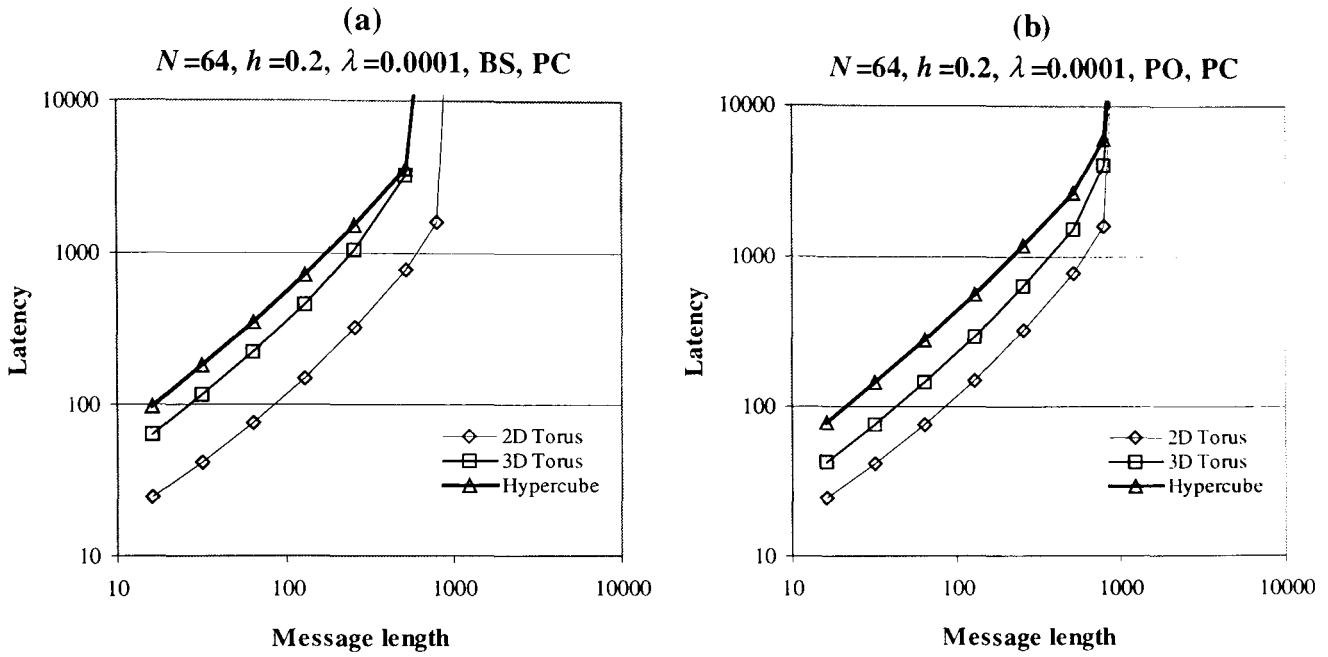


Figure 6.8: The effect of message length M on the average message latency in a 64-node 2D-torus, 3D-torus, and hypercube under constant (a) bisection width and (b) pin-out constraints, with hotspot traffic portion h=0.2, and message generation rate λ_g =0.0001,when pipelined wire delay model is applied.

the 2D torus shows a lower latency for short messages. Note that under constant pin-out constraint the latency in the three networks get closer for long messages.

Figure 6.9 shows the effects of the hotspot traffic fraction on the saturation traffic rate of the 64-node 2D torus, 3D torus and hypercube, when M=64, for both the constant bisection width and pin-out constraints, using pipelined wire delay model. The results reveal that the 2D torus behaves better than the others when h increases. Under the pin-out constraint the saturation rates in the three networks are closer although the hypercube saturates later than the torus when h=0.1 and earlier when h=1. Under the constant bisection width constraint, the relative performance merits of the three networks does not

change as h increases, while the ranking is changed between the 3D torus and hypercube from h=0.1 to h=1 under the pin-out constraint.

6.3.3 The results for matrix-transpose and digit-reversal permutation traffic patterns

We have examined different scenarios for different network sizes and topologies with different matrix-transpose and digit-reversal traffic portions and different number of virtual channels and observed almost the same trends in the latency curves as for uniform traffic. This is not so surprising as we have already seen in Chapter 5 that the effect of matrix-transpose and digit-reversal traffic portions on the total mean message latency is small. Therefore, for the sake of brevity, we do not report the analyses for matrix-transpose and digit-reversal traffic patterns since the conclusion of these is similar to that for the uniform traffic pattern.

6.4 Conclusions

Many studies have stressed the performance benefits of adaptive over deterministic routing in the presence non-uniform traffic patterns [55, 148] such as hotspots [142]. This chapter examined the relative performance merits of adaptively routed multi-dimensional k-ary n-cubes under uniform, hotspot, matrix-transpose and digit-reversal traffic patterns.

Our analysis has considered virtual channels and taken into account the cost of both network links and the internal hardware of routers. We believe previous analyses reported in the literature [4, 6, 44] could not be entirely fair since they considered only the cost of network links and ignored the cost of router hardware.

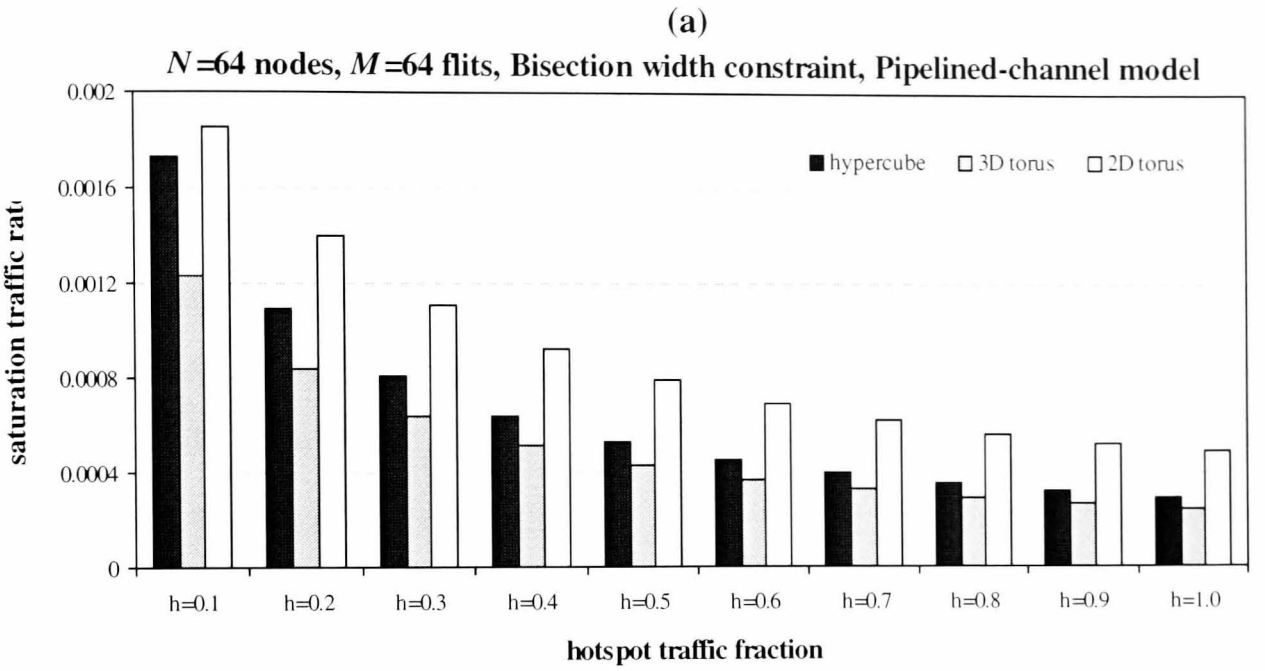


Figure 6.9: The effect of hotspot traffic portion *h* on the saturation traffic rate in a 64-node 2D-torus, 3D-torus, and hypercube under constant (a) bisection width and (b) pin-out constraints, with message length *M*=64 flits, when pipelined wire delay model is applied.

We conducted our comparisons for three network sizes: small, medium and large networks with respectively 64, 512 and 4096 nodes. With uniform traffic and a pipelined wire delay model, both constant bisection bandwidth and pin-out constraints give results in agreement with those achieved by Abraham [4] and Agarwal [6]. However, when a non-pipelined wire delay model is considered, the constant pin-out constraint results in a different conclusion from that reached by Abraham and Agarwal. While they concluded that higher dimensional *k*-ary *n*-cubes should perform better than their lower dimensional counterparts with a constant pin-out constraint, we have found that for relatively large networks (thousands of nodes), it is, surprisingly, the 3D torus that is the network with the best performance while for moderate size networks (hundreds of nodes) the 2D torus gives the best results. Under the stated conditions, it is only in small networks (less than hundred) that the hypercube dominates.

The results, in the presence of hotspot traffic, indicate that, under a constant bisection bandwidth constraint, the torus has better performance than the hypercube, just as reported in previous work with deterministic routing, e.g. Dally's [44] and Agrawal's [6]. However, this present study has reached a different conclusion from previous ones (e.g. in [4, 6]) under the constant pin-out constraint. Our results have shown that for moderate and large network sizes the 2D torus always shows better performance than the hypercube under constant bisection bandwidth and pin-out constraints when hotspot traffic is present.

With matrix-transpose and digit-reversal traffic patterns the results obtained were almost the same as for a uniform traffic pattern.

Chapter 7

Conclusions and Future Directions

The interconnection network is a crucial component in any parallel computer since any interaction between the processing elements ultimately depends on its effectiveness [136]. Although many network architectures have been studied [110], and indeed deployed, none has proved clearly superior in all roles, since the communication requirements of different applications vary widely. Nevertheless, the k-ary n-cube has undoubtedly been the most popular interconnection network used in practice [7, 13, 92, 93, 98, 131, 147, 170] because, on balance, it has the most desirable properties [169, 173]. It has been studied extensively in many aspects and but still merits further exploration of its properties.

This thesis has undertaken just such an exploration of the characteristics and performance capabilities of k-ary n-cubes. The work has, in particular, focussed on combining adaptive routing and wormhole switching with virtual channels, a scenario of especial interest to current research, using mathematical models validated through simulation experiments. The performance of adaptive routing in wormhole-routed k-ary n-cubes was evaluated under different traffic conditions by developing analytical models for calculating average message latency.

Unlike other models, reported in the literature, those proposed in this study are more realistic and take account of more details of a real system. Previous attempts to consider equivalent levels of details have been based on measurement and simulation experiments. However, to simulate a massively parallel machine, say a system with several thousand nodes, these latter techniques can absorb enormous computational power in order to model realistically the interaction between network parameters and their effect on communications performance. Accurate and comprehensive analytical models are, in principle, powerful alternatives to such compute-intensive tasks and can indeed save considerable time and expense for network designers and researchers. This work has shown that such cost-effective models can be built with sufficient detail and accuracy to give a useful insight into performance merits of k-ary n-cubes. This should make it possible for prospective manufacturers to inform critical technical decisions prior to the actual construction of new machines that employ adaptive routing by reducing the potentially enormous design-space and allowing detailed effort to focus on the most promising scenarios.

The models developed were then used to compare performance merits of low-dimensional k-ary n-cubes to their high-dimensional counterparts under two major implementation constraints, constant bisection width and constant node pin-out. Several previous comparative analyses of networks have also used wiring density and pin-out count to quantify implementation cost in VLSI and multiple-chip technology. However, none has taken account of the cost of the routers, which may be complex and expensive to implement. We have extended cost constraints to include the internal architecture costs of the switch, and therefore have developed a significantly more realistic cost model. We have used this new and more realistic cost model in tandem with the more accurate and realistic analytical models proposed earlier to undertake a more reliable comparison of the systems in question. Incorporating these details has indeed resulted in notably different

conclusions from those reported by previous studies illustrating that such comparisons are very sensitive to the detail and realism of the assumptions made.

7.1 Summary of the results

This thesis has detailed several important achievements.

Firstly, expressions for calculating the number of nodes at and within a given distance from a chosen centre in a k-ary n-cube, were derived. Such expressions are useful in the study of spanning trees, widely used in collective communication algorithms and in the problem of resource placement in networks [20, 31, 70-72, 156].

Secondly an accurate analytical model to compute the mean message latency in k-ary n-cubes with fully adaptive wormhole routing was developed. This model achieves a high degree of accuracy under different operating conditions because it computes the exact expression for the probability of message blocking at any router. The model was extended to include bidirectional k-ary n-cubes and traffic patterns that exhibit communication locality. Using this model to draw a comparison between unidirectional and bidirectional k-ary n-cubes under both constant bisection bandwidth and pin-out constraints, has shown that bidirectional k-ary n-cubes outperform their unidirectional counterparts. The model also showed that the higher-dimensional k-ary n-cube networks (with large n), e.g. hypercubes, are more scalable than their low-dimensional counterparts (with large k), e.g. tori, because their total network bandwidth scales better with network size.

Thirdly, the first analytical model to compute the mean message latency in the presence of hotspot traffic in wormhole-routed k-ary n-cubes was then presented. Performance analysis has revealed that increasing the number of virtual channels can improve network performance when the hotspot traffic rate, h, is low. However, with the aid of the new

model, it was shown that when h is relatively large (defining a high percentage of hotspot traffic), adding virtual channels cannot improve performance noticeably. It was also shown that when h is small, the dominating factor causing network saturation is the uniform traffic component, while for large h, the dominating factor is the average latency for hotspot messages. Interestingly, comparing unidirectional and bidirectional k-ary n-cubes under both constant bisection bandwidth and pin-out constraints, has shown that bidirectional k-ary n-cubes perform better when the hotspot traffic rate is low, but that the opposite conclusion is reached when hotspot traffic is relatively large.

Fourthly, analytical models with fully adaptive routing were used to study the performance of cubes executing matrix-transpose and digit-reversal permutations. Simulation experiments are in good agreement with the latency results predicted by the analytical models. In fact these results show that matrix-transpose and digit-reversal traffic does not have a large impact on the overall network performance when fully adaptive routing algorithms are employed in unidirectional k-ary n-cubes. This is because adaptive routing in such networks is able to exploit all network channels and distribute the traffic load over network channels almost balanced.

Finally, the relative performance merits of k-ary n-cubes of differing dimensionality was assessed in the context of adaptive routing under uniform, hotspot, matrix-transpose and digit-reversal traffic patterns. A cost-model considering virtual channels and taking into account both network links and router hardware was adopted. Three network sizes have been considered: small, medium and large networks with respectively 64, 512, and 4096 nodes. The results obtained under uniform traffic and both constant bisection bandwidth and pin-out constraints are in agreement with those achieved by Abraham [4] and Agarwal [6] when a pipelined wire delay model is used. Since many current systems as well as those which will one day be constructed using system-on-chip technology use non-pipelined channels, however, we have also conducted a comparison for this latter case.

When a non-pipelined wire delay model is considered, assuming a constant pin-out constraint, our model leads to different conclusions from that of Abraham and Agarwal except in the case of small networks. We found that the 3D torus has the best performance in large networks, the 2D torus at moderate sizes and the hypercube proving superior only in small systems. When hotspot traffic is introduced, the 2D torus shows always better performance than the hypercube in moderate and large networks under both constant bisection bandwidth and pin-out constraints, contrary to the conclusions of previous studies [4, 6]. A similar result applies to matrix-transpose and digit-reversal traffic with uniform traffic.

7.2 Directions for the future work

There are number of issues and open problems that require further investigation. These can be grouped in two broad categories: (1) those which makes the proposed models more realistic, and (2) those which tackle other important issues in interconnection networks.

7.2.1 Developing more realistic models

There are a number of suggestions as to how the proposed models might be modified to capture other real-world situations. For example, in the proposed model for hotspot traffic, we considered only one hotspot. However, in real environments this may be overly restrictive and it would be useful to adapt the model to handle multiple hotspots. This does not require new tools and only requires a small amount of additional complexity.

The proposed models consider networks with virtual channels each with only a one-flit buffer, thus implementing a pure wormhole switching method. Some current implementations however use deeper buffers to effect partial or full virtual cut-through [97]. Again it would be useful to extend the models to include this scenario.

Many studies have revealed that the Poisson model cannot properly emulate the traffic characteristics in some actual applications such as those incorporating multimedia streams. Self-similar and pseudo self-similar traffic models have been suggested as a more faithful alternative [82, 146] but most studies considering such traffic models are based on simulation and measurement experiments. A more challenging extension of our work would be to analytically model k-ary n-cubes under self-similar traffic load.

7.2.2 Future research in interconnection networks

Moving beyond the core of the present work, there remain many interesting problems in the field which would benefit from the same analytical approach adopted in this thesis. A selection of such problems is listed below as an illustration of the potential of this line of attack.

In real systems increasing size will increase failure rates, and incorporation of fault-tolerant techniques will be of great importance [59]. Many studies [10, 11, 38, 101, 109, 116, 181, 182] have investigated fault-tolerant routing in interconnection networks and assessed performance via simulation [59]. There is currently no analytical model of fault-tolerant routing in the kind of networks under discussion above.

Many studies [28, 113, 114, 122, 125, 126, 135, 144, 145] have been conducted on designing, implementation, and simulation- and experimental-based performance evaluation of collective communication algorithms on different networks including k-ary n-cubes. Such operations are used in many applications [127] and are basic operations in DSM machines using cache coherency mechanisms [46, 106]. There is currently no mathematical model, for collective communication in multicomputer interconnection networks.

Several recent studies [14, 101, 124, 151-153] have revealed that deadlocks occur very infrequently in the network, especially when enough routing freedom is provided [153]. Routing algorithms based on deadlock avoidance, including Duato's algorithm, reserve some virtual channels or routing options to specifically deal with deadlocks, and as a result they are not utilized most of the time. Routing algorithms based on *deadlock recovery* [59] allow messages to use all available virtual channels to cross the network, and efficiently handle infrequently occurred deadlocks. These algorithms are attracting interest in the research community and developing analytical model for them would be beneficial.

Given that integration limits are being achieved, the use of optical and optoelectronic interconnection networks will be a big challenge in the next decade [150]. They have been widely studied [123, 183] and have still many issues to be further explored. Developing analytical tools to study the performance merits of these networks and comparing them with their fully electronic counterparts is an open problem.

References

[1] S. Abraham, Issues in the architecture of direct interconnection schemes for multiprocessors, PhD Thesis, University of Illinois at Urbana-Champaign, 1990.

[2] S. Abraham, Interconnection networks: dimensions in design, *Proceedings Workshop of International Conference on Parallel Processing,* 1996, pp. 45-51.

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

[4] S. Abraham, K. Padmanabhan, Performance of multicomputer networks under Pin-out constraints, *Journal of Parallel and Distributed Computing,* Vol. 12, No. 3, 1991, pp. 237-248.

[5] V. Adve, M.K. Vernon, Performance analysis of mesh interconnection networks with deterministic routing, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 3, 1994, pp. 225-246.

[6] A. Agarwal, Limits on interconnection network performance, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 2, No. 4, 1991, pp. 398-412.

[7] A. Agarwal, R. Bianchini, D. Chaiken, K.L. Johnson, D. Kranz, J. Kubiatowicz, B.H. Lim, K. Mackenzie, D. Yeung, The MIT Alewife machine: architecture and performance, *Proceedings of 22nd Annual International Symposium Computer Architecture*, 1995, pp.2-13.

[8] S.B. Akers, D. Harel, B. Krishnamurthy, The star graph: an attractive alternative to the n-cube, Proceedings of International Conference on Parallel Processing, 1987, pp. 393-400.

[9] S.B. Akers, B. Krishnamurthy, A group-theoretic model for symmetric interconnection networks, *IEEE Transactions on Computers*, Vol. 38, No. 4, 1989, pp. 555-566.

[10] J. Al-Sadi, K. Day, M. Ould-Khaoua, Fault-tolerant routing in hypercubes using probability vectors, *Parallel Computing*, Vol. 27, No. 10, 2001, pp. 1381-1399.

[11] J. Al-Sadi, K. Day, M. Ould-Khaoua, Unsafety vectors: A fault-tolerant routing for k-ary n-cubes, *Microprocessors and Microsystems*, Vol. 25, No. 5, 2001, pp. 239-

246.

[12] J. R. Anderson, S. Abraham, Multidimensional network performance with unidirectional links, *Proceedings of International Conference on Parallel Processing,* 1997, pp. 26-33.

[13] E. Anderson, J. Brooks, C. Grassl, S. Scott, Performance of the Cray T3E multiprocessor, *Proceedings of Supercomputing Conference*, 1997, pp. 19.

[14] K.V. Anjan, T.M. Pinkston, J. Duato, Generalised theory for deadlock-free adaptive wormhole routing and its application to DISHA concurrent, *Proceedings 10th Int. Parallel Processing Symposium (IPPS'96)*, 1996, pp. 815-821.

[15] K. Aoyama, A.A. Chien, The cost of adaptivity and virtual lanes in a wormhole router, *VLSI Design*, Vol. 2, No. 4, 1995, pp. 315-333.

[16] R. Arlanskas, iPSC/2 system: a second generation hypercube, *Proceedings of 3rd ACM Conference on Hypercube Concurrent Computers and Applications*, 1988, pp. 38-42.

[17] Y. Ashir, I.A. Stewart, On embedding cycles in *k*-ary *n*-cubes, *Parallel Processing Letters*, Vol. 7, 1997, pp. 49-55.

[18] Y. Ashir, I.A. Stewart, A. Ahmed, Communication algorithms in *k*-ary *n*-cube interconnection networks, *Information Processing Letters*, Vol. 61, 1997, pp. 43-48.

[19] W.C. Athas, C.L. Seitz, Multicomputers: message passing concurrent computers, *IEEE Computer*, Vol. 21, No. 8, 1988, pp. 9-24.

[20] M.M. Bae, B. Bose, Resource placement in torus-based networks, *IEEE Transactions on Computers,* Vol. 46, No. 10, 1997, pp. 1083-1092.

[21] A. Bakoglu, Circuits, interconnections, packaging for VLSI, Addison-Wesley, 1990.

[22] M. Banikazemi, V. Moorthy, L. Herger, D. K. Panda, and B. Abali, Efficient Virtual Interface Architecture Support for the IBM SP Switch-Connected NT Clusters, *Proceedings International Parallel and Distributed Processing Symposium (IPDPS'2000)*, 2000, pp. 33-42.

[23] D. Basak, D. Panda, Designing clustered multiprocessor systems under packaging and technological advances, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 7, No. 9, 1996, pp. 962-978.

[24] D. Basak, D. Panda, Alleviating consumption channel bottleneck in wormhole-routed *k*-ary *n*-cube systems, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9, No. 5, 1998, pp. 481-496.

[25] S. Bettayeb, On *k*-ary hypercube, *Theoretical Computer Science,* Vol. 140, 1995, pp. 333-339.

[26] L.N. Bhuyan, D.P. Agrawal, Generalized hypercube and hyperbus structures for a computer network, *IEEE Transactions on Computers,* Vol. 33, 1984, pp. 323-333.

[27] T. Boku, K. Itakura, H. Nakamura, K. Nakazawa, CP-PACS: a massively parallel

processor for large scale scientific calculations, *Proceedings of ACM Supercomputing Conference*, 1997, pp. 108-115.

[28] R.V. Boppana, S. Chalasani, C.S. Raghavendra, Resource deadlocks and performance of wormhole multicast routing algorithms, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9, No. 6, 1998, pp. 535-549.

[29] R.V. Boppana, S. Chalasani, A comparison of adaptive wormhole routing algorithms, *Proceedings of 20th Annual International Symposium on Computer Architecture*, 1993, pp.351-360.

[30] R.V. Boppana, S. Chalasani, A framework for designing deadlock-free wormhole routing algorithms, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 7, No. 2, 1996, pp. 169-183.

[31] B. Bose, B. Broeg, Y. Kwon, Y. Ashir, Lee distance and topological properties of *k*-ary *n*-cubes, *IEEE Transactions on Computers,* Vol. 44, 1995, pp. 1021-1030.

[32] Y. Boura, C.R. Das, Modeling virtual channel flow control in *n*-dimensional hypercubes, *Proceedings International Symposium on High Performance Computer Architecture*, 1995, pp. 166-175.

[33] Y. Boura, C.R. Das, A performance model for adaptive routing in hypercubes, *Proceedings of International Workshop on Parallel Processing*, 1994, pp. 11-16.

[34] Y.M. Boura, C.R. Das, Performance analysis of buffering schemes in the routers, *IEEE Transactions on Computers*, Vol. 46, No. 6, 1997, pp. 687-695.

[35] Y.M. Boura, C.R. Das, A class of partially adaptive routing algorithms for n-dimensional meshes, *Proceedings 22nd International Conference on Parallel Processing*, 1993, pp. 175-182.

[36] Y. Boura, Design and analysis of routing schemes and routers for wormhole-routed mesh architectures, Ph.D. Dissertation, Department of Computer Science and Engineering, Penn State University, 1995.

[37] B. Broeg, Topological properties of k-ary n-cubes, PhD Thesis, Department of Computer Science, Oregon State University, June 1995.

[38] M.S. Chen, K.G. Shin, Adaptive fault-tolerant routing in hypercube multicomputers, *IEEE Transactions on Computers*, Vol. 39, No. 12, 1990, pp. 1406-1416.

[39] A.A. Chien, A cost and speed model for *k*-ary *n*-cube wormhole routers, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9, No. 2, 1998, pp. 150-162.

[40] A.A. Chien, J.K. Kim, Planar adaptive routing: low cost adaptive networks for multiprocessors, *Proceedings 19th International Symposium on Computer Architecture*, 1992, pp. 268-277.

[41] J. Choi, J.J. Dongarra, D.W. Walker, Parallel matrix transpose algorithms on distributed memory concurrent computers, *Parallel Computing*, Vol. 21, No. 9, 1995, pp. 1387-1405.

[42] B. Ciciani, M. Colajanni, C. Paolucci, Performance evaluation of deterministic wormhole routing in k-ary n-cubes, *Parallel Computing*, Vol. 24, 1998, pp. 2053-2075.

[43] M. Colajanni, B. Ciciani, F. Quaglia, Performance analysis of wormhole switching with adaptive routing in two-dimensional torus, *Proceedings EuroPar'99, LNCS* 1685, 1999, pp. 165-172.

[44] P.F. Corbett, Rotator graphs: An efficient topology for point-to-point multiprocessor networks, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 3, No. 5, 1992, pp. 622-626.

[45] Cray Research Inc., The Cray T3E scalable parallel processing system, on Cray's Web Page at http://www.cray.com/PUBLIC/product-info/T3E/ CRAY_T3E.html.

[46] D. Dai, D.K. Panda, Effective use of virtual channels in wormhole routed DSM systems, Technical Report, OSU-CISRC-10/97-TR46, Department of Computer and Information Science, The Ohio-State University, 1997.

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

[48] W.J. Dally, Performance analysis of *k*-ary *n*-cubes interconnection networks, *IEEE Transactions on Computers,* Vol. C-39, No. 6, 1990, pp. 775-785.

[49] W.J. Dally, Virtual channel flow control, *IEEE Transactions on Parallel and Distributed Systems,* Vol. 3, No. 2, 1992, pp. 194-205.

[50] W.J. Dally, C.L. Seitz, The torus routing chip, *Journal of Distributed Computing*, Vol. 1, No. 3, 1986, pp. 187-196.

[51] 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, 1993, pp. 66-74.

[52] W.J. Dally, Network and processor architecture for message-driven computers, in *R. Suaya and G. Birtwistle (Eds.): VLSI and parallel computation,* Morgan Kaufmann Publishers, 1990.

[53] W.J. Dally, L.R. Dennison, D. Harris, K. Kan, T. Xanthopolous, The reliable router: a reliable and high-performance communication substrate for parallel computers, *Proceedings 1st Workshop on Parallel Computer Routing and Communication*, 1994, pp. 241-255.

[54] S. Dandamudi, Hierarchical interconnection networks for multicomputer systems, PhD Thesis, Computer Science Department, University of Saskatchewan, Saskatoon, Canada, 1988.

[55] K. Day and A. Al-Ayyoub, The cross product of interconnection networks, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 8, No. 2, 1997, pp. 109-118.

[56] E. Demaine, S. Srinivas, A novel routing algorithm for *k*-ary *n*-cube interconnection networks, *International Journal of High Speed Computing*, Vol. 8,

No. 1, 1996, pp. 81-92.

[57] K.S. Ding, C.T. Ho, J.J. Tsay, Matrix transpose on meshes with wormhole and XY routing, *Discrete Applied Mathematics*, Vol. 83, 1998, pp. 41-59.

[58] J.T. Draper, J. Ghosh, A comprehensive analytical model for wormhole routing in multicomputer systems, *Journal of Parallel and Distributed Computing*, Vol. 23, 1994, pp. 202-214.

[59] J. Duato, C. Yalamanchili, L. Ni, Interconnection networks: an engineering approach, IEEE Computer Society Press, 1997.

[60] J. Duato, P. Lopez, Performance evaluation of adaptive routing algorithms for *k*-ary *n*-cubes, *Proceedings Parallel Computers Routing and Communication*, *LNCS* 853, 1994, pp. 45-59.

[61] J. Duato, On the design of deadlock-free adaptive routing algorithms for multicomputers: design methodologies, *Proceedings Parallel Architectures and languages Europe*, 1991, pp. 390-405.

[62] J. Duato, Deadlock-free adaptive routing algorithms for multicomputers: evaluation of a new algorithm, *Proceedings 3rd IEEE International Symposium on Parallel and Distributed Processing*, 1991, pp. 840-847.

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

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

[65] J. Duato, Why commercial multicomputers do not use adaptive routing, *IEEE Technical Committee on Computer Architecture Newsletter*, 1994, pp. 20-22.

[66] J. Duato, M. Malumbers, Optimal topology for distributed shared-memory multiprocessors: hypercubes again?, *Proceedings of EuroPar'96*, 1996, pp. 205-212.

[67] W. Feng, K. Shin, The effect of virtual channels on the performance of wormhole algorithms in multicomputer networks, University of Michigan, Directed Study Report, May 1994.

[68] A. Ferreira, A.G. vel Lejbman, S.W. Song, Bus-based parallel computers: a viable way for massive parallelism, *Proceedings of Parallel Architectures and Languages*, 1994, pp. 553-564.

[69] M. Fillo, S. W. Keckler, W. J. Dally, N. P. Carter, A. Chang, Y. Gurevich, W.S. Lee, The M-Machine Multicomputer, *Proceedings 28th IEEE/ ACM International Symposium on Microarchitectures*, 1995, pp.146-156.

[70] P. Fragopoulou, S.G. Akl, H. Meijer, Optimal communication primitives on the generalized hypercube network, *Journal of Parallel and Distributed Computing*, Vol. 32, 1996, pp. 173-187.

[71] P. Fragopoulou, S.G. Akl, Efficient algorithms for global data communication on

the multidimensional torus network, *Journal of Parallel and Distributed Computing*, Vol. 24, 1995, pp. 55-71.

[72] P. Fragopoulou, S.G. Akl, Optimal communication algorithms on star graphs using spanning tree constructions, *Journal of Parallel and Distributed Computing*, Vol. 24, No. 1, 1995, pp. 55 - 71.

[73] M.A. Franklin, Pin limitation and partitioning of VLSI interconnection network, *IEEE Transactions on Computers*, Vol. C-36, No. 5, 1987, pp. 547-553.

[74] G. L. Frazier, Buffering and flow control in communication switches for scalable multicomputers, PhD Thesis, University of California, Los Angles, 1995.

[75] H. Fujii, Y. Yasuda, H. Akashi, Y. Inagami, M. Koga, O. Ishihara, M. Kashiyama, H. Wada, T. Sumimoto, Architecture and performance of the Hitachi SR 2201 massively parallel processor system, *Proceedings 11th International Parallel Processing Symposium*, 1997, pp. 233-241.

[76] S.A. Ghozati, H.C. Wasserman, The *k*-ary *n*-cube network: modelling, topological properties and routing strategies, *Computers and Electrical Engineering*, Vol. 25, 1999, pp. 155-168.

[77] C.J. Glass, L.M. Ni, The turn model for adaptive routing, *Proceedings 19th International Symposium Computer Architecture*, 1992, pp. 278-287.

[78] I.S. Gopal, Prevention of store-and-forward deadlock in computer networks, *IEEE Transactions on Communications*, Vol. 33, No. 12, 1985, pp. 1258-1264.

[79] M. Grammatikakis, D. F. Hsu, M. Kratzel, J. F. Sibeyn, Packet routing in fixed-connection networks: a survey, *Journal of Parallel and Distributed Computing*, Vol. 54, pp. 77-132, 1998.

[80] L. Gravano, G.D. Pifarre, P.E. Berman, J.L. Sanz, Adaptive deadlock- and livelock-free routing with all minimal paths in torus networks, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 12, 1994, pp.1233-1251.

[81] R.I. Greenberg, L. Guan, Modelling and comparison of wormhole routed mesh and torus networks, *Proceedings 5th IASTED Conference on Parallel and Distributed Computing Systems*, 1997, at http://www.math.luc.edu/~rig/pubs/.

[82] M. Grosslauser, J.C. Bolot, On the relevance of long-range dependence in network traffic, *Proceedings of the ACM SIGCOMM'96*, 1996, pp. 15-24.

[83] L. Guan, Message routing and problem solving in multiprocessor networks, PhD Thesis, Computer Science Department, University of Maryland, 1997.

[84] W.J. Guan, W.K. Tsai, D. Blough, An analytical model for wormhole routing in multicomputer interconnection networks, *Proceedings International Conference Parallel Processing*, 1993, pp. 650-654.

[85] F.T. Hady, A performance study of wormhole routed networks through analytical modeling and experimentation, PhD Thesis, Electrical Engineering Department, University of Maryland, 1993.

[86] F. Hady, B. L. Menezes, The performance of crossbar-based binary hypercubes, *IEEE Transactions on Computers*, Vol. 44, No. 10, 1995, pp. 1208-1215.

[87] N.C. Hock, Queueing modelling fundamentals, John Wiley and Sons (Eds.) 1996.

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

[89] W. Hsu, P. Yew, Performance evaluation of wire-limited hierarchical networks, *Journal of Parallel and Distributed Computing*, Vol. 41, No. 2, 1997, pp. 156-172.

[90] K. Hwang, Advanced computer architecture: parallelism, scalability and programmability, McGraw-Hill (Ed.), 1993.

[91] Intel Corp., iPSC/1 reference manual, 1986.

[92] Intel Corp., Paragon XP/S product overview, Supercomputer Systems Division, Beaverton, Oregon, 1991.

[93] Intel Corporation, A Touchstone DELTA system description, 1991.

[94] J. Jaja, Load balancing and routing in the hypercube and related networks, *Journal of Parallel and Distributed Computing*, Vol. 14, 1992, pp. 431-435.

[95] J.H. Kai, A.A. Chien, An evaluation of planar adaptive routing, *Proceedings Symposium of Parallel and Distributed Processing*, 1992, pp. 470-478.

[96] S.W. Keckler, The importance of locality and load balancing for multiprocessors, MIT Concurrent VLSI Architecture Memo <ftp://ftp.ai.mit.edu/pub/users/skeckler /cva/sched.ps.Z>, April 1994.

[97] P. Kermani, L. Kleinrock, Virtual cut-through: a new computer communication switching technique, *Computer Networks*, Vol. 3, 1979, pp. 267-286.

[98] R.E. Kessler, J.L Swarszmeier, Cray T3D: a new dimension for Cray research, *Proceedings CompCon*, 1993, pp. 176-182.

[99] J. Kim, C.R. Das, Hypercube communication delay with wormhole routing, *IEEE Transactions on Computers*, Vol. 43, No. 7, 1994, pp. 806-814.

[100] J.H. Kim, A.A. Chien, Network performance under bimodal traffic loads, *Journal of Parallel and Distributed Computing*, Vol. 28, 1995, pp. 43-64.

[101] J. Kim, A. Chien, Z. Liu, Compressionless routing: A framework for adaptive and fault-tolerant routing, *IEEE Transactions Parallel and Distributed Systems*, Vol. 8, No. 3, 1997, pp. 229-244.

[102] J.H. Kim, A.A. Chien, The impact of packetization in wormhole-routed networks, *Proceedings Parallel Architectures Languages*, 1993, at http://www-csag.ucsd. edu/papers/ RoutNetArch-p.html.

[103] L. Kleinrock, On the modeling and analysis of computer networks, *Proceedings of the IEEE*, Vol. 81, No. 8, 1993, pp. 1179-1191.

[104] L. Kleinrock, Queueing systems, Vol. 1, John Wiley and Sons, 1975.

[105] J. Konicek, T. Tilton, A. Veidenbaum, C.Q. Zhu, E.S. Davidson, R. Downing, M. Haney, M. Sharma, P.C. Yew, P.M. Farmwald, D. Kuck, D. Lavery, R. Lindsey, D. Pointer, J. Andrews, T. Beck, T. Murphy, S. Turner, N. Warter, The Organization of the Cedar System, *International Conference Parallel Processing*, 1991, pp. 49-56.

[106] A. Kumar, L. Bhuyan, Evaluating virtual channels for cache-coherence shared-memory multiprocessors, *Proceedings 10th ACM International Conference on Supercomputing*, 1996.

[107] J. Kuskin, D. Ofelt, M. Heinrich, J. Heinlein, R. Simoni, K. Gharachorloo, J. Chapin, D. Nakahira, J. Baxter, M. Horowitz, A. Gupta, M. Rosenblum, J. Hennessy, The Stanford FLASH multiprocessor, *Proceedings 21st International Symposium Computer Architectures*, 1994, pp. 302-313.

[108] J. Laudon, D. Lenoski, The SGI Origin: A ccNUMA highly scalable server, *Proceedings 24th International Symposium on Computer Architecture*, 1997, pp. 241-251.

[109] T.C. Lee, J.P. Hayes, A fault-tolerant communication scheme for hypercube computers, *IEEE Transactions on Computers*, Vol. 41, No. 10, 1992, pp. 1242-1256.

[110] F.T. Leighton, Introduction to parallel algorithms and architectures: arrays, trees, hypercubes, Morgan Kaufmann, 1992.

[111] C. Leiserson, Z. Abuhamdeh, D. Douglas, C. Feynman, M. Ganmukhi, J. Hill, W. Hillis, B. Kuszmaul, M. St. Pierre, D. Wells, M. Wong-Chan, Y. Saw-Wen, R. Zak, The network architecture of the Connection Machine CM-5, *Journal of Parallel and Distributed Computing*, Vol. 33, No. 2, 1996, pp. 145-158.

[112] D. Lenoski, J. Laudon, K. Gharachorloo, W. Weber, A. Gupta, J. Hennessy, M. Horowitz, M.S. Lam., The Stanford DASH multicomputer, *IEEE Computer*, Vol. 25, No. 3, 1992, pp. 63-79.

[113] X. Lin, L.M. Ni, Deadlock-free multicast wormhole routing in multicomputer networks, *Proceedings 18th International Symposium Computer Architecture*, 1991, pp. 116-125.

[114] X. Lin, P. McKinley, L.M. Ni, Deadlock-free multicast wormhole routing in 2D-mesh multicomputers, *IEEE Transactions Parallel and Distributed Systems*, Vol. 5, No. 8, 1994, pp. 793-804.

[115] X. Lin, P. McKinley, L.M. Ni, The message flow model for routing in wormhole-routed networks, *IEEE Transactions Parallel and Distributed Systems*, Vol. 6, No. 7, 1995, pp. 755-760.

[116] D.H. Linder and J.C. Harden, An adaptive and fault tolerant wormhole routing strategy for *k*-ary *n*-cubes, *IEEE Transactions on Computers*, Vol. 40, No. 1, 1991, pp. 2-12.

[117] K.J. Liszka, J.K. Antonio, H.J. Siegel, Problems with comparing interconnection networks: is an alligator better than an armadillo?, *IEEE Concurrency*, Vol. 5, No. 4, 1997, pp. 18-28.

[118] Z. Liu, A.A. Chien, Hierarchical adaptive routing: a framework for fully adaptive and deadlock-free wormhole routing, *Proceedings Symposium Parallel and Distributed Processing*, 1994, pp. 688-695.

[119] P. Lopez, J. Duato, Deadlock-free adaptive routing algorithms for the 3D-torus: limitations and solutions, *Proceedings Parallel Architectures and Languages*,

1993, pp. 684-687.

[120] S. Loucif, M. Ould-Khaoua, L.M. Mackenzie, Analysis of fully-adaptive routing in wormhole-routed tori, *Parallel Computing*, Vol. 25, No. 12, pp. 1477-1487, 1999.

[121] S. Loucif, Performance evaluation of distributed crossbar switch hypermesh, PhD Thesis, Department of Computing Science, University of Glasgow, 1999.

[122] M. Malumbres, J. duato, An efficient implementation of tree-based multicast routing for distributed shared-memory multiprocessors, *Journal of Systems Architecture*, Vol. 46, No.11, 2000, pp. 1019-1032.

[123] G.C. Marsden, P.J. Marchand, P. Harvey, S.C. Esener, Optical transpose interconnection system architectures, *Optics Letters*, Vol. 8, No. 13, 1993, pp. 1083-1085.

[124] J.M. Martinez, P. Lopez, J. Duato, T. M. Pinkston, Software-based deadlock recovery techniques for true fully adaptive routing in wormhole networks, *Proceedings International Conference on Parallel Processing (ICPP'97)*, 1997, pp. 182-189.

[125] P.K. Mckinley, C. Trefftz, Efficient broadcast in all-port wormhole-routed hypercubes, *Proceedings International Conference Parallel Processing*, 1993, pp. 288-291.

[126] P.K. Mckinley, H. Xu, A.H. Esfahanian, L.M. Ni, Unicast-based multicast communication in wormhole-routed networks, *IEEE Transactions Parallel and Distributed Systems*, Vol. 5, No. 12, 1994, pp. 1254-1265.

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

[128] P.M. Merlin, P.J. Shweitzer, Deadlock avoidance in store-and-forward networks, *IEEE Transactions on Communication*, Vol. 28, No. 3, 1980, pp. 345-354.

[129] P. Mohapatra, Wormhole routing techniques in multicomputer systems, *ACM Computing Surveys*, Vol. 30, No. 3, 1998, pp.375-411.

[130] D. Nassimi, S. Sahni, Finding connected components and connected ones on a mesh-connected parallel computer, *SIAM journal on Computing*, Vol. 9, 1980, pp. 744-757.

[131] nCUBE Systems, *N-cube handbook*, 1986.

[132] nCUBE Systems, nCUBE 2: nCUBE 6400 processor manual, 1990.

[133] nCUBE Systems, nCUBE-3, at http//www.ncube.com.

[134] R. Nelson, Probability, stochastic processes, and queuing theory: the mathematics of computer performance modeling, Springer-Verlag, 1995.

[135] L.M. Ni, Should scalable parallel computers support efficient hardware multicast?, *Proceedings Workshop on Challenges for Parallel Processing in International Conference on Parallel Processing*, 1995, pp. 2-7.

[136] L.M. Ni, D.K. Panda, Sea of interconnection networks: what's your choice, *Proceedings International Conference Parallel Processing*, 1994.

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

[138] M. Noakes, D.A. Wallach, W.J. Dally, The J-machine multicomputer: an architectural evaluation, *Proceedings 20th International Symposium Computer Architecture*, 1993, pp. 224-235.

[139] M. Noakes, W.J. Dally, System design of the J-machine, *Proceedings Advanced Research in VLSI,* MIT Press, 1990, pp. 179-192.

[140] S.F. Nugent, The iPSC/2 direct-connect communication technology, *Proceedings Conference Hypercube Concurrent Computers and Applications*, 1988, pp. 51-60.

[141] M. Ould-Khaoua, Hypergraph-based interconnection networks for large lulticomputers, PhD Thesis, Compting Science Department, Glasgow University, 1994.

[142] M. Ould-Khaoua, A performance model of Duato's adaptive routing algorithm in *k*-ary n-cubes, *IEEE Transactions on Computers,* Vol. 48, No. 12, 1999, pp. 1-8.

[143] M. Ould-Khaoua, Message latency in the 2-dimensional mesh with wormhole routing, *Microprocessors and Microsystems*, Vol. 22, No. 9, 1999, pp. 509-514.

[144] D. Panda, Issues in designing efficient and practical algorithms for collective communication on wormhole-routed systems, *Proceedings Workshop on Challenges for Parallel Processing* of *International Conference Parallel Processing*, 1995, pp. 8-15.

[145] D. Panda, S. Singal, R. Kesavan, Multidestination message passing in wormhole *k*-ary *n*-cube networks with base routing conformed paths, *IEEE Transactions Parallel and Distributed Systems,* Vol. 10, No. 1, 2000, pp. 76-96.

[146] K. Park, G. Kim, M. Crovella, On the effect of traffic self-similarity on network performance, *Proceedings of the SPIE International Conference on performance and Control of Network Systems*, 1997, at http://www.cs.bu.edu/pub/barford/ss_lrd.html.

[147] C. Peterson, J. sutton, P. Wiley, iWARP: a 100-MPOS VLIW microprocessor for multicomputers, *IEEE Micro*, Vol. 11, No. 13, 1991.

[148] G.F. Pfister, W.C. Brantley, D.A. George, S.L. Harvey, W.J. Kleinfelder, K.P. McAuliffe, F.A. Melton, V.A. Norton, J. WEiss, The IBM research parallel processor prototype (RP3): introduction and architecture, *Proceedings International Conference Parallel Processing*, 1985, pp. 764-771.

[149] G.J. Pfister, V.A. Norton, Hotspot contention and combining in multistage interconnection networks, *IEEE Transactions on Computers*, Vol. 34, No. 10, 1985, pp. 943-948.

[150] T.M. Pinkston, The GLORI strategy for multiprocessor: Integrating optics into the interconnect architecture, PhD thesis, Department of Electrical Engineering and Computer Science, Stanford University, 1992.

[151] T. M. Pinkston, Flexible and efficient routing based on progressive deadlock recovery, *IEEE Transactions on Computers*, Vol. 48, No. 7, 1999, pp. 649-669.

[152] T. M. Pinkston, S. Warnakulasuriya, Characterization of deadlocks in k-ary n-cube networks, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 10, No. 9, 1999, pp. 38-49.

[153] T. M. Pinkston, S. Warnakulasuriya, On deadlocks in interconnection networks, *Proceedings 24th International Symposium on Computer Architecture (ISCA'97)*, 1997, pp. 38-49.

[154] F.P. Preparata and J. Vuillemin, The cube-connected cycles: a versatile network for parallel computation, *Communications of the ACM*, Vol. 24, 1981, pp.300-309.

[155] K. Qiu, S.G. Akl, On some properties of the star graph, *VLSI Design*, Vol. 2, No. 4, 1995, pp. 389-396.

[156] P. Ramanathan, S. Chalasani, Resource placement with multiple adjacency constraints in *k*-ary *n*-cubes, *IEEE Transactions on Parallel and Distributed Systems,* Vol. 6, No. 5, 1995, pp. 511-519.

[157] S. Ramany, D. Eager, The interaction between virtual channel flow control and adaptive routing in wormhole networks, *Proceedings 8th ACM International Conference on Supercomputing*, 1994, pp.136-145.

[158] S. Ramany, Routing in wormhole networks, PhD Dissertation, Computer Science Department, University of Saskatchewan, 1995.

[159] D.A. Reed, Cost-performance bounds for multicomputer networks, *IEEE Transactions Computers*, Vol. C-32, No. 1, 1984, pp. 1183-1196.

[160] D.A. Reed, R.M. Fujimoto, Multicomputer networks: message-based parallel processing, MIT Press, 1987.

[161] J. Riordan, An introduction to combinatorial analysis, John Wiley & Sons, 1958.

[162] J. Saltz, S. Petiton, H. Berryman, A. Rifkin, Performance effects of irregular communication patterns on massively parallel multiprocessors, *Journal of Parallel and Distributed Computing* Vol. 13, 1991, pp. 202-212.

[163] H. Sarbazi-Azad, M. Ould-Khaoua, L.M. Mackenzie, Employing k-ary n-cubes for parallel Lagrange interpolation, *Parallel Algorithms and Applications*, Vol. 16, 2001, pp. 283-299.

[164] H. Sarbazi-Azad, M. Ould-khaoua, L. Mackenzie and S.G. Akl, Lagrange interpolation on the star graph, to appear, *Journal of Parallel and Distributed Computing*.

[165] H. Sarbazi-Azad, M. Ould-Khaoua, L.M. Mackenzie, A parallel algorithm for Lagrange interpolation on the cube-connected cycles, *Microprocessors and Microsystems,* Vol. 24, No.3, June 2000, pp.135-140.

[166] H. Sarbazi-Azad, L.M. Mackenzie, M. Ould-Khaoua, The effect of the number of virtual channels on the performance of wormhole-routed mesh interconnection networks, *Proceedings UK Performance Engineering Workshop*, 2000, pp. 95-102.

[167] S.L. Scott, J.R. Goodman, The impact of pipelined channels on *k*-ary *n*-cube networks, *IEEE Transactions Parallel and Distributed Systems*, Vol. 5, No. 1, 1994, pp. 2-16.

[168] H.J. Seigel, Interconnection networks for large-scale parallel Processing, McGraw-Hill, New York, 1990.

[169] H.J. Siegel, C.B. Stunkel, Trends in parallel machine interconnection networks, *IEEE Computing in Science and Engineering*, 1996, pp. 69-71.

[170] C.L. Seitz, The Cosmic cube, *Communication of the ACM*, Vol. 28, No. 1, 1985, pp. 22-33.

[171] C.L. Seitz, The hypercube communication chip, Department of Computer Science, CalTech, Display File 5182:DF:85, 1985.

[172] C.L. Seitz, Mosaic C: an experimental fine-grain multicomputer, *Proceedings of Future Tendencies in Computer Science, Control, and Applied Mathematics*, 1992, pp. 69-85.

[173] C.B. Stunkel, Commercially viable MPP networks, Research Report RC20444 (4-29-96), IBM Research Division, T.J. Watson Research Center, New York, 1996.

[174] C. Su and K.G. Shin, Adaptive deadlock-free routing in multicomputers using one extra channel, *Proceedings International Conference Parallel Processing*, 1993, pp. 175-182.

[175] M.N.S. Swamy, K. Thulasiraman, Graphs, networks and algorithms, John Wiley & Sons, 1981.

[176] A.S. Tanenbaum, Computer networks, Prentice-Hall, 1989.

[177] C.D. Thompson, A Complexity theory for VLSI, PhD Thesis, Computer Science Department, Carnegie-Mellon University, 1980.

[178] S. Warnakulasuriya, T. M. Pinkston, A Formal Model of Message Blocking and Deadlock Resolution in Interconnection Networks, *IEEE Transactions Parallel and Distributed Systems*, Vol. 11, No. 3, 2000, pp. 212-229.

[179] W.A. Whiteworth, Choice and chance, Cambridge University Press, 1901.

[180] C.S. Yang, Y.M. Tsai, S.L. Chi, S.B. Shi, Adaptive wormhole routing in *k*-ary *n*-cubes, *Parallel Computing*, Vol. 21, 1995, pp. 1925-1943.

[181] J. Wu, Adaptive fault-tolerant routing in cube-based multicomputers using safety vectors, *IEEE Transactions Parallel and Distributed Systems*, Vol. 9, No. 4, 1998, pp. 321-334.

[182] Y. Yasuda, H. Fujii, H. Akashi, Y. Inagami, T. Tanaka, J. Nakagoshi, H. Wada, T. Sumimoto, Deadlock-free fault tolerant routing in the multidimensional crossbar network and its implementation for the Hitachi SR2201, *Proceedings of 11th International Parallel Processing Symposium*, 1997, pp. 346-352.

[183] F. Zane, P. Marchand, R. Paturi, S. Esener, Scalable network architectures using the optical transpose interconnection system (OTIS), *Proceedings International Conference on Massively Parallel Processing Using Optical Interconnections (MPPOI'96)*, 1996, pp. 114-121.

Publications during the course of this research

CHAPTER 2

H. Sarbazi-Azad, M. Ould-Khaoua, L. Mackenzie, S. Akl, **On some properties of *k*-ary *n*-cubes**, *Proceedings of IEEE International Conference on Parallel and Distributed Systems (ICPADS'2001)*, 26-29 June, 2001, KyongJu City, Korea, pp. 517-524.

H. Sarbazi-Azad, M. Ould-Khaoua, L. Mackenzie, S. Akl, **On the combinatorial properties of *k*-ary *n*-cube interconnection networks**, under review in *Networks*.

CHAPTER 3

H. Sarbazi-Azad, M. Ould-Khaoua, L. Mackenzie, **An accurate analytical model of adaptive wormhole routing in *k*-ary *n*-cube interconnection networks**, *Performance Evaluation,* Vol. 43, No. 2-3, 2001, pp. 165-179.

H. Sarbazi-Azad, M. Ould-Khaoua, L. Mackenzie, **Performance analysis of *k*-ary *n*-cubes with fully adaptive routing,** *Proceedings of IEEE 7th International Conference on Parallel and Distributed Systems (ICPADS'2000)*, Iwate, Japan, July 4-7, 2000, pp. 249-255.

CHAPTER 4

H. Sarbazi-Azad, M. Ould-Khaoua, L. Mackenzie, **Analytical modeling of wormhole-routed *k*-ary *n*-cubes in the presence of hotspot traffic**, *IEEE Transactions on Computers,* Vol. 50, No. 7, 2001, pp. 623-634.

M. Ould-Khaoua, H. Sarbazi-Azad, **An analytical model of adaptive wormhole routing in hypercubes in the presence of hotspot traffic,** *IEEE Transactions on Parallel and Distributed System,* Vol. 12, No.3, 2001, pp. 283-292.

H. Sarbazi-Azad, M. Ould-Khaoua, L. Mackenzie, **On the performance of adaptive wormhole routing in the bidirectional 2D torus interconnection network: A hotspot analysis,** *Microprocessors and Microsystems,* Vol. 25, No. 6, 2001, pp. 277-285.

H. Sarbazi-Azad, M. Ould-Khaoua, L. Mackenzie, **An analytical model of fully-adaptive wormhole-routed *k*-ary *n*-cubes in the presence of hotspot traffic,** *Proceedings of IEEE/ACM International Parallel and Distributed Processing Symposium (IPDPS'2000)*, Cancun, Mexico, May 1-5, 2000, pp. 605-610.

H. Sarbazi-Azad, M. Ould-Khaoua, L. Mackenzie, **Hotspot analysis in wormhole-routed tori,** *Proceedings of 19th IEEE International Performance, Computing, and Communications Conference (IEEE-IPCCC'2000)*, Phoenix, Arizona, U.S.A., Feb. 20-22, 2000, pp. 337-343.

H. Sarbazi-Azad, M. Ould-Khaoua, L. Mackenzie, **Performance modeling of adaptive wormhole-routed multi-computers with hotspot traffic**, *Proceedings of CSICC'2000*, Tehran, Iran, March 7-9, 2000.

CHAPTER 5

H. Sarbazi-Azad, M. Ould-Khaoua, L. Mackenzie, **Message latency in hypercubes in the presence of matrix-transpose traffic**, *The Computer Journal,* Vol. 43, No.5, 2000, pp. 411-419.

H. Sarbazi-Azad, M. Ould-Khaoua, L. Mackenzie, **An analytical model of adaptive wormhole routing in hypercubes with bit-reversal traffic pattern**, *Parallel Computing,* Vol. 27, No. 13, 2001, pp.1801-1816.

H. Sarbazi-Azad, M. Ould-Khaoua, L. Mackenzie, **A performance model of adaptive routing in *k*-ary *n*-cubes with matrix-transpose traffic,** *Proceedings of International Conference on Parallel Processing (ICPP2000)*, IEEE Computer Society Press, Toronto, Canada, August 21-24, 2000, pp. 345-352.

H. Sarbazi-Azad, M. Ould-Khaoua, L. Mackenzie, **An analytic model for communication latency in wormhole-switched *k*-ary *n*-cube interconnection networks with digit-reversal traffic,** *Proceedings of International Symposium on High Performance Computing (ISHPC2K),* Tokyo, Japan, Oct.16-18, 2000, LNCS 1940, Springer-Verlag, 2000, pp. 218-229.

H. Sarbazi-Azad, M. Ould-Khaoua, L. Mackenzie, **Message latency in wormhole-switched *k*-ary *n*-cubes under digit-reversal traffic workload,** to appear, *Journal of Supercomputing,* 2002.

H. Sarbazi-Azad, M. Ould-Khaoua, L. Mackenzie, **Modeling *k*-ary *n*-cubes with matrix-transpose traffic,** under review in *Journal of Parallel and Distributed Computing.*

CHAPTER 6

H. Sarbazi-Azad, M. Ould-Khaoua, L. Mackenzie, **A more realistic comparative analysis of wormhole-routed tori and hypercubes**, to appear, *Proc. IEEE & ACM Int. Parallel & Distributed Processing Symposium (IPDPS'2000) Workshops*, April 15-19, 2002, Fort Lauderdale, Florida, U.S.A.

H. Sarbazi-Azad, M. Ould-Khaoua, L. Mackenzie, **Towards a more realistic comparative analysis of multicomputer networks**, to appear, *Computation and Concurrency: Practice and Experience,* 2002.

H. Sarbazi-Azad, M. Ould-Khaoua, L. Mackenzie, **Performance comparison of interconnection networks using a new cost model**, under review in a special issue of *Cluster Computing.*

H. Sarbazi-Azad, M. Ould-Khaoua, L. Mackenzie, **A new cost model for comparative analysis of wormhole-routed tori and hypercubes with uniform traffic**, to appear, to appear, *Proceedings IASTED Int'l Conf. Parallel and Distributed Computing and Networks (PDCN'2002)*, 18-21 Feb., 2001, Innsbruck, Austria.